|  |  |
| --- | --- |
| 成绩 |  |

装

订

线

网络空间安全与计算机学院实验报告

实验课程名称**：神经网络与深度学习实验**

实验项目名称： 卷积神经网络（1）卷积算子

学生姓名： 刘新媛

专 业： 人工智能

学 号： 20221205037

实验地点： C1-320

实验时间： 2024.11.6第7-8节

指导教师： 魏勇刚

预习报告部分

1. **实验目的：**使用Pytorch实现卷积运算和卷积算子
2. **实验原理：**

1、卷积的主要功能是在一个图像（或特征图）上滑动一个卷积核，通过卷积操作得到一组新的特征。

对于一个输入矩阵X和一个滤波器（卷积核）W,他们的卷积为公式1-1所示：

（1-1）

2、二维卷积运算中，**零填充**是指在输入矩阵周围对称地补上P个0。

对输入用零进行填充使得其尺寸变大。根据卷积的定义，如果不进行填充，当卷积核尺寸大于1时，输出特征会缩减。对输入进行零填充则可以对卷积核的宽度和输出的大小进行独立的控制。

对于一个输入矩阵X和一个滤波器（卷积核）W,步长为S，对输入矩阵进行零填充，那么最终输出矩阵大小则为公式2-1、2-2所示：

（2-1）

（2-2）

3、用拉普拉斯算子对物体边缘进行提取，拉普拉斯算子为一个大小为3×3的卷积核，中心元素值是8，其余元素值是−1，如图3-1所示：

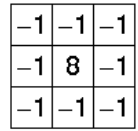


图 3-1八邻域的 拉普拉斯卷积核

1. **实验内容：**

使用pytorch实现：

1.自定义二维卷积算子

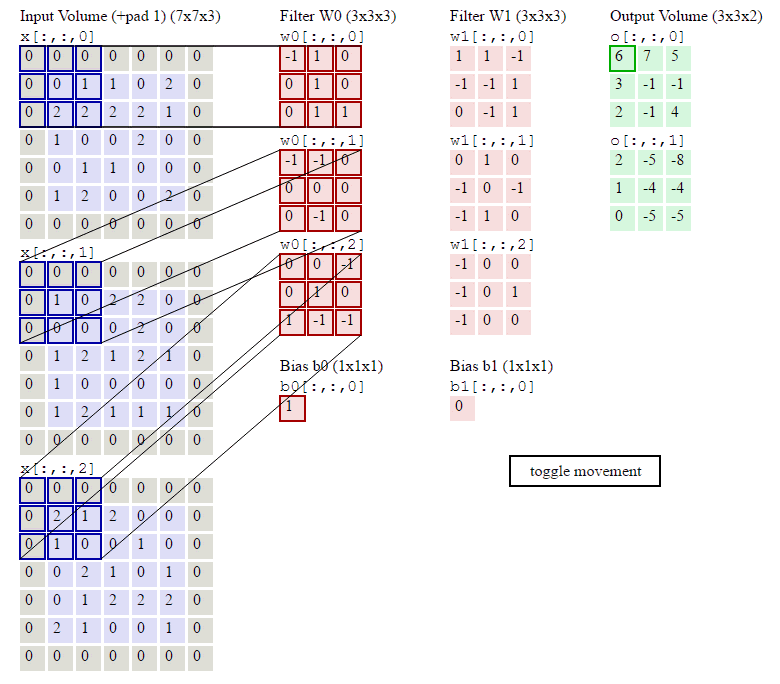
2.自定义带步长和零填充的二维卷积算子

3.实现图像边缘检测

4.自定义卷积层算子和汇聚层算子

5.学习torch.nn.Conv2d()、torch.nn.MaxPool2d()；torch.nn.avg\_pool2d()，简要介绍使用方法。

6.分别用自定义卷积算子和torch.nn.Conv2d()编程实现下面的卷积运算



1. **实验所用设备：**pycharm、pytorch框架

实验报告部分

1. **实验步骤：**

**1.自定义二维卷积算子**

实现自定义的二维卷积算子，最关键的一步为通过两个嵌套循环遍历输出矩阵的每个位置，并计算对应位置的卷积值。循环内部使用X[:, i:i + u, j:j + v]提取输入矩阵中当前卷积核覆盖的子矩阵，并与卷积核self.weight相乘，然后使用torch.sum在指定维度上求和，得到当前位置的卷积结果。

i:i + u：这部分表示从**当前行的索引i**开始，选取u个连续的元素，其中u是卷积核的高度。这样，就从当前行中提取了一个高度为u的子矩阵。

j:j + v：这部分表示从**当前列的索引j**开始，选取v个连续的元素，其中v是卷积核的宽度。就从当前列中提取了一个宽度为v的子矩阵。

**2.自定义带步长和零填充的二维卷积算子**

实现带步长和零填充的二维卷积算子，在第一问的基础上加上步长和零填充参数默认stride=1, padding=0，先对输入矩阵X进行零填充。创建一个新的矩阵new\_X，其尺寸为[B, M + 2 \* padding, N + 2 \* padding]，再根据填充后的输入矩阵，按照公式2-1、2-2计算输出矩阵的高度和宽度。在每次循环中，根据步长计算子矩阵的起始位置，并从填充后的输入矩阵new\_X中提取与卷积核大小相同的子矩阵。将这个子矩阵与卷积核进行元素乘法操作，然后对结果进行求和，得到当前位置的卷积值。

**3.实现图像边缘检测**

实验中我用到的图像为“carmeramen.tif”灰度图，如图3-2所示，对其进行边缘检测关键的步骤就是把图像转换为符合卷积操作的输入矩阵：使用图像处理库（如PIL）读取图像文件---->将其大小调整为256x256---->将图像数据转换为NumPy数组，以便进行数值计算--->将NumPy数组转换为PyTorch张量，因为PyTorch的卷积操作是基于张量的---->添加批次维度，经过以上处理后就可以像问题1、2进行卷积运算了，最后再可视化灰度图，其边缘检测后的结果如图3-3所示。

Cameraman 

图3-2 carmeramen 图3-3 carmeramen边缘检测图

**4.自定义卷积层算子和汇聚层算子**

**（1）卷积层算子：**

在前面实现的二维卷积运算中，卷积的输入数据是二维矩阵。但实际应用中，一幅大小为M×N的图片中的每个像素的特征表示不仅仅只有灰度值的标量，通常有**多个特征**，可以表示为D维的向量，因此，图像上的卷积操作的输入数据通常是一个**三维张量**，分别对应了图片的高度M、宽度N和深度D，其中深度D也被称为**输入通道数D**，所以一个卷积层通常会组合多个不同的卷积核来提取特征，卷积运算后会输出多张特征图，输出特征图的个数通常将其称为**输出通道数P**。其中多输入通道的卷积运算如图4-1所示，多输出通道的卷积运算如图4-2所示。

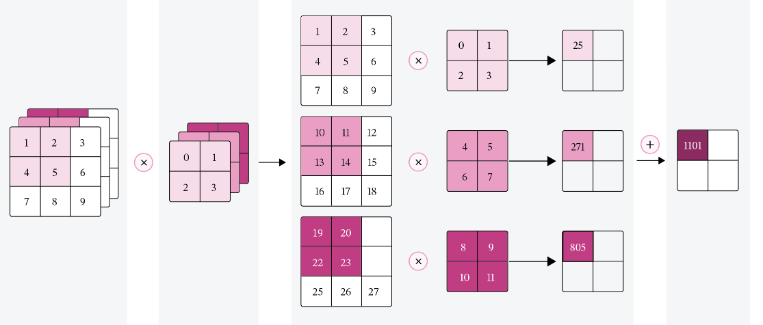


图 4-1 多输入通道的卷积运算（[图源paddle](https://aistudio.baidu.com/projectdetail/8499646)）

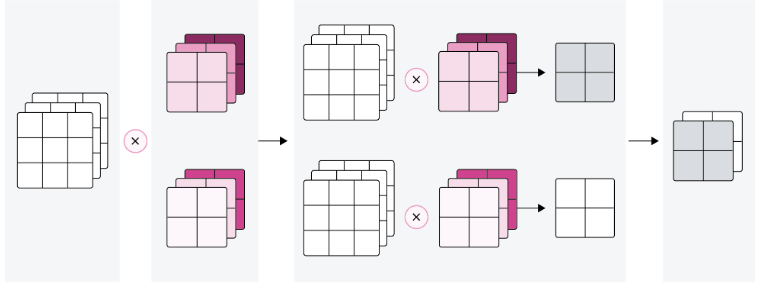


图 4-2 多输出通道的卷积运算（[图源paddle](https://aistudio.baidu.com/projectdetail/8499646)）

实现多通道的卷积层，在之前二维卷积的基础上添加了single\_forward方法来实现对每个单个输入通道的卷积操作，然后在forward方法中，对每个输出通道，遍历每个输入通道，并**对每个输入通道执行single\_forward方法**，得到每个输入通道的卷积结果，并将所有输入通道的卷积结果相加，最后将所有输出通道的特征图堆叠起来，形成最终的输出张量。

1. **汇聚层算子：**

汇聚层的作用是进行**特征选择**，降低特征数量，从而减少参数数量，汇聚之后特征图会变得更小，可以有效地减小神经元的个数，节省存储空间并提高计算效率。常用的汇聚方法有是：**平均汇聚和最大汇聚**。汇聚层输出的计算尺寸与卷积层一致，即公式2-1、2-2所示。

**平均汇聚：**将输入特征图划分为2×2大小的区域，对每个区域内的神经元**活性值取平均值**作为这个区域的表示；

**最大汇聚：**使用输入特征图的每个子区域内所有神经元的**最大活性值**作为这个区域的表示。

实现汇聚层，我在构造函数\_\_init\_\_中先定义接了收三个参数：size（池化窗口的大小，默认为2x2），mode（池化模式，'max'或'avg'，默认为'max'），stride（步长，默认为1），再在forward函数中实现汇聚：计算出汇聚后的特征图像高和宽--->使用两个嵌套循环遍历输出特征图的每个位置并根据mode参数执行最大池化或平均池化。

**5.学习****torch.nn.Conv2d()、torch.nn.MaxPool2d()；torch.nn.AvgPool2d()，简要介绍使用方法。**

**torch.nn 是 PyTorch 中用于构建神经网络的模块（不是函数）**

* **torch.nn.Conv2d()**

**功能：**Conv2d 是二维卷积层，将输入数据与滤波器（卷积核）进行卷积，生成特征图。

**参数（**parameters**）：**

in\_channels：输入数据的通道数。

out\_channels：输出数据的通道数，即卷积核的数量。

kernel\_size：卷积核的大小。

stride：卷积的步长，默认为1。

padding：边缘填充，默认为0。

（除此之外，还有dilation、groups、bias、padding\_mode等参数）

**变量（**Variables**）：**在训练过程中会被调整

weight: 模块可以学习的权重

bias:模块可以学习的偏差

**使用方法：**导入torch.nn-->创建Conv2d层实例-->准备输入数据-->将输入数据传入

* **torch.nn.MaxPoo2d()**

**功能：**二维最大池化层，用于下采样，减少特征图的空间维度，保留最重要的特征

**参数（**parameters**）：**

kernel\_size：池化窗口的大小。

stride：池化的步长，默认等于kernel\_size。

padding：边缘填充，默认为0。

（除此之外，还有dilation、return\_indices、ceil\_mode等参数）

**变量：**无，不包含可学习的参数，因此没有像卷积层那样的权重（weights）和偏置（biases）变量。

**使用方法：**导入torch.nn-->创建MaxPool2d层实例-->通常输入数据为卷积层输出-->将输入数据传入最大池化层

* **torch.nn.AvgPool2d()**

**功能：**二维平均池化层，与最大池化类似，但是它计算池化窗口内所有元素的平均值

**参数（**parameters**）：**

kernel\_size：池化窗口的大小。

stride：池化的步长，默认等于kernel\_size。

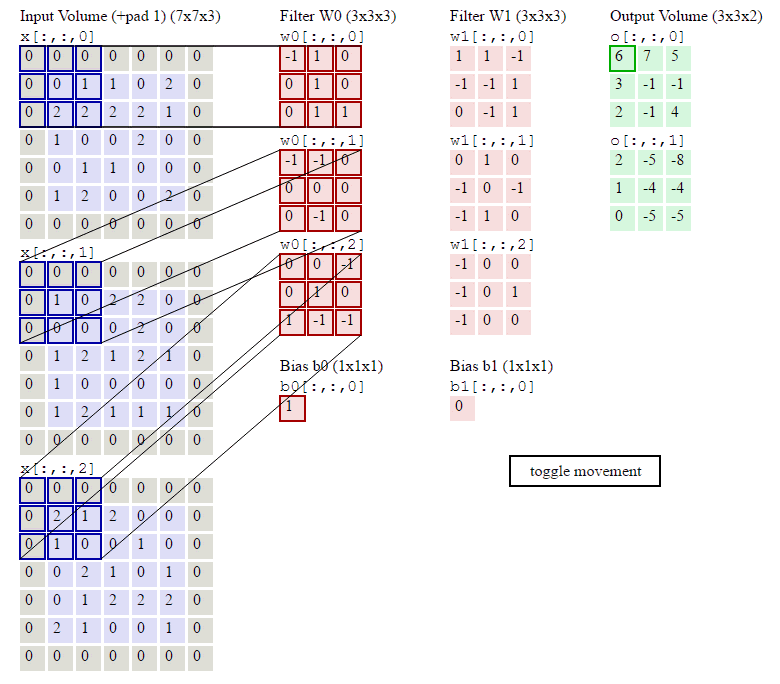
padding：边缘填充，默认为0。

（除此之外，还有dilation、return\_indices、ceil\_mode等参数）

**变量：**无，不包含可学习的参数，因此没有像卷积层那样的权重（weights）和偏置（biases）变量。

**使用方法：**导入torch.nn-->创建Pool2d层实例-->通常输入数据为卷积层输出-->将输入数据传入平均池化层

**6.分别用自定义卷积算子和torch.nn.Conv2d()编程实现下面的卷积运算**



使用问题4创建的卷积层算子，按照上图定义输入特征图-->定义卷积核权重-->定义卷积核偏置-->分别调用自定义的卷积层算子和pytorch自带的Conv2d卷积操作实现卷积操作。

1. **实验数据及结果分析：**

**[点击查看实验代码及结果分析](https://blog.csdn.net/qq_73704268/article/details/143575796?spm=1001.2014.3001.5501)**

1. **实验结论：**
2. 通过对“carmeramen.tif”灰度图的边缘检测实验，结果符合预期，证实了卷积算子在特征提取中的作用。
3. 通过自定义二维卷积算子和带步长、零填充的二维卷积算子，成功实现了卷积操作且结果与调用的API结果一致。
4. 对torch.nn.Conv2d()、torch.nn.MaxPool2d()和torch.nn.AvgPool2d()的学习，详细了解了PyTorch框架中这些内置层的使用方法和参数配置。
5. **总结及心得体会：**
6. 这次实验比较简单，主要是为后序搭建卷积神经网络打基础，主要是手动实现了一下卷积层和池化层，就像老师说的一样，调函数其他专业的也会，但是作为AI专业的也应该知道函数内部是如何实现的。一个之前不知道知识就是pytorch自带的卷积操作中权重和偏置变量也可以自己修改为固定的数，只需要用.weight、.bias直接访问卷积层变量就可以，还学到了如何使用nn.Parameter()创建参数，pytorch中的nn模块中已经定义好很多网络层，所以我们搭建模型时可以直接使用，但是实验中自定义网络层时就需要自定义**可学习参数**，那就用到了nn.Parameter()这个函数，该函数会创建一个可学习的参数矩阵，该矩阵是默认可梯度求导的。
7. 有一个马虎的点就是在写第二题时，我使用步长为1，零填充为1时卷积输出的图像形状一直和原图不一样，检查了好久forward方法┭┮﹏┭┮，最后才发现原来是创建的卷积核没有改（依旧是问题一2\*2的卷积核），而保证图像大小不变是需要3\*3卷积核的。

|  |
| --- |
| **参考链接：** |
| [pytorch官方文档-Conv2d的介绍](https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html" \l "torch.nn.Conv2d) |
| [pytorch官方文档 --pooling层介绍](https://pytorch.org/docs/stable/nn.html" \l "pooling-layers) |
| [paddle卷积神经网络介绍-](https://aistudio.baidu.com/projectdetail/8499646) |
| [pytorch中卷积操作的初始化方法(kaiming\_uniform\_详解)](https://blog.csdn.net/weixin_44503976/article/details/117284487?ops_request_misc=&request_id=&biz_id=102&utm_term=Kaiming%E5%9D%87%E5%8C%80%E5%88%9D%E5%A7%8B%E5%8C%96%E2%80%99&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-3-117284487.142^v100^pc_search_result_base7&spm=1018.2226.3001.4187" \o "pytorch中卷积操作的初始化方法(kaiming_uniform_详解))  ---学会了修改pytorch自带卷积操作的变量 |
| [作业六--卷积](https://blog.csdn.net/qq_73704268/article/details/143089937?spm=1001.2014.3001.5502) ---之前写过的作业，里面有一部分是关于边缘检测的 |
| [pytorch中nn.Parameter()使用方法](https://blog.csdn.net/m0_47256162/article/details/127822519?ops_request_misc=&request_id=&biz_id=102&utm_term=%E6%A8%A1%E5%9E%8B%E8%AE%AD%E7%BB%83%E6%97%B6nn.Parameter()%E8%AF%BB%E5%8F%96%E5%8F%82%E6%95%B0&utm_medium=distribute.pc_search_result.none-task-b) ----简洁干练，清楚的表达了nn.Parameter的使用方法 |