人工智能 机器学习 神经网络 深度学习（Deep Learning） 卷积神经网络（CNN）

能否对卷积神经网络工作原理做一个直观的解释?

我目前大四，在用CNN做手写识别毕业设计，已经接触机器学习4个月了。但CNN是目前最让我困惑的，其简直就像黑匣子，只有输入输出，然后看看准确率，完全不...显示全部

关注问题 写回答 邀请回答 好问题 14 10 条评论 分享 ···

查看全部 67 个回答

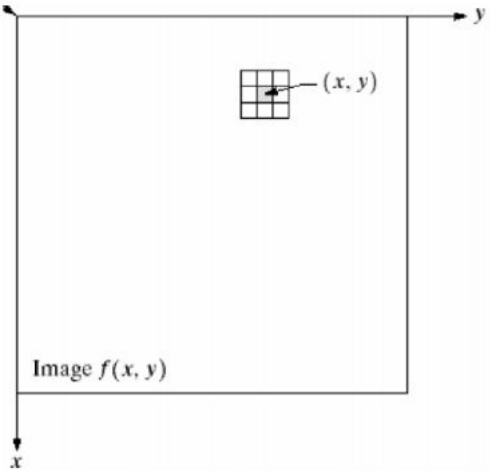
OwlLite
神经科学话题下的优秀回答者

编辑推荐

2,806 人赞同了该回答

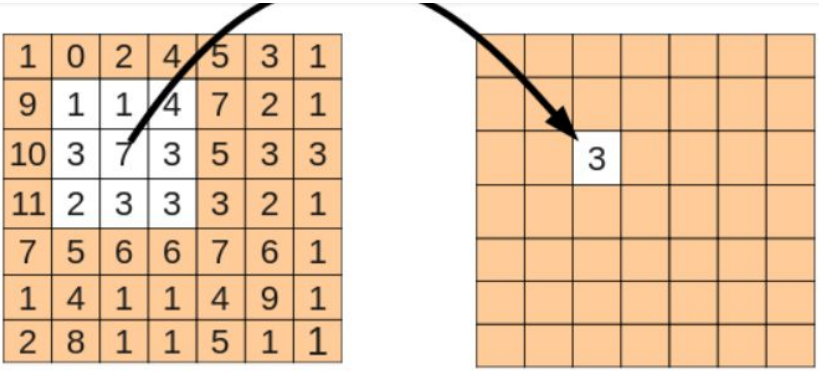
从最基础的开始

对二维数字信号（图像）的操作，可以写成矩阵形式。



比如对图像做平滑，一个典型的8领域平滑，其结果中的每个值都来源于原对应位置和其周边8个元素与一个3X3矩阵的乘积：

▲ 赞同 2806 ▼ 109 条评论



也就相当于对原矩阵，按照顺序将各区域元素与W矩阵相乘，W 矩阵为

$$W = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

这也被称作核(Kernel, 3X3)

其处理效果如下：

Figure: Input image; Mean image with 3×3 kernel and Mean filter with 5×5 kernel.

也就是，这个核对图像进行操作，相当于对图像进行了低通滤波。因此这个核也被称为滤波器，整个操作过程按照概念称为卷积。

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

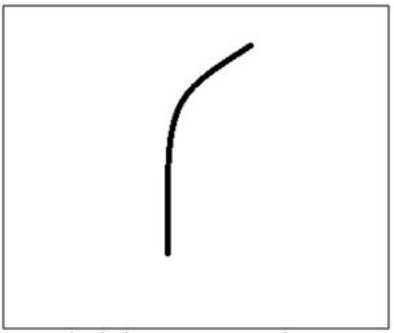
$$\frac{1}{273}\begin{bmatrix}1&4&7&4&1\\4&16&26&16&4\\7&26&41&26&7\\4&16&26&16&4\\1&4&7&4&1\end{bmatrix}$$

滤波器跟卷积神经网络有什么关系呢。不如我们预想一个识别问题：我们要识别图像中的某种特定曲线，也就是说，这个滤波器要对这种曲线有很高的输出，对其他形状则输出很低，这也就像是神经元的**激活**。

我们设计的滤波器和想要识别的曲线如下：

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

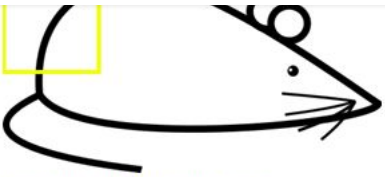
Pixel representation of filter



Visualization of a curve detector filter

假设上面的核（滤波器）按照卷积顺序沿着下图移动：

Original image



Visualization of the filter on the image

那么当它移动到上面的位置时，按照矩阵操作，将这个区域的图像像素值与滤波器相乘，我们得到一个很大的值（6600）：

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

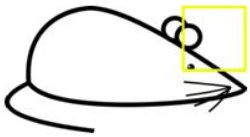
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

而当这个滤波器移动到其他区域时，我们得到一个相对很小的值：



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

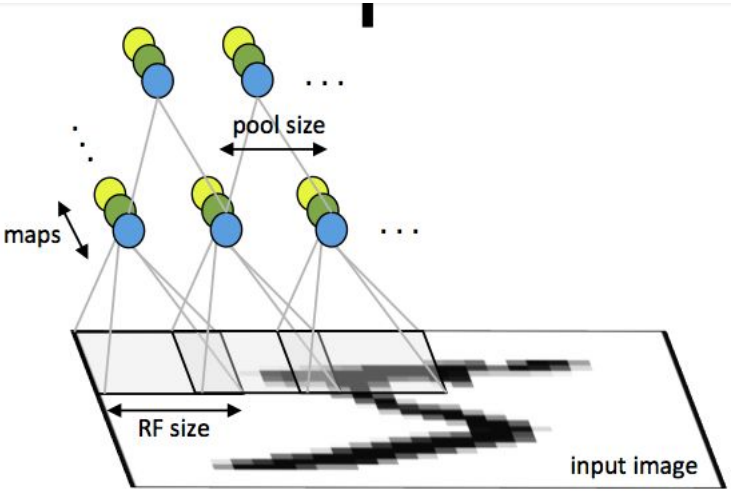
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

如此，我们对整个原图进行一次卷积，得到的结果中，在那个特定曲线和周边区域，值就很高，在其他区域，值相对低。这就是一张**激活图**。对应的高值区域就是我们所要检测曲线的位置。

在训练卷积审计网络（CNN）的某一个卷积层时，我们实际上是在训练一系列的滤波器(filter)。比如，对于一个32x32x3（宽32像素x高32像素xRGB三通道）的图像，如果我们在CNN的第一个卷积层定义训练12个滤波器，那就这一层的输出便是32X32X12.按照不同的任务，我们可以对这个输出做进一步的处理，这包括激活函数，池化，全连接等。



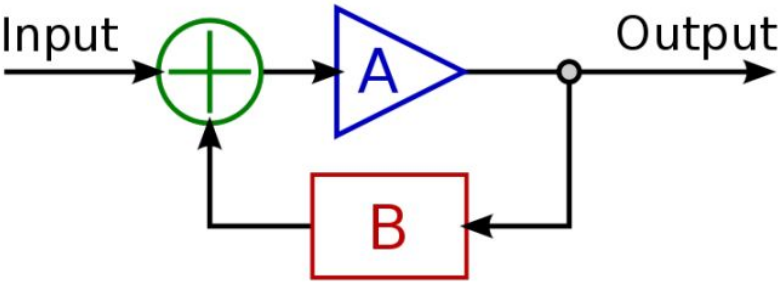
简单来说，训练CNN在相当意义上是在训练每一个卷积层的滤波器。让这些滤波器组对特定的模式有高的激活，以达到CNN网络的分类/检测等目的。

卷积神经网络的第一个卷积层的滤波器用来检测低阶特征，比如边、角、曲线等。随着卷积层的增加，对应滤波器检测的特征就更加复杂（理性情况下，也是我们想要的情况）。比如第二个卷积层的输入实际上是第一层的输出（滤波器激活图），这一层的滤波器便是用来检测低价特征的组合等情况（半圆、四边形等），如此累积，以检测越来越复杂的特征。实际上，我们的人类大脑的视觉信息处理也遵循这样的低阶特征到高阶特征的模式（Owl of Minerva：为什么无彩色系（黑白灰色）在色彩搭配中可以和谐地与任何彩色搭配？）。最后一层的滤波器按照训练CNN目的的不同，可能是在检测到人脸、手写字体等时候激活[1]。

所以，在相当程度上，构建卷积神经网络的任务就在于构建这些滤波器。也就是，将这些滤波器变成这样(改变滤波器矩阵的值，也就是Weight)的——能识别特定的特征。这个过程叫做**训练。**

在训练开始之时，卷积层的滤波器是完全随机的，它们不会对任何特征激活（不能检测任何特征）。这就像刚出生的孩子，TA不知道什么是人脸、什么是狗，什么是上下左右。TA需要学习才知道这些概念，也就是通过接触人脸、狗、上下左右，并被告知这些东西分别是人脸、狗、上下左右。然后TA才能在头脑中记住这些概念，并在之后的某一次见到之后能准确的给出结果。

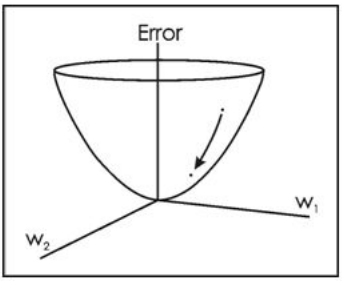
把一个空白的滤波其，修改其权重(weights)以使它能检测特定的模式，整个过程就如工程里面的反馈。



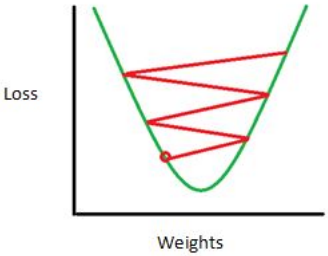
想想一下，如果有一只无意识的猴子，完全随机的修改一个5X5滤波器矩阵的25个值，那完全可能经过一定的轮次之后，这个滤波器能够检测棱角等特征。这是一种无反馈的训练情况。对神经网络的训练当然不能如此，我们不可能靠运气去做这件事情。

举个例子，我们要训练一个用于分类的神经网络，让它能判定输入图像中的物体最可能是十个类别的哪一类。那么，训练过程就是这样的：

第一次训练，输入一张图像，这个图像通过各层卷积处理输出量一组向量[1,1,1,1,1,1,1,1,1,1], 也就是，对于完全由随机滤波器构建的网络，其输出认为这张图等概率的是十个类别中的某一种。但是对于训练，我们有一个Gound Thuth, 也就是这张图中物体所属的类别：[0，0，1，0，0，0，0，0，0，0]，也就是属于第三类。这时候我们可以定义一个损失函数（squared error）.我们假定L是这个损失函数的输出。这时候我们



这是一个典型的最优化问题。当然地，在工程上我们几乎不可能一次就把滤波器的权重W修改到使L最小的情况，而是需要多次训练和多次修改。



如果情况理想的话，权重修改的方向是使得L的变化收敛的。这也就是说很可能达到了我们训练这个神经网络的目的——让各个卷积层的滤波器能够组合起来最优化的检测特定的模式。

---

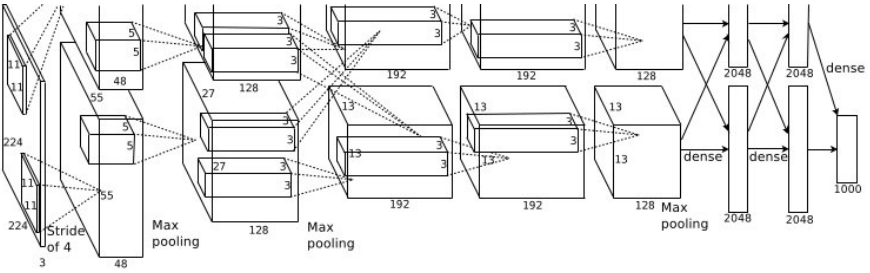
[1] Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

编辑于 2017-09-02

更多回答

慕云
数学 通信 计算机

1,365 人赞同了该回答



先放一张比较经典的图片初步了解一下卷积网络的结构，由于本文主要探讨卷积的作用，为了防止被自己带离节奏，这里不对该图做进一步解释，所以有感兴趣的可以参考**AlexNet**的讲解。但是如果你理解了此文，那么我觉得AlexNet你就能自行尝试理解了。

正文

卷积做了些什么：

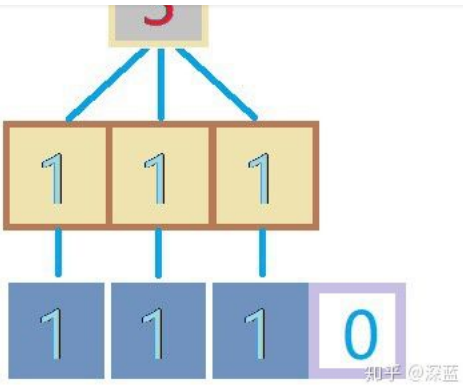
卷积这个名词，总容易使人感到不明觉厉，学数字信号处理的时候，就曾被这个名字搞得有点蒙圈，然而后来发现抛开名字，卷积做的事情并不复杂。

【注：深度学习中的卷积与信号处理中的卷积概念是有区别的，这里只介绍一下卷积神经网络中的卷积；所以以下所有卷积均指前者】

卷积操作可以被看做对输入的一种处理，大多数人正是因为把问题看得太复杂了，所以才感到有种被支配的感觉。其实卷积的操作就是加法和乘法的组合，与平时经常遇到的函数运算的区别是卷积操作具有**时间**或**序列**概念，是对数据的一个连续处理的过程，可以先通过一个一维结构数据的卷积来帮助理解。

假如有一段数据**[1,1,1,0]**

我们现在要对其做卷积操作

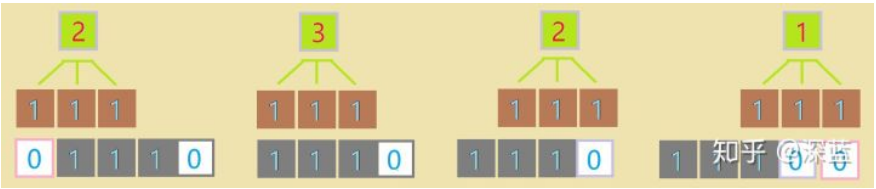


输入为**[1;1;1;0]**

中间一层我们称之为**卷积核**

而卷积操作过程其实很简单，也就是将卷积核与数据对应相乘，然后求和。所以对于上图，其操作过程便是1*1+1*1+1*1=3，“3”便是得到的结果或输出。但这只并不是卷积的整个过程，前面提到过，卷积操作是具有时间或序列属性的，而这只是整个卷积过程的其中一步，卷积复杂的地方也就在此，其实想想也没什么，剩下的都是些重复性操作，而整个卷积过程的输出便是这每一小小步产生结果的组合了。

下面便是一个完整的卷积过程，为了让输入数据长度与输出数据相同；我们可以进行补零操作，也就是对边缘没有数据的地方按零处理（与卷积网络的Padding作用类似）。



于是对于数据【**1,1,1,0**】在进行对边缘补一个零的操作后进行卷积的输出结果为

【**2；3；2；1**】（长度与输入相同）

而卷积的精髓其实是卷积核，换一个卷积核，卷积的输出结果便会迥然不同。

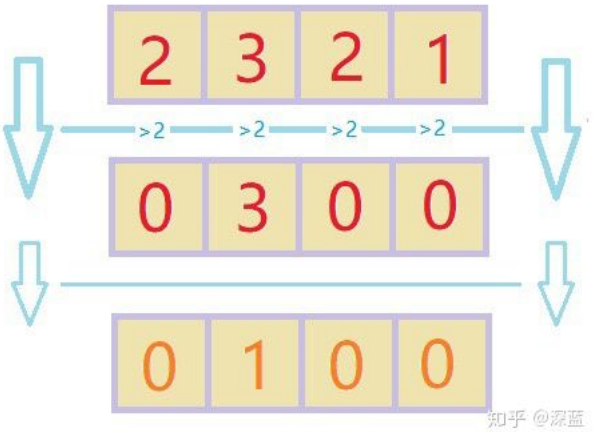
我们再来观察一下：

对数据【1,0,1,1,0】进行卷积处理的输出分别为：【-1,2,0,0,1】与【1,0,0,2,1】

由此可以发现，该例中卷积核的作用就是找到与它具有相同结构的数据，**相似度越大，其对应输出也就相对越大**（但这并不能作为一个结论去在实际中应用，因为实际卷积过程其实很复杂，而且是要涉及到反向传播的。比方说当卷积核有值取“0.5”时这句话就会失效，0.5*0.5的结果显然没有0.5*1的大，不过这也许能为卷积网络的设计提供一些想法上的参考。所以为了便于理解，本例卷积核仅取“+1”，“-1”，要处理的信息取值范围为[0,1]，而此处仅取“0”或“1”）；

既然这个最大值是如此重要，我们完全可以忽略其余部分，把最重要的信息提取出来（这个过程有点类似于Sigmoid操作）；

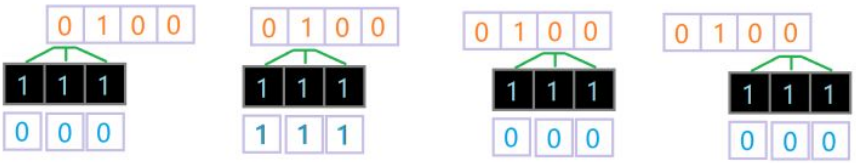
下图为提取过程：

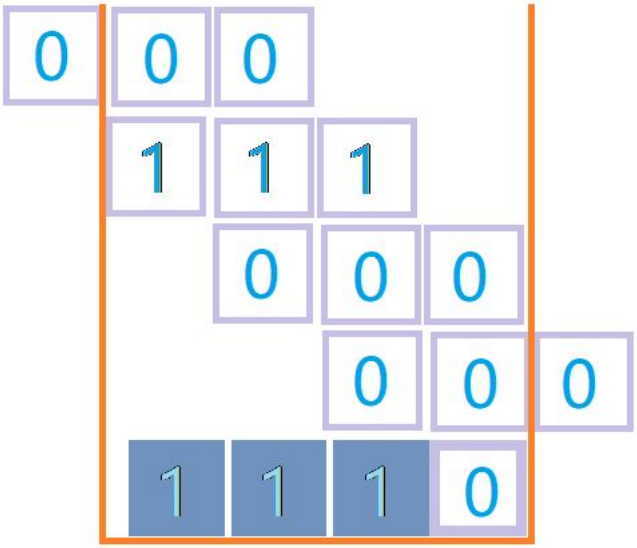


图中最底层对应为“1”的地方我们便可以理解为在它的上一层，包含有我们所检测的结构，这一信息；

也就是说，通过这一层中为“1”的位置可以反向找到与卷积核具有相同结构的数据。

过程见下图：

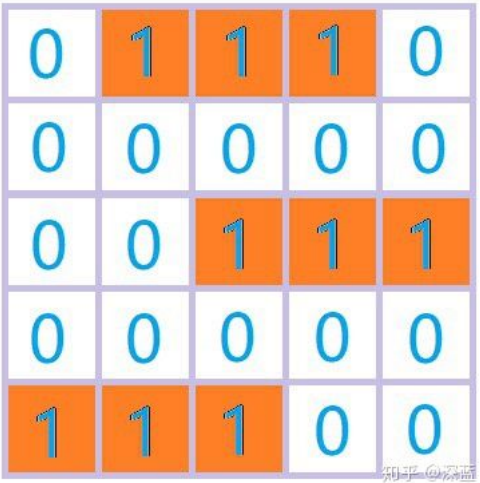




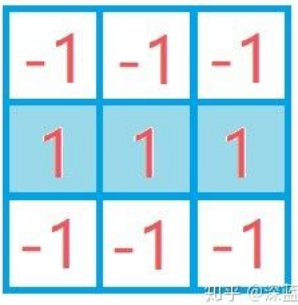
于是输入与输出就通过卷积操作对应了起来。

再举一个例子：

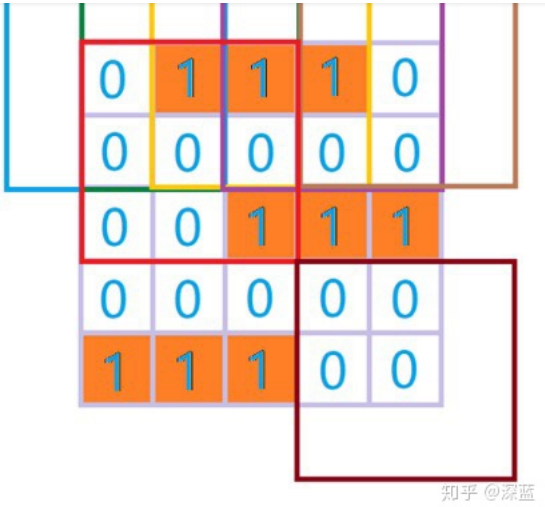
对二维结构的数据的卷积操作与一维数据的卷积类似，只不过卷



我们可以利用用下面这个卷积核；



对数据进行卷积操作；

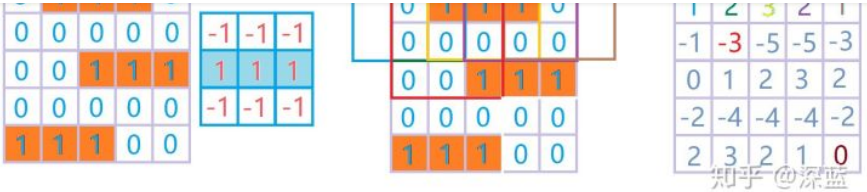


得到输出为:

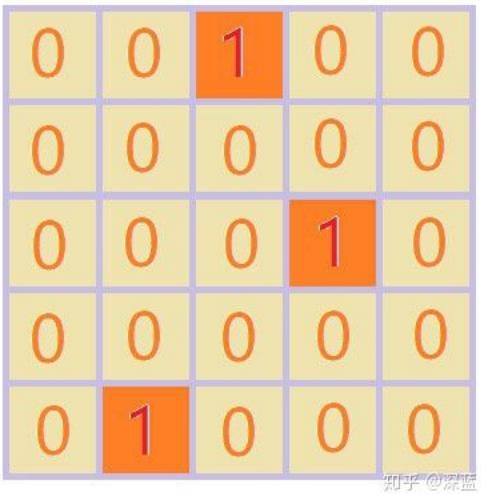
(注意颜色之间的对应)

1	2	3	2	1
-1	-3	-5	-5	-3
0	1	2	3	2
-2	-4	-4	-4	-2
2	3	2	1	0

或许全局看更好一些;

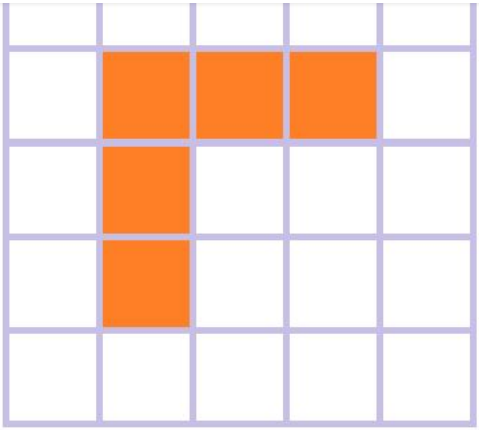


与一维数据的处理一样，我们同样可以提取出关键信息：



所以从图中可以看出，每一个“1”都与上层的**橘黄色【1,1,1】**所对应。

同样对于下面形状，



可以用下面卷积核来处理；

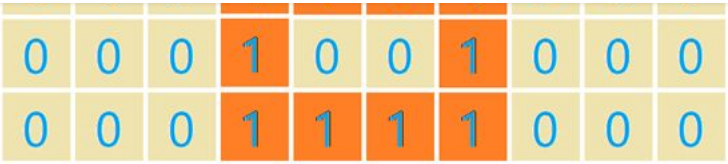
1	1	1
1	-1	-1
1	-1	-1

过程不再赘述。

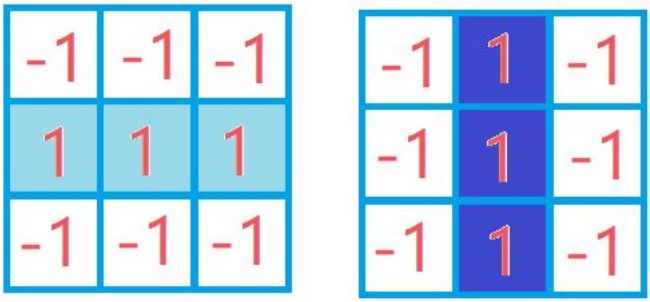
二次卷积操作过程：

下面通过一个例子理解一下二次卷积操作：

对下面数据

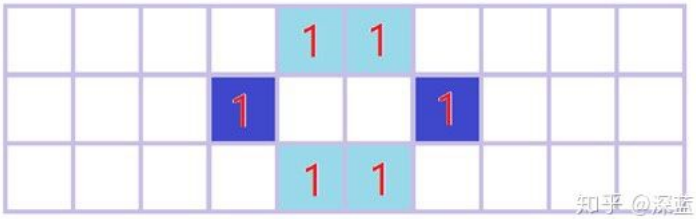


我们可以先用下面两个卷积核进行第一次处理；

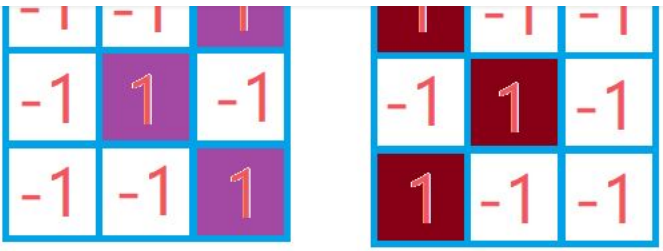


卷积输出经过关键信息提取后的结果为：

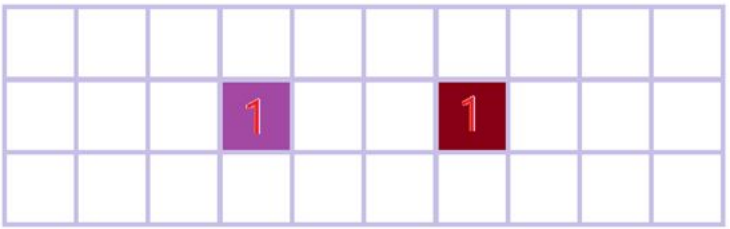
（注意颜色的相互对应）



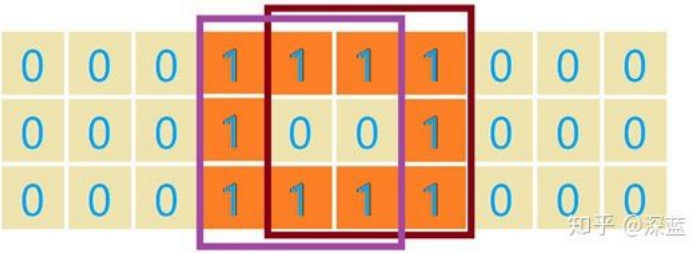
然后对上面的处理结果用下面卷积核进行第二次卷积操作；



处理后的结果为（注意颜色）：



而上图每一个涂颜色框都与上上层数据的一个结构相关联；下面用不同颜色标了出来；



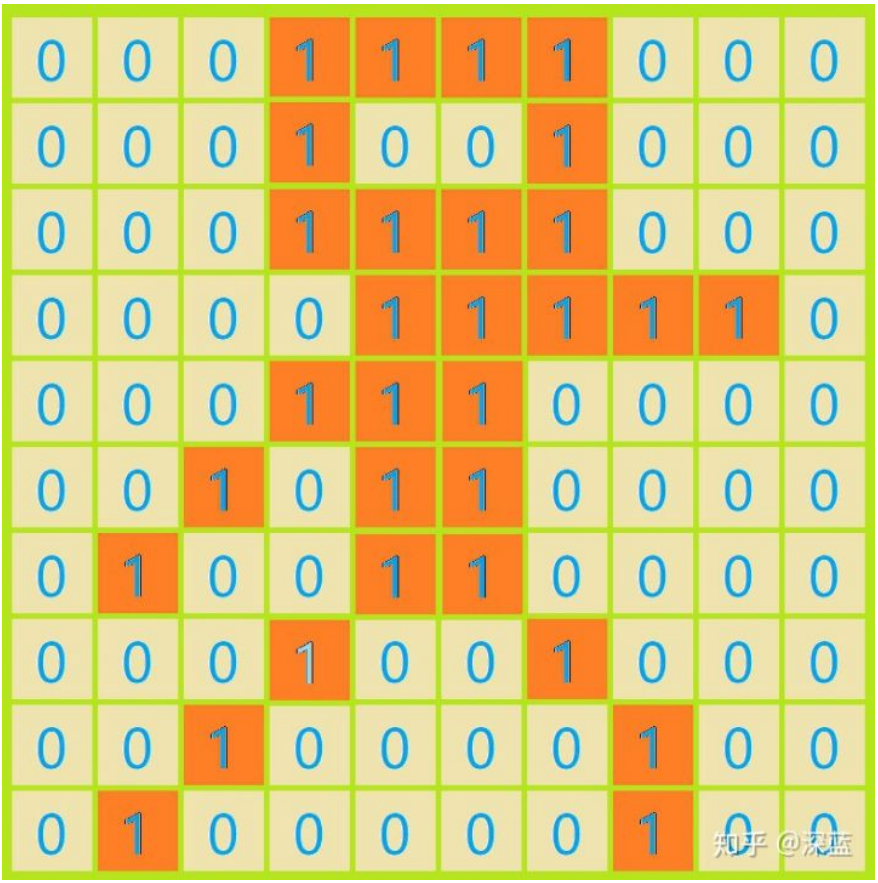
更深层次的探究：

有了这些理解我们就可以尝试一点更高级的玩意儿了，比方说识别一个“人”出来。

对你没听错，前提是我们的“人”长下面这样。

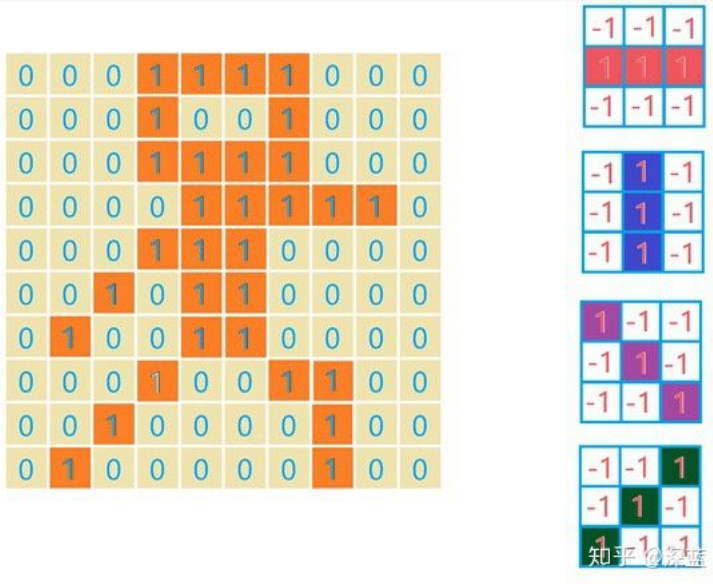
。

。

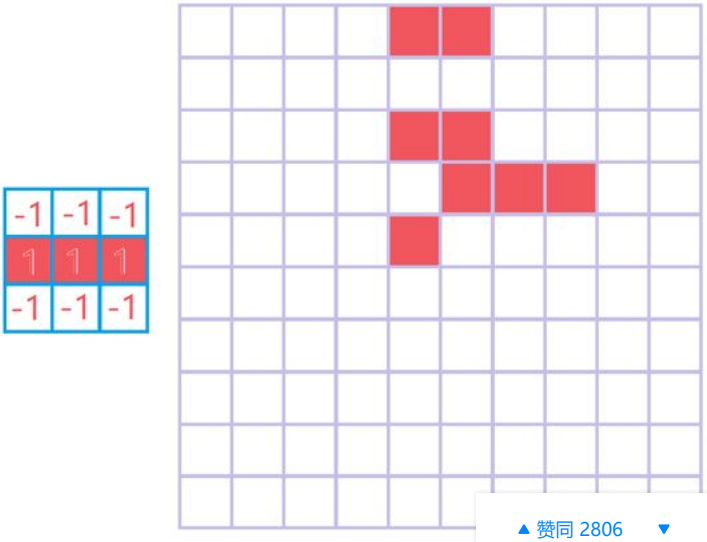


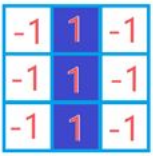
这里给出**简要过程**：

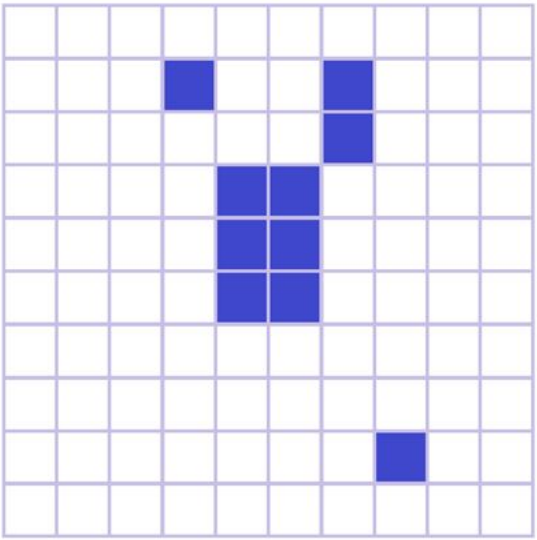
第一步：

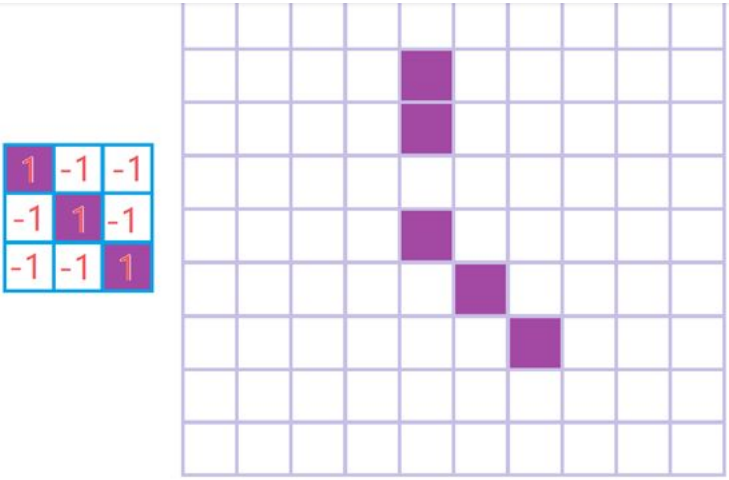


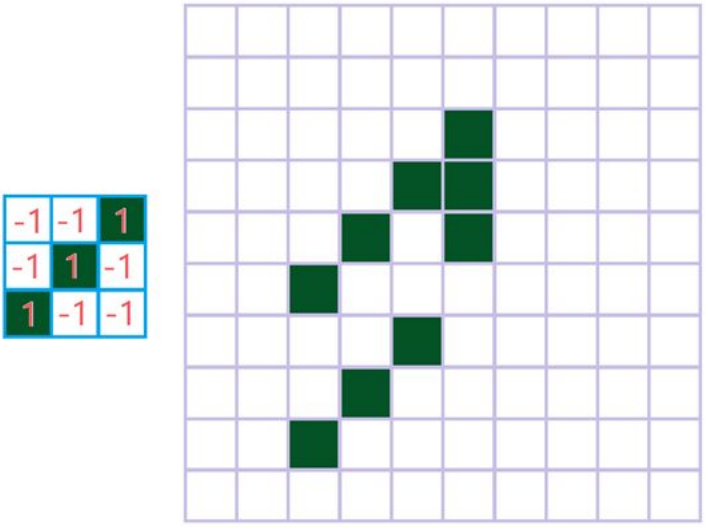
每一个卷积核对应的**最后**输出为：



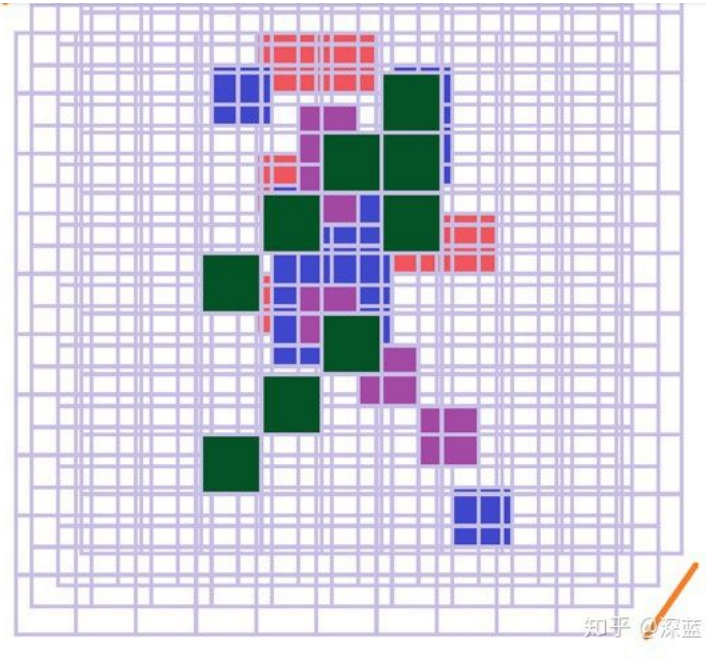

-1 1 -1
-1 1 -1
-1 1 -1





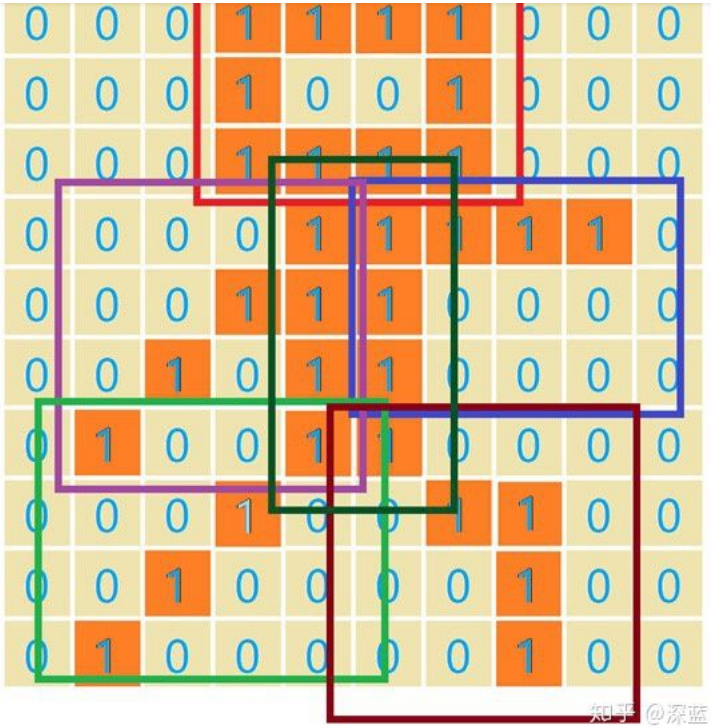


我们还可以将四层输出形成一个三维结构的数据（尝试思考一下三维卷积操作，是不是想出来了呢？对就是你想的那样）：

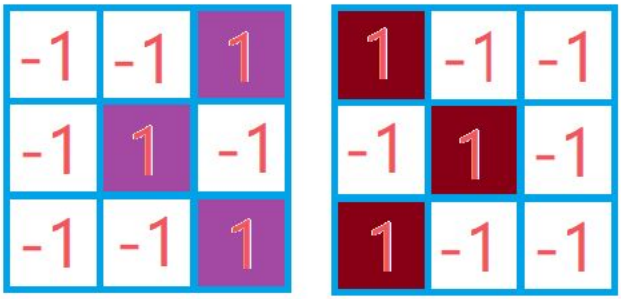


接下来我们就可以根据第一层卷积提取的结构，来提取更大的结构了；

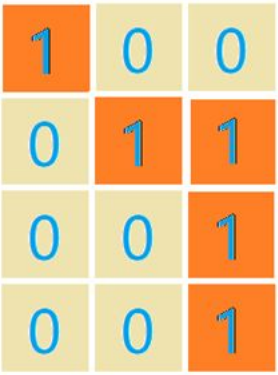
如：头；前胳膊，后胳膊，左腿，右腿；还有身子；见下图

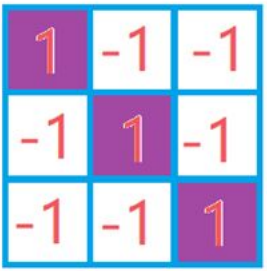


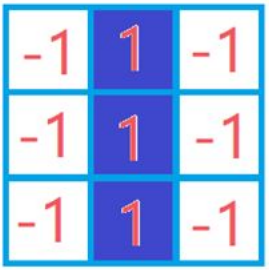
对于头：我们之前已经见识过了，当时利用的是下面两个卷积核对第一层输出进行的处理；



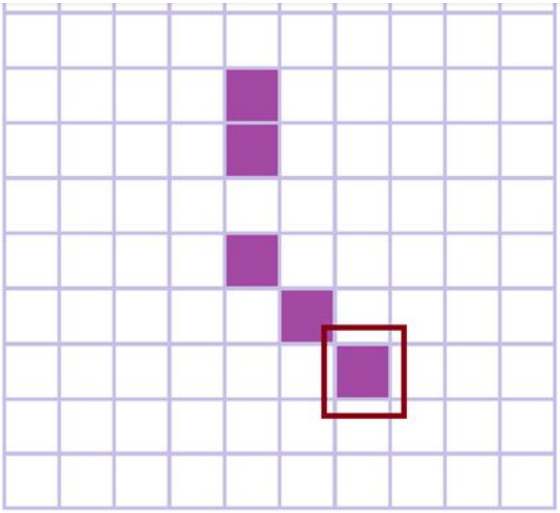
下**图左**为这个人的右腿，经过第一层的卷积，其主要由下**图右**两部分组成，所以是否包含腿的信息可以经由下**图右**两部分结构的提取获得；

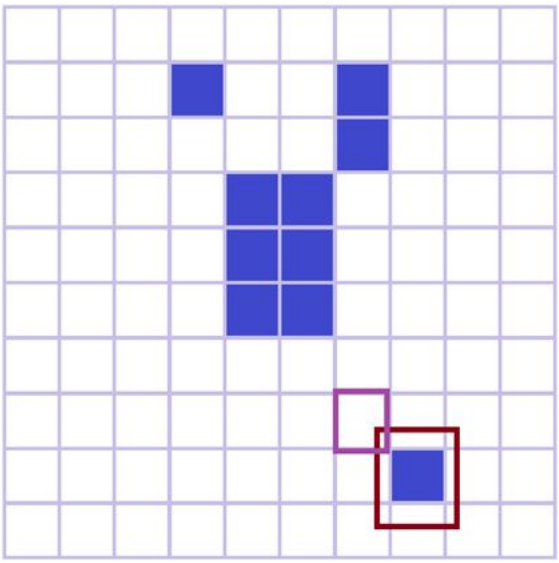




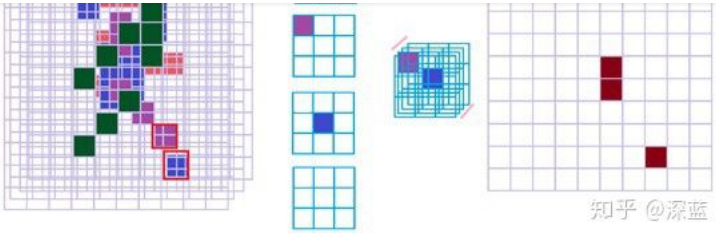


可以看出其实我们主要是从第一层卷积中的这两层（下图）提取出来蕴含“前腿”的信息的；

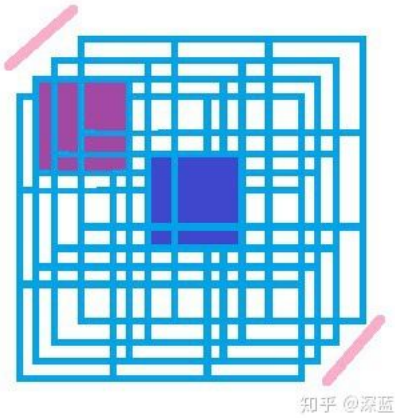




具体过程如下：

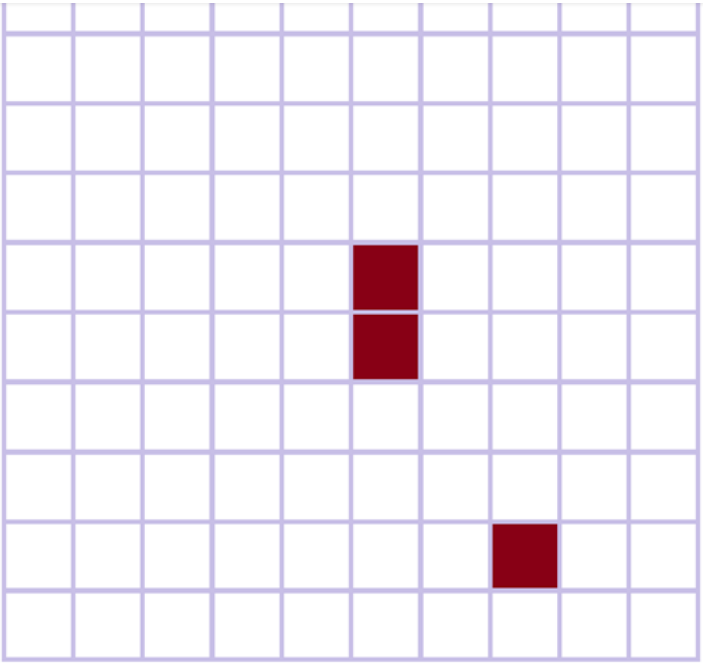


而卷积核的样子是这样的；

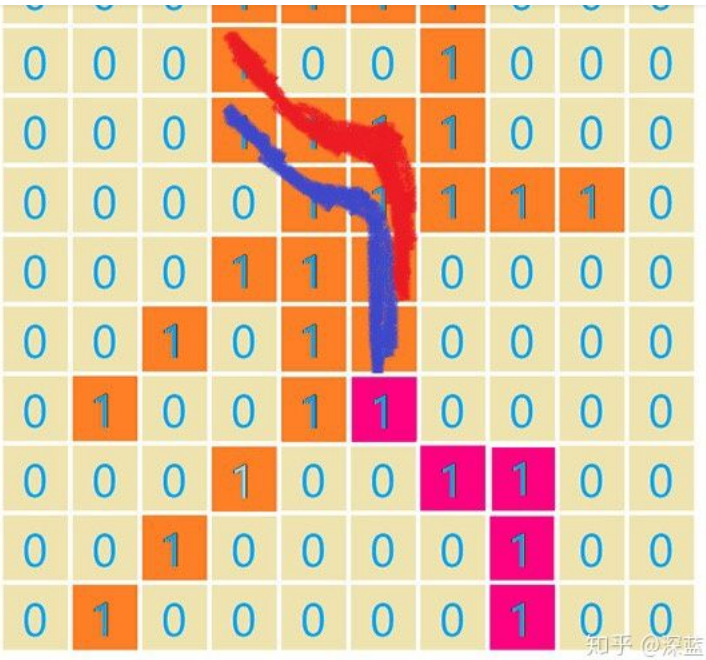


它是一个三维结构，共四层，每层都是3*3的结构；**而它就是一个三维卷积操作的卷积核。**

输出结果为：

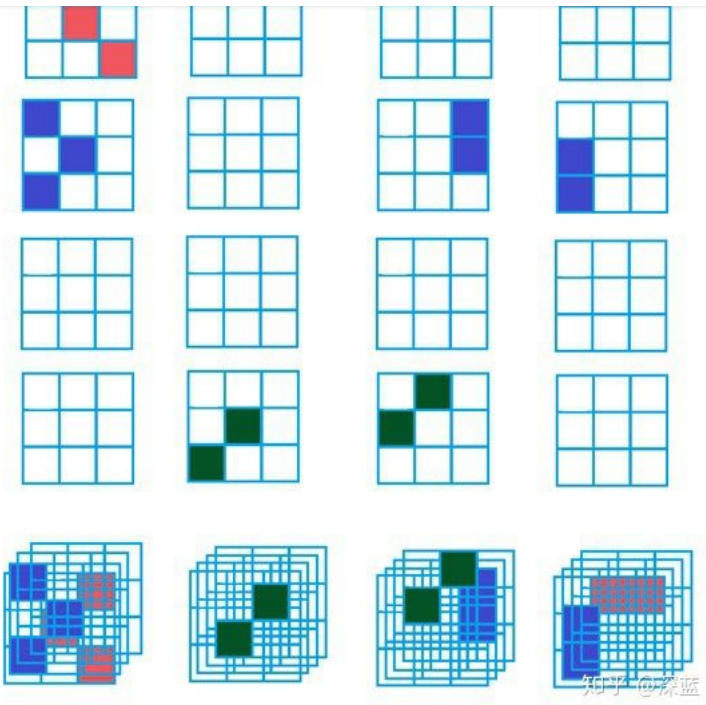


共有三条“腿”被检测了出来，从原始图中可以看出，只要是具有与我们要检测的“前腿”具有相同结构的地方都被我们检测了出来；见下图



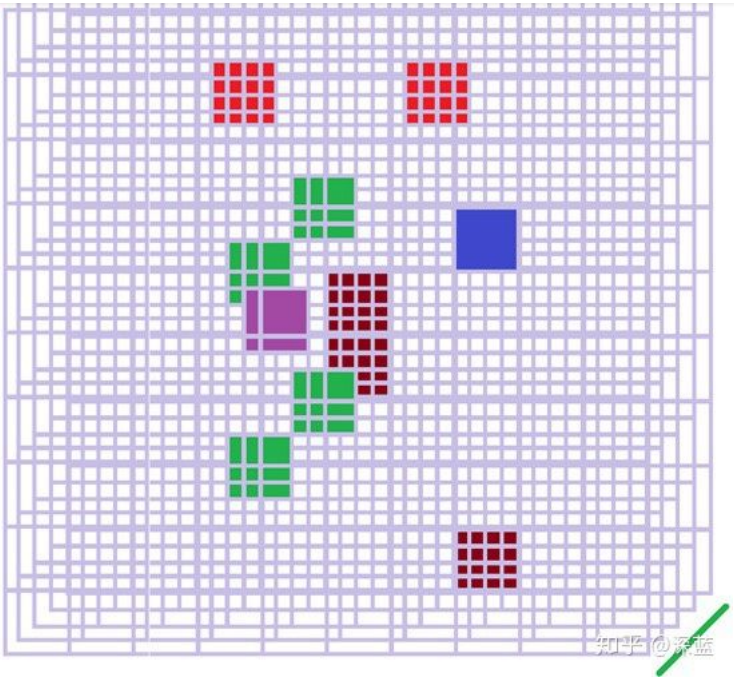
类似的:

分别取下面卷积核对**第一层输出**处理



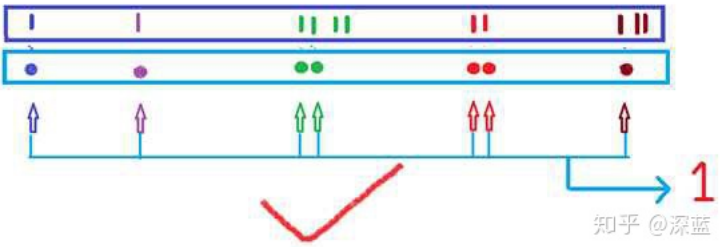
得到输出结果为：

顺便将它们三维化；



上图便是对“前腿”，“头”，“后腿”，“左臂”，“右臂”的检测结果（可以看出有些结构检测到不只一个；而对于头部，我们也是分为前后两部分分别检测的）。

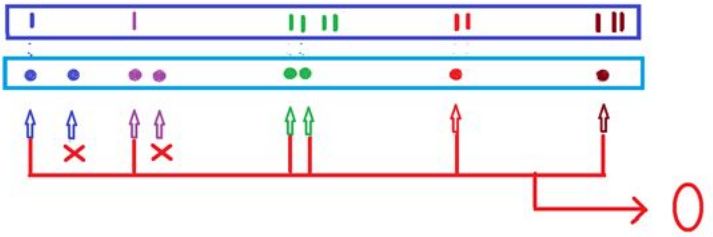
所以我们怎么利用上层输出，来预测要检测的“图片”中是否有人呢？这就需要全连接层了，这里不做过多讨论，不过对于本例子，我觉得全连接层可以理解为一把锁，而钥匙便是上层输出，当锁孔与钥匙匹配时，锁就能被打开。见下图：



要进行最后一层的操做，我们需要将上层输出展开，也就是将三维结构的数据展为一维，图上紫色方框便是展开后的一维数据，也就是我所比喻的“**锁**”，而青色

当然，如果不匹配的话，输出为“0”：

如图：



实际中的Fully Connect层要做的不止图上画的这些，权重也不能理解为简单的钥匙。就是说我们的这把“钥匙”不仅要考虑“凸”的地方，还要考虑“凹”的地方，因为全连接层还要对一些节点做出“惩罚”。打个比方就能明白，在本例子中，如果我们输入的图片对应的像素值全为“1”，相当于一张只有白色的图片，经过卷积也能提取很多信息，但最后结果肯定不应该输出为“1”，所以还要对一些节点做出“惩罚”，当“惩罚”太大时输出就不是“1”了。

总结：

写本文的主要原因是想记录一下我个人的一些想法，其中自娱自乐的性质大一些。而且很多自己觉得没必要记下的也没写，但是并不妨碍本文的理解，例如对于Pooling层本例并没有用到，所以就没有提及。

完

编辑于 2018-11-24

真诚赞赏，手留余香

赞赏

3 人已赞赏

机器之心
人工智能话题下的优秀回答者

1,195 人赞同了该回答

> 近日，Dishashree Gupta 在 Analyticsvidhya 上发表了一篇题为《Architecture of Convolutional Neural Networks (CNNs) demystified》的文章，对用于图像识别和分类的卷积神经网络架构作了深度揭秘；作者在文中还作了通盘演示，期望对 CNN 的工作机制有一个深入的剖析。机器之心对本文进行了编译，原文链接在此，希望对你有帮助。

机器视角：长文揭秘图像处理和卷积神经网络架构

引言

先坦白地说，有一段时间我无法真正理解深度学习。我查看相关研究论文和文章，感觉深度学习异常复杂。我尝试去理解神经网络及其变体，但依然感到困难。

展开阅读全文

接着有一天，我决定一步一步，从基础开始。我把技术操作的步骤分解开来，并手动执行这些步骤

▲ 赞同 1195 ▼ 55 条评论 分享 收藏 喜欢 ···

查看全部 67 个回答

▲ 赞同 2806 ▼ 109 条评论