

问题

写代码实现卷积操作

问题背景

一次面试失败得来的深刻教训，自己的学习太不扎实了，理论基础薄弱，一来真格就不会。其实这个问题之前在看面经的时候就有说到过，虽然理论弄明白了，但还是心存侥幸没有动手把代码写出来.....

问题解答

传统卷积运算是将卷积核以滑动窗口的方式在输入图上滑动，当前窗口内对应元素相乘然后求和得到结果，一个窗口一个结果。**相乘然后求和恰好也是向量内积的计算方式**，所以可以将每个窗口内的元素拉成向量，通过向量内积进行运算，多个窗口的向量放在一起就成了矩阵，每个卷积核也拉成向量，多个卷积核的向量排在一起也成了矩阵，于是，卷积运算转化成了矩阵乘法运算。下图很好地演示了矩阵乘法的运算过程：

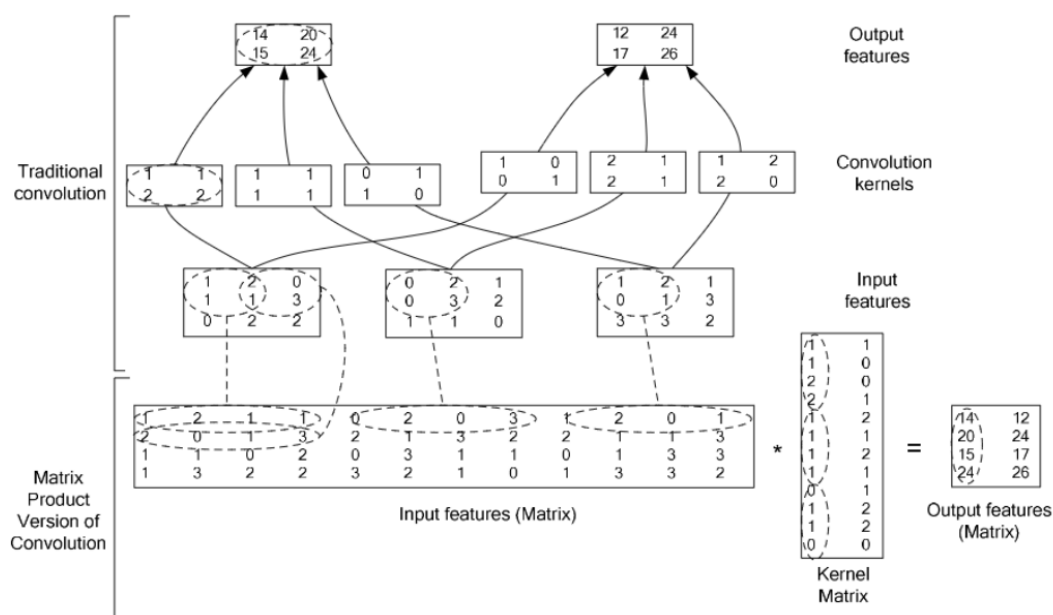
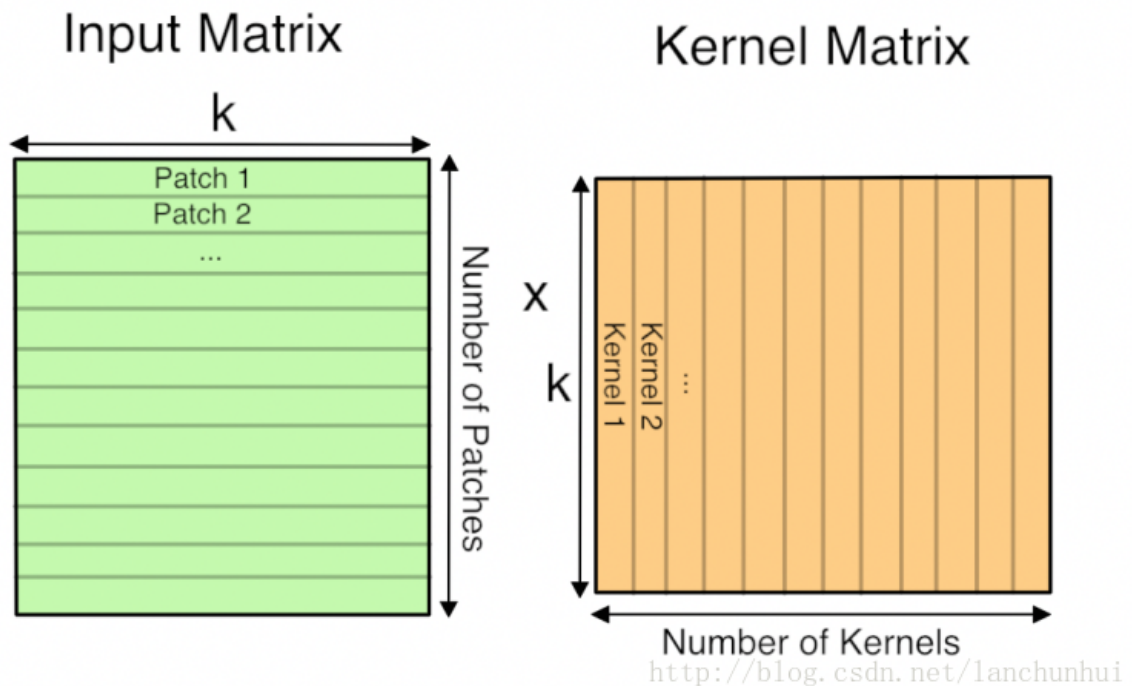
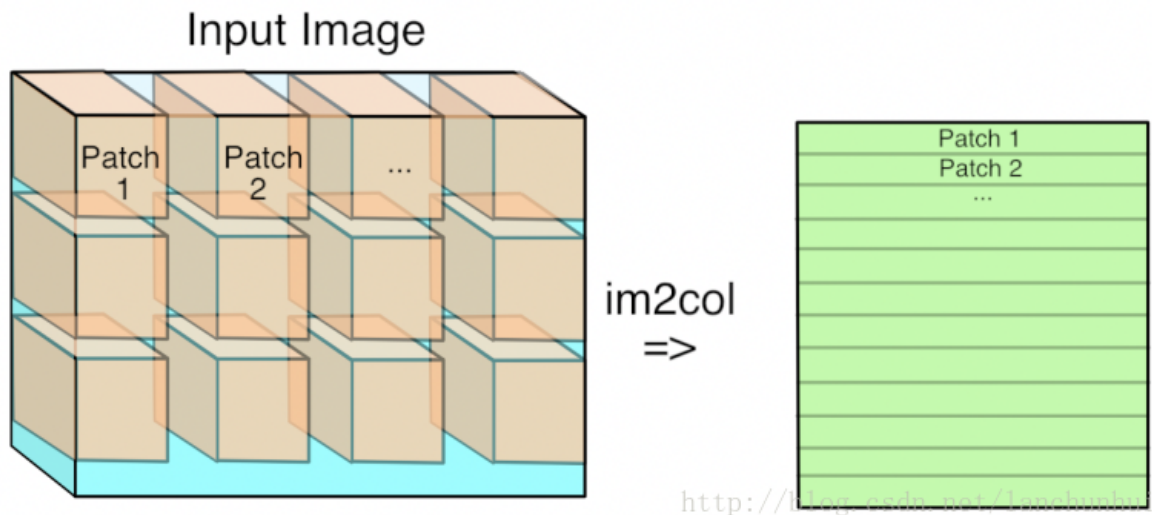


Figure 2. Example convolution operations in a convolutional layer (biases, sub-sampling, and non-linearity omitted). The top figure presents the traditional convolution operations, while the bottom figure presents the matrix version.

将卷积运算转化为矩阵乘法，从乘法和加法的运算次数上看，两者没什么差别，但是转化成矩阵后，运算时需要的数据被存在连续的内存上，这样访问速度大大提升（cache），同时，矩阵乘法有很多库提供了高效的实现方法，像BLAS、MKL等，转化成矩阵运算后可以通过这些库进行加速。

缺点呢？这是一种空间换时间的方法，消耗了更多的内存——转化的过程中数据被冗余存储。

还有两张形象化的图片帮助理解：



代码实现

太久没写python代码，面试的时候居然想用c++来实现，其实肯定能实现，但是比起使用python复杂太多了，所以这里使用python中的numpy来实现。

一、滑动窗口版本实现

```

1  #!/usr/bin/env python3      #加上这一句之后，在终端命令行模式下就可以直接输入这个文件的
    名字后运行文件中的代码
2  # -*- coding = utf-8 -*-
3  import numpy as np
4
5  # 为了简化运算，默认batch_size = 1
6  class my_conv(object):
7      def __init__(self, input_data, weight_data, stride, padding = 'SAME'):
8          self.input = np.asarray(input_data, np.float32)
9          self.weights = np.asarray(weight_data, np.float32)
10         self.stride = stride
11         self.padding = padding
12     def my_conv2d(self):
13         """

```

```

14         self.input: c * h * w # 输入的数据格式
15         self.weights: c * h * w
16         """
17         [c, h, w] = self.input.shape
18         [kc, k, _] = self.weights.shape # 这里默认卷积核的长宽相等
19         assert c == kc # 如果输入的channel与卷积核的channel不一致即报错
20         output = []
21         # 分通道卷积, 最后再加起来
22         for i in range(c):
23             f_map = self.input[i]
24             kernel = self.weights[i]
25             rs = self.compute_conv(f_map, kernel)
26             if output == []:
27                 output = rs
28             else:
29                 output += rs
30         return output
31     def compute_conv(self, fm, kernel):
32         [h, w] = fm.shape
33         [k, _] = kernel.shape
34
35         if self.padding == 'SAME':
36             pad_h = (self.stride * (h - 1) + k - h) // 2
37             pad_w = (self.stride * (w - 1) + k - w) // 2
38             rs_h = h
39             rs_w = w
40         elif self.padding == 'VALID':
41             pad_h = 0
42             pad_w = 0
43             rs_h = (h - k) // self.stride + 1
44             rs_w = (w - k) // self.stride + 1
45         elif self.padding == 'FULL':
46             pad_h = k - 1
47             pad_w = k - 1
48             rs_h = (h + 2 * pad_h - k) // self.stride + 1
49             rs_w = (w + 2 * pad_w - k) // self.stride + 1
50         padding_fm = np.zeros([h + 2 * pad_h, w + 2 * pad_w], np.float32)
51         padding_fm[pad_h:pad_h+h, pad_w:pad_w+w] = fm # 完成对fm的zeros
padding
52         rs = np.zeros([rs_h, rs_w], np.float32)
53
54         for i in range(rs_h):
55             for j in range(rs_w):
56                 roi = padding_fm[i*self.stride:(i*self.stride + k),
j*self.stride:(j*self.stride + k)]
57                 rs[i, j] = np.sum(roi * kernel) # np.ndarray格式下的 * 是对应
元素相乘
58         return rs
59
60 if __name__ == '__main__':
61     input_data = [
62         [
63             [1, 0, 1, 2, 1],
64             [0, 2, 1, 0, 1],
65             [1, 1, 0, 2, 0],
66             [2, 2, 1, 1, 0],
67             [2, 0, 1, 2, 0],
68         ],

```

```

69         [
70             [2, 0, 2, 1, 1],
71             [0, 1, 0, 0, 2],
72             [1, 0, 0, 2, 1],
73             [1, 1, 2, 1, 0],
74             [1, 0, 1, 1, 1],
75         ],
76     ],
77 ]
78 weight_data = [
79     [
80         [1, 0, 1],
81         [-1, 1, 0],
82         [0, -1, 0],
83     ],
84     [
85         [-1, 0, 1],
86         [0, 0, 1],
87         [1, 1, 1],
88     ]
89 ]
90 conv = my_conv(input_data, weight_data, 1, 'SAME')
91 print(conv.my_conv2d())

```

二、矩阵乘法版本实现

```

1  #!/usr/bin/env python3      #加上这一句之后，在终端命令行模式下就可以直接输入这个文件的
    名字后运行文件中的代码
2  # -*- coding = utf-8 -*-
3  import numpy as np
4
5  # 为了简化运算，默认batch_size = 1
6  class my_conv(object):
7      def __init__(self, input_data, weight_data, stride, padding = 'SAME'):
8          self.input = np.asarray(input_data, np.float32)
9          self.weights = np.asarray(weight_data, np.float32)
10         self.stride = stride
11         self.padding = padding
12     def my_conv2d(self):
13         """
14         self.input: c * h * w # 输入的数据格式
15         self.weights: c * h * w
16         """
17         [c, h, w] = self.input.shape
18         [kc, k, _] = self.weights.shape # 这里默认卷积核的长宽相等
19         assert c == kc # 如果输入的channel与卷积核的channel不一致即报错
20         # rs_h与rs_w为最后输出的feature map的高与宽
21         if self.padding == 'SAME':
22             pad_h = (self.stride * (h - 1) + k - h) // 2
23             pad_w = (self.stride * (w - 1) + k - w) // 2
24             rs_h = h
25             rs_w = w
26         elif self.padding == 'VALID':
27             pad_h = 0
28             pad_w = 0
29             rs_h = (h - k) // self.stride + 1
30             rs_w = (w - k) // self.stride + 1

```

```

31         elif self.padding == 'FULL':
32             pad_h = k - 1
33             pad_w = k - 1
34             rs_h = (h + 2 * pad_h - k) // self.stride + 1
35             rs_w = (w + 2 * pad_w - k) // self.stride + 1
36             # 对输入进行zeros padding, 注意padding后依然是三维的
37             pad_fm = np.zeros([c, h+2*pad_h, w+2*pad_w], np.float32)
38             for i in range(c):
39                 pad_fm[i, pad_h:pad_h+h, pad_w:pad_w+w] = self.input[i]
40             # 将输入和卷积核转化为矩阵相乘的规格
41             mat_fm = np.zeros([rs_h*rs_w, kc*k*k], np.float32)
42             mat_kernel = self.weights
43             mat_kernel.shape = (kc*k*k, 1) # 转化为列向量
44             row = 0
45             for i in range(rs_h):
46                 for j in range(rs_w):
47                     roi = pad_fm[:, i*self.stride:(i*self.stride+k),
j*self.stride:(j*self.stride+k)]
48                     mat_fm[row] = roi.flatten() # 将roi扁平化, 即变为行向量
49                     row += 1
50             # 卷积的矩阵乘法实现
51             rs = np.dot(mat_fm, mat_kernel).reshape(rs_h, rs_w)
52             return rs
53
54 if __name__ == '__main__':
55     input_data = [
56         [
57             [1, 0, 1, 2, 1],
58             [0, 2, 1, 0, 1],
59             [1, 1, 0, 2, 0],
60             [2, 2, 1, 1, 0],
61             [2, 0, 1, 2, 0],
62         ],
63         [
64             [2, 0, 2, 1, 1],
65             [0, 1, 0, 0, 2],
66             [1, 0, 0, 2, 1],
67             [1, 1, 2, 1, 0],
68             [1, 0, 1, 1, 1],
69         ],
70     ],
71     ]
72     weight_data = [
73         [
74             [1, 0, 1],
75             [-1, 1, 0],
76             [0, -1, 0],
77         ],
78         [
79             [-1, 0, 1],
80             [0, 0, 1],
81             [1, 1, 1],
82         ],
83     ]
84     conv = my_conv(input_data, weight_data, 1, 'SAME')
85     print(conv.my_conv2d())

```

参考资料

[1、im2col：将卷积运算转为矩阵相乘](#)

[2、面试基础--深度学习 卷积及其代码实现](#)

By Yee

2020.05.10