

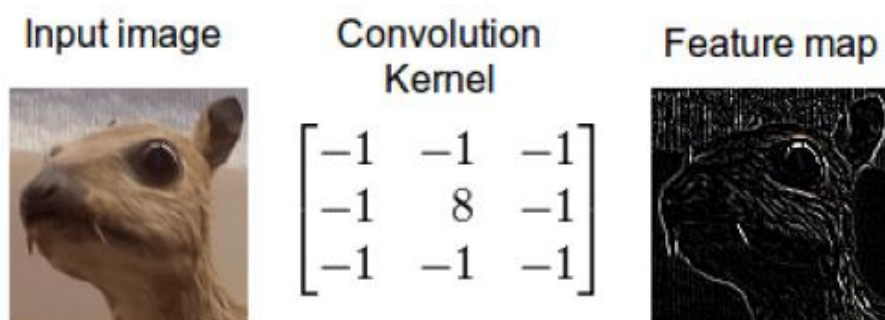
学习笔记 1-CNN-2017-07-19

从卷积、池化到 CNN 深入

什么是卷积？

你可以把卷积想象成一种混合信息的手段。想象一下装满信息的两个桶，我们把它们倒入一个桶中并且通过某种规则搅拌搅拌。也就是说卷积是一种混合两种信息的流程。

我们混合两桶信息：第一桶是**输入的图像**，由三个矩阵构成——RGB 三通道，其中每个元素都是 0 到 255 之间的一个整数。第二个桶是**卷积核 (kernel)**，单个浮点数矩阵。可以将卷积核的大小和模式想象成一个搅拌图像的方法。**卷积核的输出是一幅修改后的图像，在深度学习中经常被称作 feature map**。对每个颜色通道都有一个 feature map。



为什么要进行卷积？

图像中可能含有很多我们不关心的噪音。

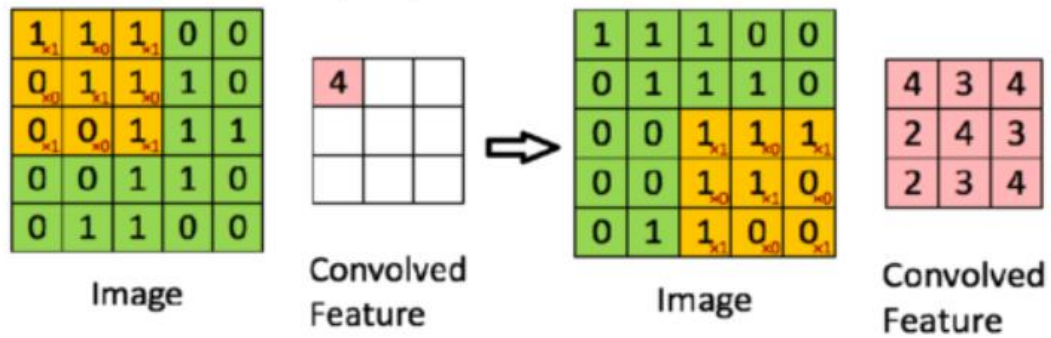
拿识别衣服样式的例子来说——如果你想要区分衣服的式样，那么衣服的颜色、商标之类的细节就不那么重要。最重要的可能是衣服的外形。如果我们过滤掉这些多余的噪音，那我们的算法就不会因颜色、商标之类的细节分心了。我们可以通过卷积轻松地实现这项处理。

通过卷积运算，可以使原信号特征增强，并且降低噪音。所以，卷积应用经常被称作滤波，而卷积核经常被称作滤波器。通过卷积，把不感兴趣的过滤掉，得到我们只关心的那部分信息。

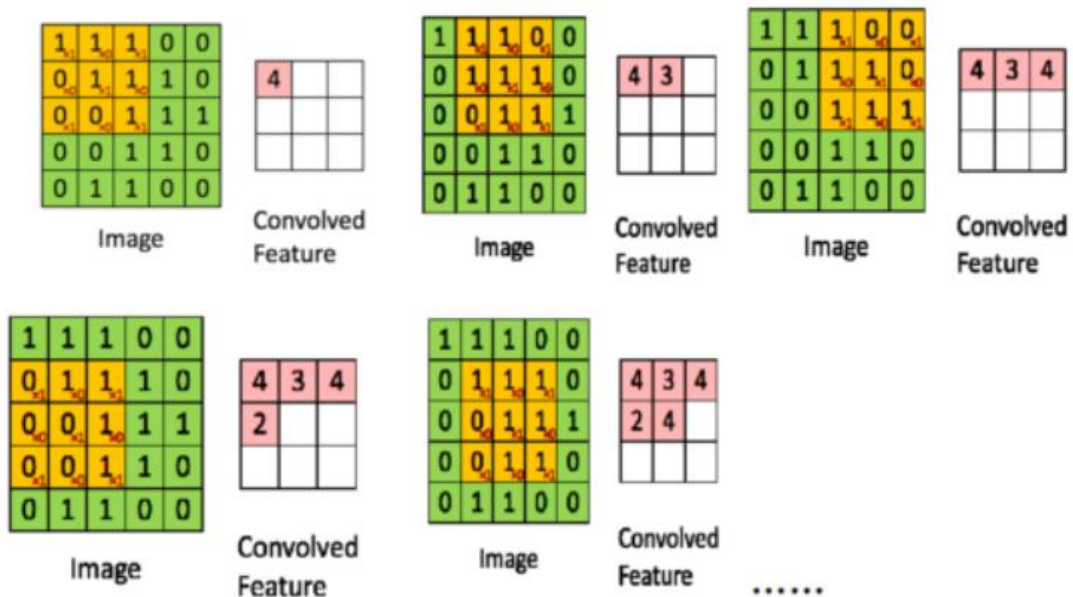
所以我感觉，与其说卷积是一种混合两种信息的过程，不如说它通过混合两种信息过滤掉我们不关心的部分（噪音）。

卷积是如何进行的？

原始图像的 size 为 5×5 ，滤波器的 kernel size 为 3×3 ，stride（步长）为 1，权重固定。生成的 feature map 的边长为： $(5-3) / 1 + 1 = 3$ 。



具体卷积过程如下图所示：

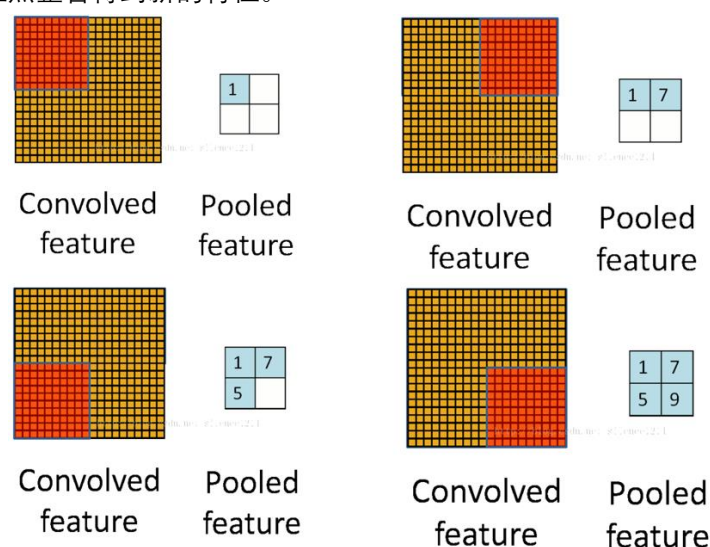


池化(Pooling)是什么？

我们之所以决定使用卷积后的特征是因为图像具有一种“静态性”的属性，这也就意味着在一个图像区域有用的特征极有可能在另一个区域同样适用。因此，为了描述大的图像，一个很自然的想法就是对不同位置的特征进行聚合统计。

例如，人们可以计算图像一个区域上的某个特定特征的平均值（或最大值）。这些概要统计特征不仅具有低得多的维度（相比使用所有提取到的特征），同时还会改善结果(不容易过拟合 over-fitting)。这种聚合的操作就叫做**池化 (pooling)**。有时也称为平均池化或者最大池化（取决于计算池化的方法）。池化的最终的结论是要把原来的维度减少到 $1/n$ 。

卷积层是对图像的一个邻域进行卷积得到图像的邻域特征，池化层就是使用 pooling 技术将小邻域内的特征点整合得到新的特征。



比如上方左侧矩阵 A 是 20*20 的矩阵要进行大小为 10*10 的池化，那么左侧图中的红色就是 10*10 的大小，对应到右侧的矩阵，右侧每个元素的值，是左侧红色矩阵每个元素的值得和再处于红色矩阵的元素个数，也就是平均值形式的池化。

池化的优点

- 1 显著减少参数数量，Pooling 的结果可以使得特征减少，参数减少。
- 2 池化单元具有平移不变性，Pooling 可以保持某种不变性（旋转、平移、伸缩等）

池化的种类

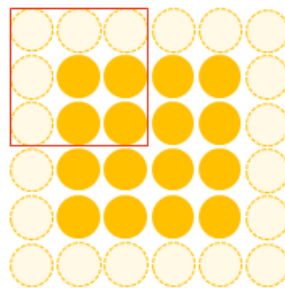
- **平均池化**：计算图像区域的平均值作为该区域池化后的值。
- **最大池化**：选图像区域的最大值作为该区域池化后的值。
- **随机池化**：只需对 feature map 中的元素按照其概率值大小随机选择，即元素值大的被选中的概率也大
- **重叠池化 (Overlapping Pooling)**：相邻池化窗口之间会有重叠区域
- **空金字塔池化 (Spatial Pyramid Pooling)**：空间金字塔池化可以把任何尺度的图像的卷积特征转化成相同维度，这不仅可以让 CNN 处理任意尺度的图像，还能避免 cropping 和 warping 操作，导致一些信息的丢失，具有重要的意义。

Zero Padding

如果不想最后减少维度，只希望卷积，怎么办呢？

比如：我们需要做一个 300×300 的原始矩阵，用一个 3×3 卷积核来扫，扫出来，按照之前公式，结果的矩阵应该是： 298×298 的矩阵，但是这样很难计算，减得也不多，反而增加我计算难度，还不如池化（pooling）来得干脆是吧！那我们就在 300×300 矩阵外面周围加一圈“0”，记住，是在外面外包一层“0”。重点是：这样的 300×300 就变成了 302×302 的矩阵，这样就可以完全避开卷积后那两层的抵消。

再举个例子，下面 4×4 的图片在边缘 Zero padding 一圈后，再用 3×3 的 filter 卷积后，得到的 Feature Map 尺寸依然是 4×4 不变。



进入卷积神经网络

卷积网络的核心思想是将：**局部感受野**、**权值共享**（或者权值复制）以及**时间或空间亚采样**这三种结构思想结合起来获得了**某种程度的位移、尺度、形变不变性**。

两个特点——

- **稀疏连接(Sparse Connectivity)**
- **权值共享(Shared Weights)**

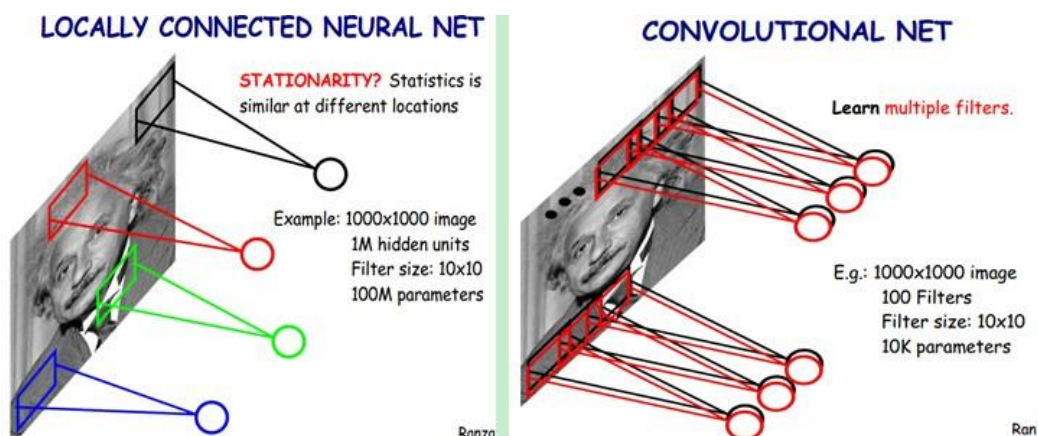
卷积神经网络包括如下形式的约束：

- 1、特征提取。**每一个神经元从上一层的局部接受域得到突触输入，因而迫使它提取**局部特征**。一旦一个特征被提取出来，只要它相对于其他特征的位置被近似地保留下来，它的精确位置就变得没有那么重要了。
- 2、特征映射。**网络的每一个计算层都是由多个特征映射组成的，每个特征映射都是平面形式的。平面中单独的神经元在约束下共享相同的突触权值集，这种结构形式具有如下的有益效果：a.平移不变性。b.自由参数数量的缩减(通过权值共享实现)。
- 3、子抽样。**每个卷积层后面跟着一个实现局部平均和子抽样的计算层，由此特征映射的分辨率降低。这种操作具有使特征映射的输出对平移和其他形式的变形的敏感度下降的作用。

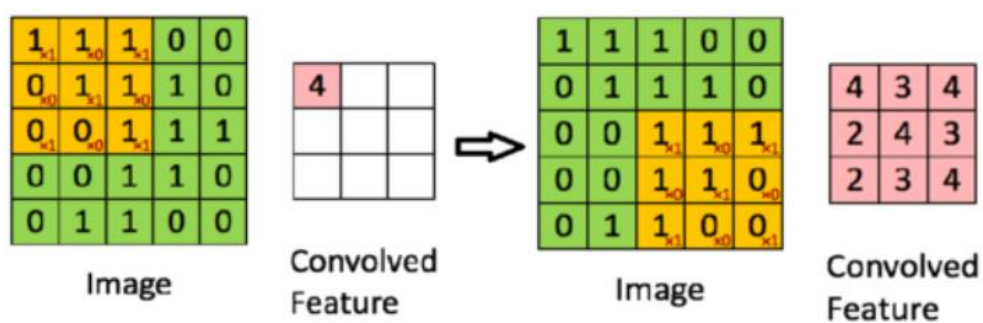
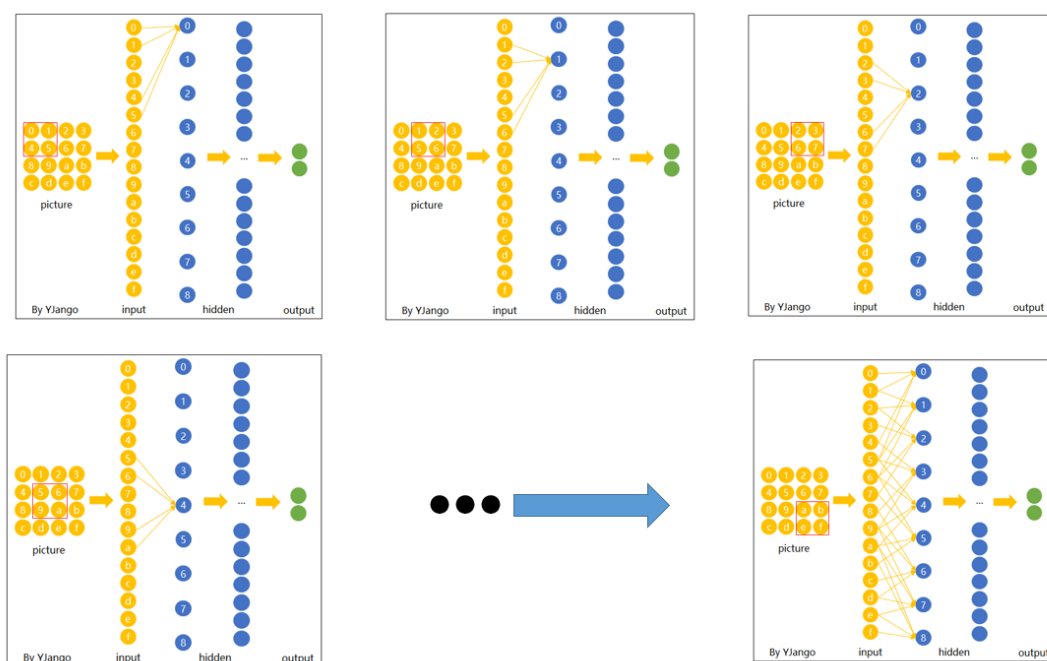
每一个神经元都不需要对全局图像做感受，**每个神经元只感受局部的图像区域**，然后在更高层，将这些感受不同局部的神经元综合起来就可以得到全局的信息了。这样，我们就可以减少连接的数目，也就是减少神经网络需要训练的权值参数的个数了。

假如每个隐层神经元只与 10×10 的局部区域像素相连，这里的 10×10 就是滤波器的 kernel size。stride 就是滤波器的步长，即从一个隐层神经元到相邻隐层神经元之间的移动长度。各个神经元之间的权重是相同的，即权重共享。这样对 kernel size 中的局部图像进行卷积操作后，就映射得到了一个表示该图像同一特征的一个 feature map。当又用一种滤波器进行相同的卷积操作后，就得到能代表该图像另一特征的一个 feature map。所以假设我们加到 100 种滤波器，每种滤波器的参数不一样，表示它提出输入图像的不同特征，例如不同的边缘。这样每种滤波器去卷积图像就得到对图像的不同特征的放映，我们称之为 Feature Map。所以 100 种卷积核就有 100 个 Feature Map。这 100 个 Feature Map 就组成了一层神经元。

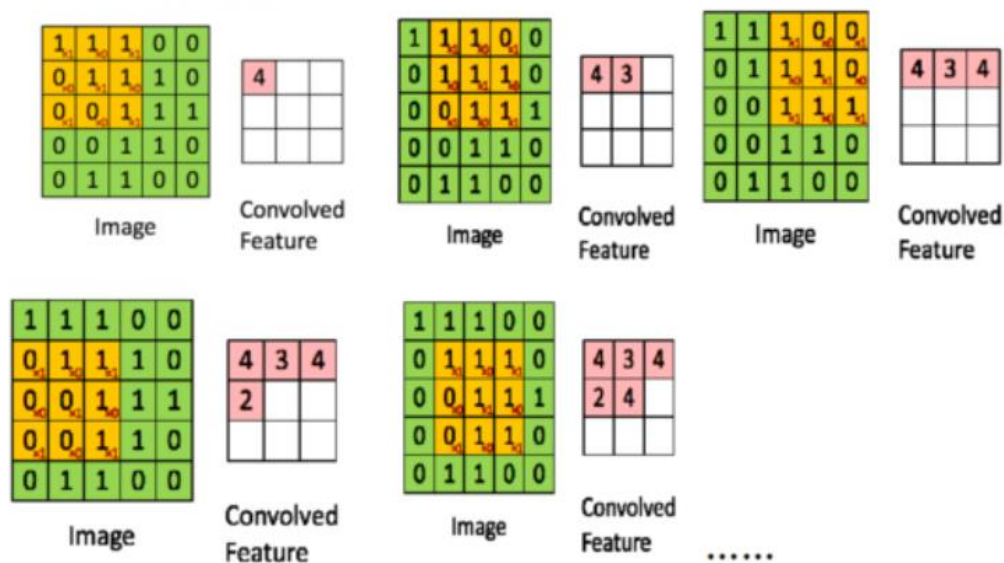
见下图右：不同的颜色表达不同的滤波器。



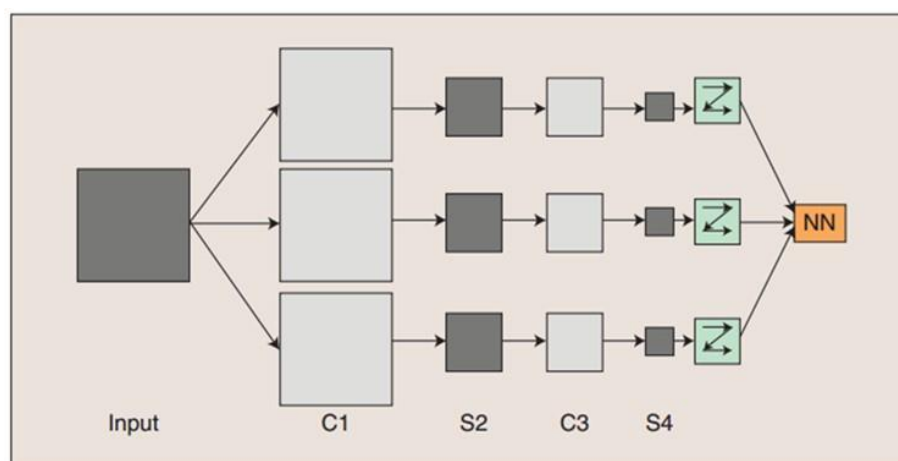
为了更好地理解，用两种不同形式的图展示卷积网络：



具体卷积过程如下图所示：



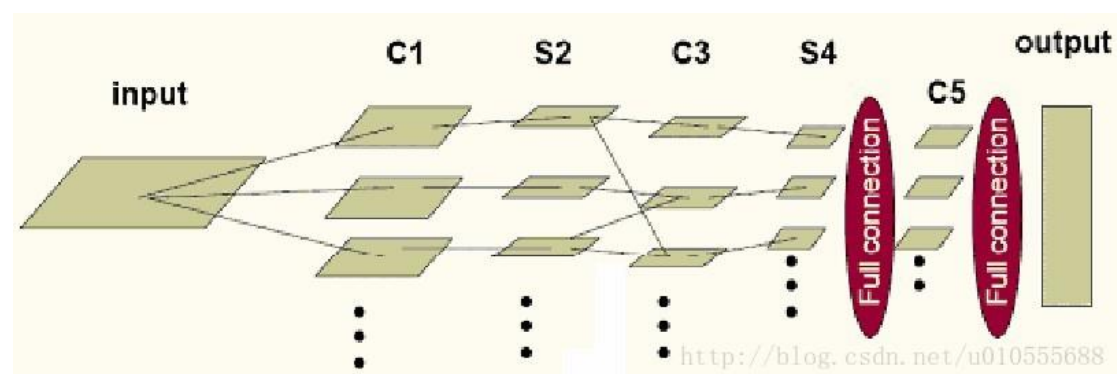
卷积神经网络是一个多层的神经网络，每层由多个二维平面组成，而每个平面由多个独立神经元组成。



输入图像通过和三个可训练的滤波器和可加偏置进行卷积，卷积后在 C1 层产生三个特征映射图，然后特征映射图中每组的四个像素再进行求和，加权值，加偏置，通过一个 Sigmoid 函数得到三个 S2 层的特征映射图。这些映射图再经过滤波得到 C3 层。这个层级结构再和 S2 一样产生 S4。最终，这些像素值被光栅化，并连接成一个向量输入到传统的神经网络，得到输出。

一般地，C 层为特征提取层，**每个神经元的输入与前一层的局部感受野相连，并提取该局部的特征，一旦该局部特征被提取后，它与其他特征间的位置关系也随之确定下来**；S 层是特征映射层，网络的每个计算层由多个特征映射组成，每个特征映射为一个平面，平面上所有神经元的权值相等。特征映射结构采用影响函数核小的 sigmoid 函数作为卷积网络的激活函数，使得特征映射具有位移不变性。

此外，由于一个映射面上的神经元共享权值，因而减少了网络自由参数的个数，降低了网络参数选择的复杂度。卷积神经网络中的每一个特征提取层（C-层）都紧跟着一个用来求局部平均与二次提取的计算层（S-层），这种特有的两次特征提取结构使网络在识别时对输入样本有较高的畸变容忍能力。

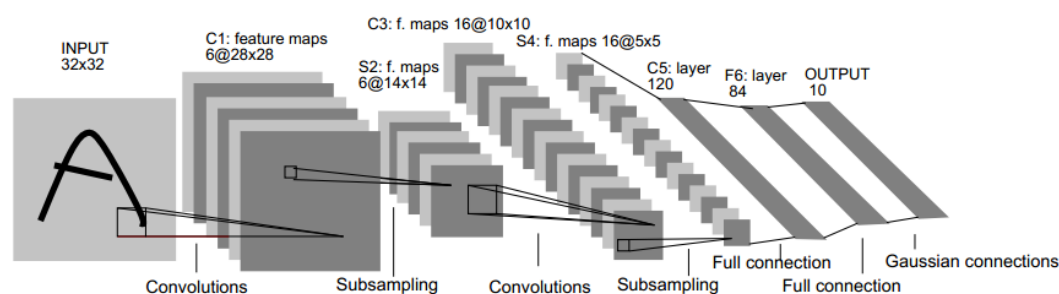


其中，input 到 C1、S4 到 C5、C5 到 output 是**全连接**，C1 到 S2、C3 到 S4 是一一对应的连接，S2 到 C3 为了消除网络对称性，去掉了一部分连接，可以让特征映射更具多样性。需要注意的是 C5 卷积核的尺寸要和 S4 的输出相同，只有这样才能保证输出是一维向量

隐层的参数个数和隐层的神经元个数无关，只和滤波器的大小、滤波器种类的多少有关。

那么隐层的神经元个数怎么确定呢？它和原图像，也就是输入的大小（神经元个数）、滤波器的大小和滤波器在图像中的滑动步长都有关！例如，我的图像是 1000x1000 像素，而滤波器大小是 10x10，假设滤波器没有重叠，也就是步长为 10，这样隐层的神经元个数就是 $(1000 \times 1000) / (10 \times 10) = 100 \times 100$ 个神经元了。注意了，这只是一种滤波器，也就是一个 Feature Map 的神经元个数，如果 100 个 Feature Map 就是 100 倍了。由此可见，图像越大，神经元个数和需要训练的权值参数个数的贫富差距就越大。

例：文字识别系统 LeNet-5



输入图像是 32x32 的大小，局部滑动窗的大小是 5x5 的，由于不考虑对图像的边界进行拓展，则滑动窗将有 28x28 个不同的位置，也就是 C1 层的大小是 28x28。**(32-5) / 1 + 1 = 28**。这里设定有 6 个不同的 C1 层，由 6 个特征图 Feature Map 构成。**特征图中每个神经元与输入中 5*5 的邻域相连。**

每一个 C1 层内的权值是相同的。

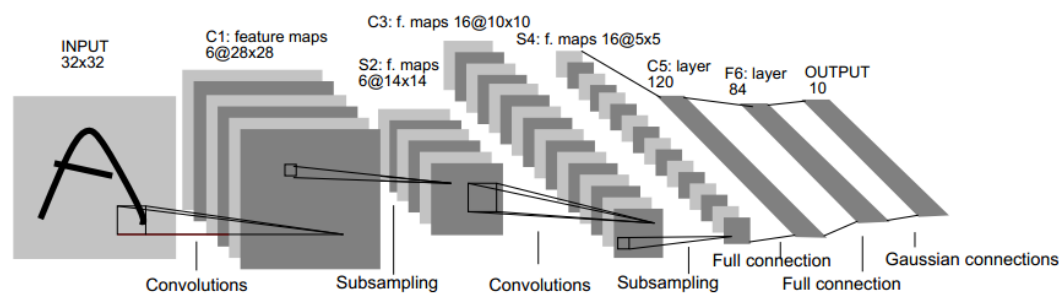
S2 层是一个下采样层。简单的说，由 4 个点下采样为 1 个点，也就是 4 个数的加权平均。特征图中的每个单元与 C1 中相对应特征图的 2*2 邻域相连接。但在 LeNet-5 系统，下采样层比较复杂，因为这 4 个加权系数也需要学习得到，这显然增加了模型的复杂度。在斯坦福关于深度学习的教程中，这个过程叫做 **Pool**。

C3 层也是一个卷积层，它同样通过 5x5 的卷积核去卷积层 S2，然后得到的特征 map 就只有 10x10 个神经元，但是它有 16 种不同的卷积核，所以就存在 16 个特征 map 了。**C3 中的每个特征 map 是连接到 S2 中的所有 6 个或者几个特征 map 的，表示本层的特征 map 是上一层提取到的特征 map 的不同组合。**

刚才说 C3 中每个特征图由 S2 中所有 6 个或者几个特征 map 组合而成。**为什么不把 S2 中的每个特征图连接到每个 C3 的特征图呢？**原因有 2 点。第一，不完全的连接机制将连接的数量保持在合理的范围内。第二，也是最重要的，**其破坏了网络的对称性**。由于不同的特征图有不同的输入，所以迫使他们抽取不同的特征（希望是互补的）。

试想一下，如果 S2 层只有 1 个平面，那么由 S2 层得到 C3 就和由输入层得到 C1 层是完全一样的。但是，S2 层由多层，那么，我们只需要按照一定的顺利组合这些层就可以了。具体的组合规则，在 LeNet-5 系统中给出了下面的表格：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X



C5 层是一个卷积层，有 120 个特征图。每个单元与 S4 层的全部 16 个单元的 5*5 邻域相连。由于 S4 层特征图的大小也为 5*5（同滤波器一样），故 C5 特征图的大小为 1*1：这构成了 S4 和 C5 之间的全连接。之所以仍将 C5 标示为卷积层而非全相联层，是因为如果 LeNet-5 的输入变大，而其他的保持不变，那么此时特征图的维数就会比 1*1 大。C5 层有 48120 个可训练连接。



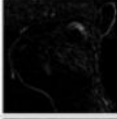


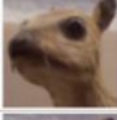

F6 层有 84 个单元（之所以选这个数字的原因来自于输出层的设计），与 C5 层全相连。

最后，输出层由欧式径向基函数（Euclidean Radial Basis Function）单元组成，每类一个单元，每个有 84 个输入。

更加详细过程参考 <http://blog.csdn.net/u010555688/article/details/24848367>

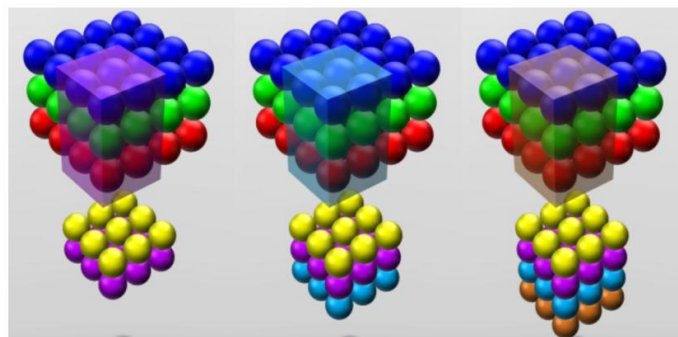
卷积核/滤波器

可以从下面这张图中感受到不同数值的 filters 所卷积过后的 Feature Map 可以探测边缘，棱角，模糊，突出等概念。

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

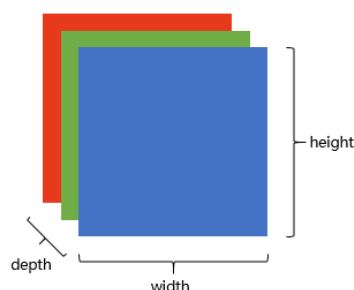
多 filters

卷积层的输出也可以是和输入一样的，是 depth 为复数的长方体。当我们增加一个 filter（紫色表示）后，就又可以得到一个 Feature Map。将不同 filters 所卷积得到的 Feature Maps 按顺序堆叠后，就得到了一个卷积层的最终输出。



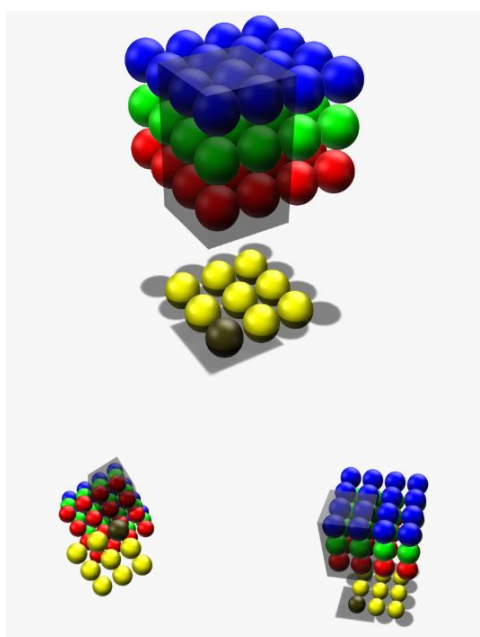
Depth 维的处理

现在我们已经知道了 depth 维度只有 1 的灰度图是如何处理的。但前文提过，图片的普遍表达方式是下图这样有 3 个 channels 的 RGB 颜色模型。当 depth 为复数的时候，每个 feature detector 是如何卷积的？



2x2 所表达的 filter size 中，一个 2 表示 width 维上的局部连接数，另一个 2 表示 height 维上的局部连接数，并却没有 depth 维上的局部连接数，是因为 depth 维上并非局部，而是**全部连接**的。在 2D 卷积中，filter 在张量的 width 维, height 维上是局部连接，在 depth 维上是贯穿全部 channels 的。

在输入 depth 为 1 时：被 filter size 为 2x2 所圈中的 4 个输入节点连接到 1 个输出节点上。
在输入 depth 为 3 时：被 filter size 为 2x2，但是贯穿 3 个 channels 后，所圈中的 12 个输入节点连接到 1 个输出节点上。



注意：三个 channels 的权重并不共享。即当深度变为 3 后，权重也跟着扩增到了三组，如下所示，不同 channels 用的是自己的权重。式子中增加的角标 r,g,b 分别表示 red channel, green channel, blue channel 的权重。

$$\begin{bmatrix} w_{r1} & w_{r2} \\ w_{r3} & w_{r4} \end{bmatrix}, \begin{bmatrix} w_{g1} & w_{g2} \\ w_{g3} & w_{g4} \end{bmatrix}, \begin{bmatrix} w_{b1} & w_{b2} \\ w_{b3} & w_{b4} \end{bmatrix}$$

Reference:

- [1] <http://blog.csdn.net/u014696921/article/details/71908332?locationNum=8&fps=1>
- [2] <http://blog.csdn.net/yingyujianmo/article/details/44945315>
- [3] <http://blog.csdn.net/yingyujianmo/article/details/44945521>
- [4] <http://www.knowsky.com/1049839.html>
- [5] http://blog.csdn.net/errors_in_life/article/details/65950699
- [6] <https://zhuanlan.zhihu.com/p/27642620>