《深度学习及其应用》课程实验报告

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 使用Python完成二维卷积和池化功能模块实验 | | | 实验序号 | 1 | 实验日期 | 20231011 |
| 姓 名 | **赵诗文** | **院系** | 计算机学院 | 班 级 | 21104502 | 学 号 | 2110450209 |
| 专 业 | **人工智能** | | | 指导教师 | 林涛 | 成 绩 | 合格 |
| **一、实验目的和要求**  目的：了解卷积操作的原理，熟悉CNN中卷积和池化操作的具体流程，为深度学习网络的进一步学习打好基础，增强Python的编程能力。  要求：  1、利用Python编程语言，使用numpy工具包，实现两个二维数据的卷积操作  2、卷积类型为same，卷积核大小为3x3，步长为1，待处理二维数据大小可以自己决定，可以是一副图片，也可以是一个数字矩阵；池化核大小为2x2，使用最大池化方法，池化步长为2  3、完成代码编写，打印出实验结果；  4、完成实验报告内容，提交报告。 | | | | | | | |
| **二、实验步骤**  **可以在实训平台上完成，也可以在实验室电脑上或者自己的电脑上完成。**  **如果使用实验平台，使用账号登录到实训平台，选择课程《深度学习及其应用》**  **实验平台 10.2.253.243:10010 ，选择课程《深度学习及其应用》，**  实验过程截图放到实验报告中。  附参考数据：  卷积核可以使用 1 0 -1  1 0 -1  1 0 -1  待卷积信号可以使用如下的8x8矩阵：  [0, 0, 0, 0, 0, 0, 0, 0],  [1, 1, 1, 1, 1, 1, 1, 1],  [2, 2, 2, 2, 2, 2, 2, 2],  [3, 3, 3, 3, 3, 3, 3, 3],  [4, 4, 4, 4, 4, 4, 4, 4],  [5, 5, 5, 5, 5, 5, 5, 5],  [6, 6, 6, 6, 6, 6, 6, 6],  [7, 7, 7, 7, 7, 7, 7, 7]    什么是same方式的卷积  same卷积是介于full卷积和valid卷积之间的一种卷积方式，其特点是卷积前后特征图的尺寸不变。由于same卷积的特点，其Padding值是固定设置的，如图1.1 所示    图1.1 same方式卷积操作 | | | | | | | |
| **三、实验结果与分析**  （截图和代码放到这里）  截图：    代码：  # 导入相关库  import numpy as np  from scipy import signal  # 定义一个3×3的卷积核  kernel = np.array([[1, 0, -1],  [1, 0, -1],  [1, 0, -1]])  # 定义一个待处理的二维数据  data = np.array([[0, 0, 0, 0, 0, 0, 0, 0],  [1, 1, 1, 1, 1, 1, 1, 1],  [2, 2, 2, 2, 2, 2, 2, 2],  [3, 3, 3, 3, 3, 3, 3, 3],  [4, 4, 4, 4, 4, 4, 4, 4],  [5, 5, 5, 5, 5, 5, 5, 5],  [6, 6, 6, 6, 6, 6, 6, 6],  [7, 7, 7, 7, 7, 7, 7, 7]])  # 使用signal.convolve2d进行卷积  result = signal.convolve2d(data, kernel, mode='same')  #输出卷积结果  print("result：\n", result)  # 定义了一个名为max\_pooling的最大池化函数  def max\_pooling(input\_matrix, kernel\_size=2, stride=2):  output\_rows = (input\_matrix.shape[0] - kernel\_size) // stride + 1  output\_cols = (input\_matrix.shape[1] - kernel\_size) // stride + 1  output\_matrix = np.zeros((output\_rows, output\_cols))  for i in range(output\_rows):  for j in range(output\_cols):  output\_matrix[i, j] = np.max(input\_matrix[i\*stride:i\*stride+kernel\_size, j\*stride:j\*stride+kernel\_size])  return output\_matrix  # 使用max\_pooling函数对卷积结果进行池化  output\_matrix = max\_pooling(result)  #输出池化结果  print("output\_matrix：\n", output\_matrix)  **使用scipy库中的signal.convolve2d函数对data和kernel进行卷积操作。这个函数的第一个参数是要进行卷积操作的数据，第二个参数是卷积核，第三个参数是卷积模式。在这个例子中，卷积模式被设置为'same'，这意味着输出的结果大小与输入的数据大小相同。定义了一个名为max\_pooling的最大池化函数。这个函数接受三个参数：输入矩阵（input\_matrix）、池化窗口的大小（kernel\_size，默认为2）和步长（stride，默认为2）。函数的主要作用是将输入矩阵按照给定的池化窗口大小和步长进行池化操作，即取每个池化窗口内的最大值作为输出矩阵对应位置的值。** | | | | | | | |