卷积操作的原理

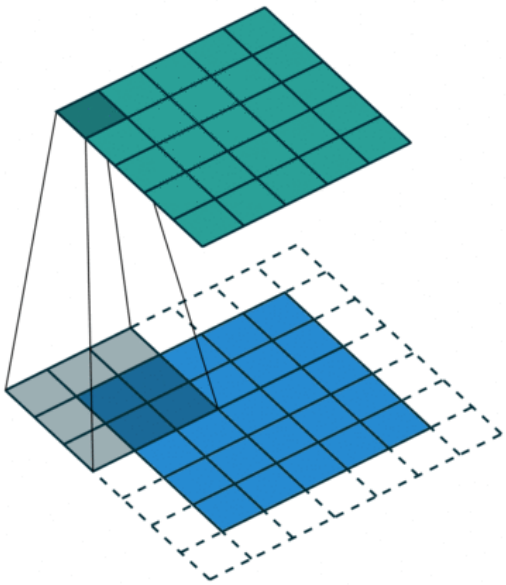
前言

同学们，大家好！
今天用一整节的篇幅，来给大家讲明白一个非常重要的知识点：卷积操作的原理。
在这篇短文中，我们按以下顺序展开讲述：

- 首先讲解卷积层在神经网络中的运算过程，从输入、输出、卷积核的尺寸和通道讲起；
- 其次验证一下可训练参数，手算卷积过程中可训练参数的数量，然后对比api计算结果；
- 最后通过一张示例图，帮大家理解感受野的概念。

单个卷积的运算

要想了解卷积层在神经网络中的计算过程，我们首先需要了解单个“卷积”是如何运算的。
想必大家在学习CNN的过程中都见过下图：



input_shape=(5,5), kernel_size=(3,3), padding='same', stride=1, output_shape=(5,5)
在这个图中：

- 蓝色部分代表原图（或“前一层特征图”），数值为原图的像素值；
- 绿色部分代表卷积所得到的结果（或“后一层特征图”），数值为计算所得像素值；
- 白色虚线部分代表扩充的padding，数值为0（此时padding='same'，向外扩充1格以保证前后特征图大小一致）；
- 在蓝白图上不停扫动的深色矩形代表卷积核（数值由随机初始化而来）。

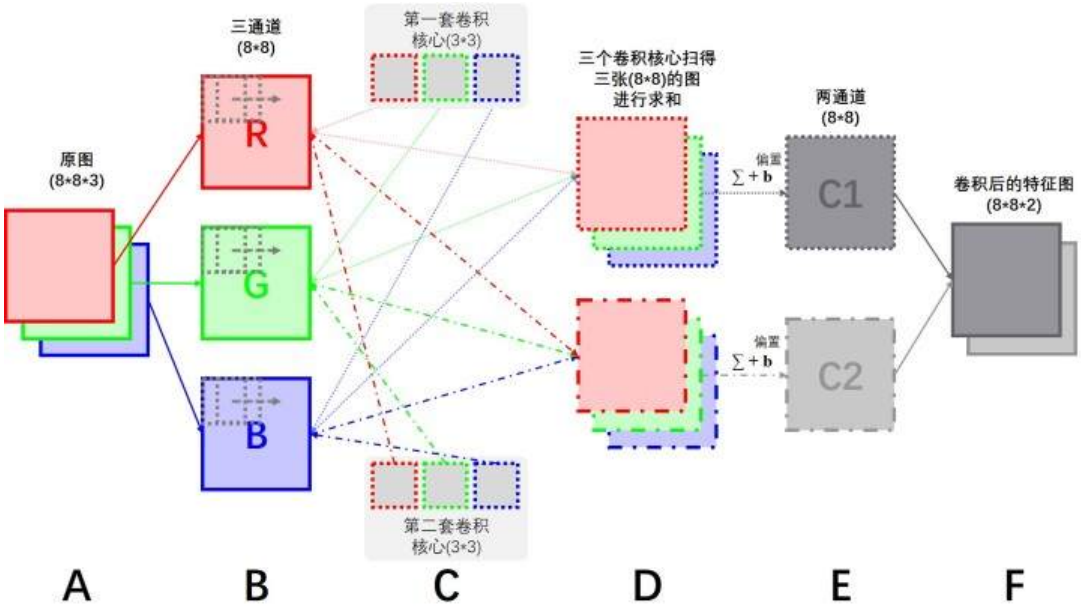
主要运算过程：

- 当卷积核扫到底图（蓝色部分+白色虚线部分）的左上角时，卷积核中的9个数字依次与底图上的9个数字相乘，而后对这乘得的9个数字进行求和，这样我们就得到了顶图（绿色部分）的第一个值。
- 以此类推，我们每进行一次上述运算，都把卷积核的位置向右移动一位，等卷积核向右移动到头时，就把卷积核向左归位并下移一行，继续进行同样的运算。
- 待上图的长、宽都与底图相等时，这一次卷积也就计算结束。

- 在实际应用中，每一个输出的特征图还会配备一个偏置项bais，上图中没有表示。

多通道卷积的运算

了解完单个卷积是如何计算的之后，我们就可以从神经网络的角度来看‘卷积层’的运算过程。下图展示的是图像（8*8*3）经一层卷积结构，输出特征图（8*8*2）的计算过程：



input_shape=(8,8,3), kernel_size=(3,3), padding='same', stride=1, output_shape=(8,8,2)

在这个图中：

- A：原图（3通道，8*8）
- B：展平了方便看的原图
- C：不同的卷积核（以颜色和虚线类型区分）
- D：不同卷积核在底图上扫过所得到的单个计算结果图（以颜色和虚线类型区分）
- E：展平了方便看的输出图
- F：输出图（2通道，8*8）

主要运算过程：

- 首先我们来关注一下输入和输出，它们的尺寸都是8*8，而输入是3通道，输出是2通道。在分析卷积结构时一定要先看输入输出，对一层是这样，对整个模型也是这样。
- 其次就准备进入我们最熟悉的卷积核计算，可是在此之前我们得知道，这个运算过程中到底发生了几次卷积核计算呢？有人可能要说，有几通道的输出就有几个卷积核，每个卷积核把输入特征图从头扫到尾。然而这种说法是不够准确的！
- 实际上，在卷积核计算数量问题上，应该是“**有几通道的输出就有几套卷积核，每套内的卷积核数量与输入通道数相等**”，就像我在上图中所画的：
  - 由C中的上下两套，每套有三个卷积核去扫输入的3张图（颜色一一对应）；
  - 得到D中的两套，每套3张计算结果图；
  - 在经求和及加入偏置量，得到要输出的2通道结果。

至此，这一个卷积层的运算就全部完成了。

验证“可训练参数”

毕竟空口无凭，下面通过“**可训练参数**”的数量，来验证一下卷积层是否按照上述过程运算。大家应该知道，一个卷积层内的“可训练参数”，其实就是指的卷积核里的那些值，以及要加的偏置项，那么如果按照前面描述的计算方法来看，一个卷积层内的”可训练参数”有多少呢？我们可知：

- 我们的卷积核边长为3（简写为 k_w 及 k_h ）
- 输入3通道（input_channel，简写为 C_{in} ）

- 输出2通道（output_channel，简写为 C_{out} ）

由此可得到：

- 单个卷积核内的参数量为： $k_w * k_h = 3 * 3 = 9$
- 卷积核数量为： $C_{in} * C_{out} = 3 * 2 = 6$
- 偏置数量为： $C_{out} = 2$

那么按理说可训练参数量应为：

$$\begin{aligned} params &= 单个卷积核内的参数量 * 卷积核心数量 + 偏置数量 \\ &= k_w * k_h * C_{in} * C_{out} + C_{out} \\ &= 3 * 3 * 3 * 2 + 2 \\ &= 56 \end{aligned}$$

让我们用keras的summary()来验证一下：

```
In [2]: from keras.models import Model, Sequential
        from keras.layers import Convolution2D

        input_shape = (8, 8, 3)
        kernel_size = (3, 3)
        padding = 'same'
        output_channel = 2

        model = Sequential()
        model.add(Convolution2D(filters=output_channel, kernel_size=kernel_size, padding=padding, input_shape=input_shape))
        model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 8, 8, 2)           56
=================================================================
Total params: 56
Trainable params: 56
Non-trainable params: 0
_________________________________________________________________
```

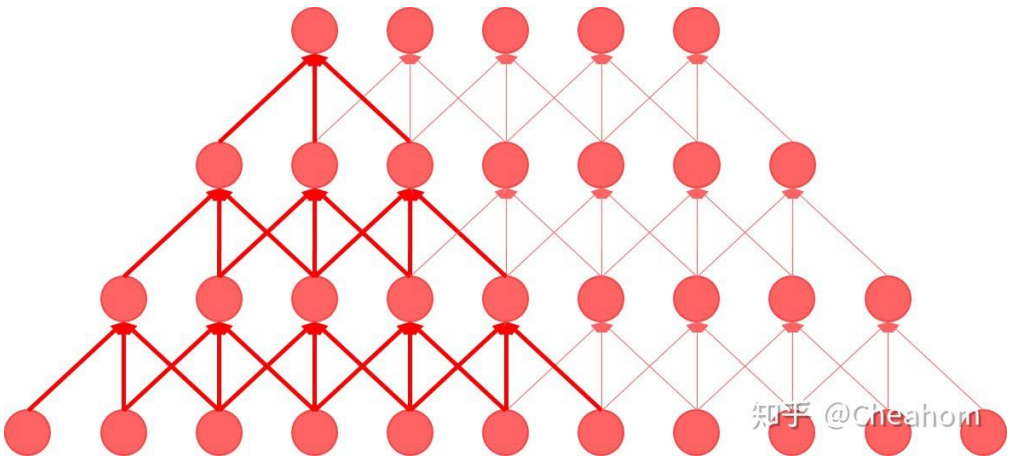
api的结果正好等于手算结果。

请记住这个公式，普通卷积层可训练参数的数量为：

$$\begin{aligned} params &= 单个卷积核内的参数量 * 卷积核心数量 + 偏置数量 \\ &= k_w * k_h * C_{in} * C_{out} + C_{out} \end{aligned}$$

感受野

这里为大家明确“感受野”的概念，简单来讲就是卷积神经网络中的某一层特征图上的一个点，对应到原图上可以关联到多少个点，我们用一张图来解释一下：



上图展示的是一个3层一维卷积，kernel_size=3，我们可以看到：顶层左一的像素与底层左起7个像素值有关，这时候就代表它的感受野有7。我们可以显而易见的得出以下两个结论：

- kernel_size不变的情况下，层数越深，感受野越大；
- 层数不变的情况下，kernel_size越大，感受野越大。