deeplearning.ai

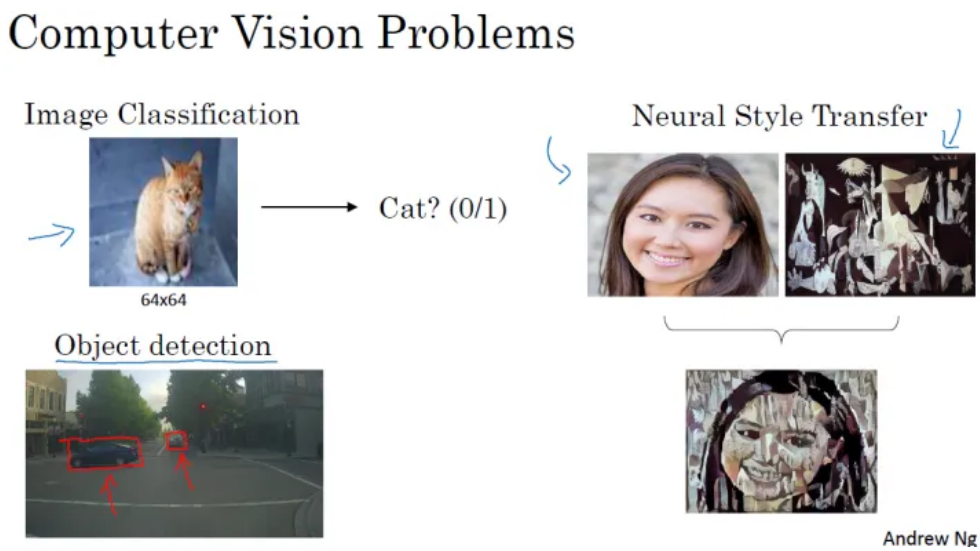
吴恩达 DeepLearning.ai 课程提炼笔记（4-1）卷积神经网络 --- 卷积神经网络基础

Ng新课终于发布了，撒花！以下为在Coursera上吴恩达老师的 deeplearning.ai 课程项目中，第四部分《卷积神经网络》第一周课程 “卷积神经网络基础” 关键点的笔记。本次笔记几乎涵盖了所有视频课程的内容。在阅读以下笔记的同时，强烈建议学习吴恩达老师的视频课程，视频请至 Coursera 或者 网易云课堂。

---

1. 计算机视觉

计算机视觉（Computer Vision）包含很多不同类别的问题，如图片分类、目标检测、图片风格迁移等等。



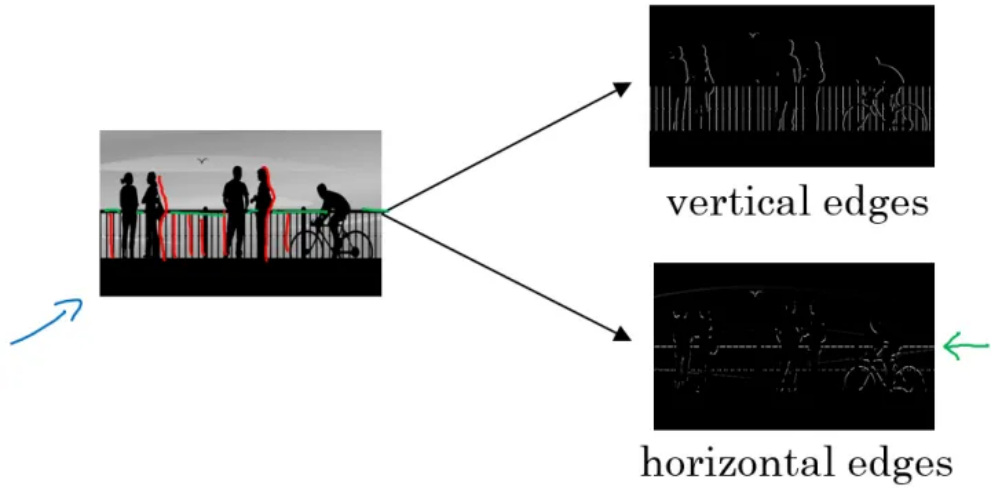
对于小尺寸的图片问题，也许我们用深度神经网络的结构可以较为简单的解决一定的问题。但是当应用在大尺寸的图片上，输入规模将变得十分庞大，使用神经网络将会有非常多的参数需要去学习，这个时候神经网络就不再适用。

卷积神经网络在计算机视觉问题上是一个非常好的网络结构。

2. 边缘检测示例

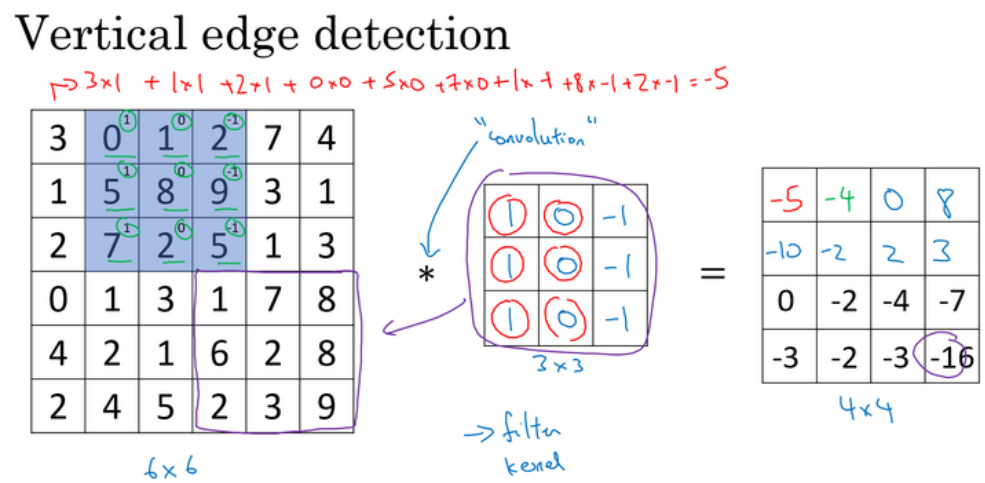
卷积运算是卷积神经网络的基本组成部分。下面以边缘检测的例子来介绍卷积运算。

所谓边缘检测，在下面的图中，分别通过垂直边缘检测和水平边缘检测得到不同的结果：



垂直边缘检测：

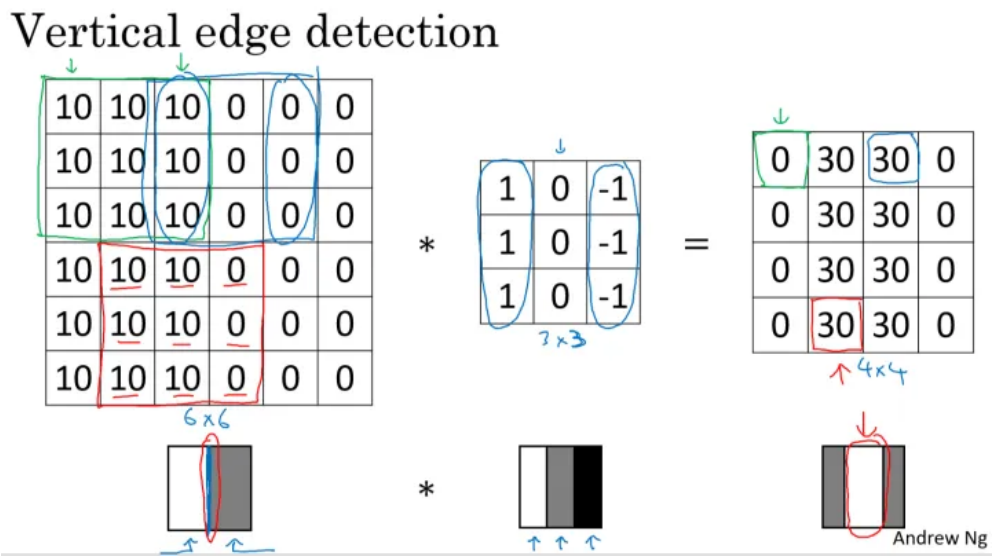
假设对于一个 6×6 大小的图片（以数字表示），以及一个 3×3 大小的 **filter**（卷积核） 进行卷积运算，以 “ $*$ ” 符号表示。图片和垂直边缘检测器分别如左和中矩阵所示：



filter 不断地和其大小相同的部分做对应元素的乘法运算并求和，最终得到的数字相当于新图片的一个像素值，如右矩阵所示，最终得到一个 4×4 大小的图片。

边缘检测的原理：

以一个有一条垂直边缘线的简单图片来说明。通过垂直边缘 **filter** 我们得到的最终结果图片可以明显地将边缘和非边缘区分出来：



卷积运算提供了一个方便的方法来检测图像中的边缘，成为卷积神经网络中重要的一部分。

多种边缘检测：

垂直和水平边缘检测

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

更复杂的 **filter**

1	0	-1
1	0	-1
1	0	-1

1	0	-1
2	0	-2
1	0	-1

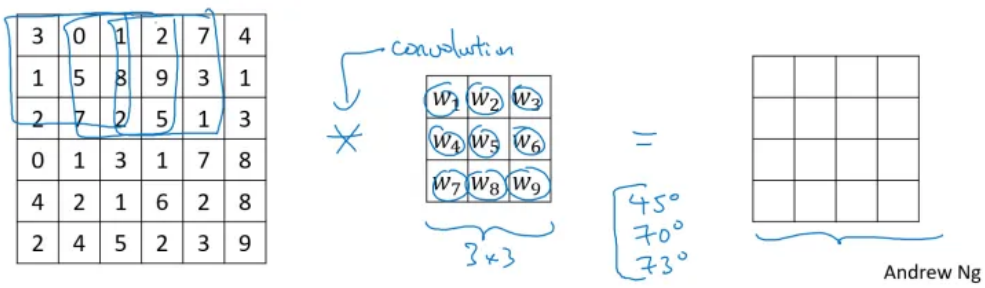
Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

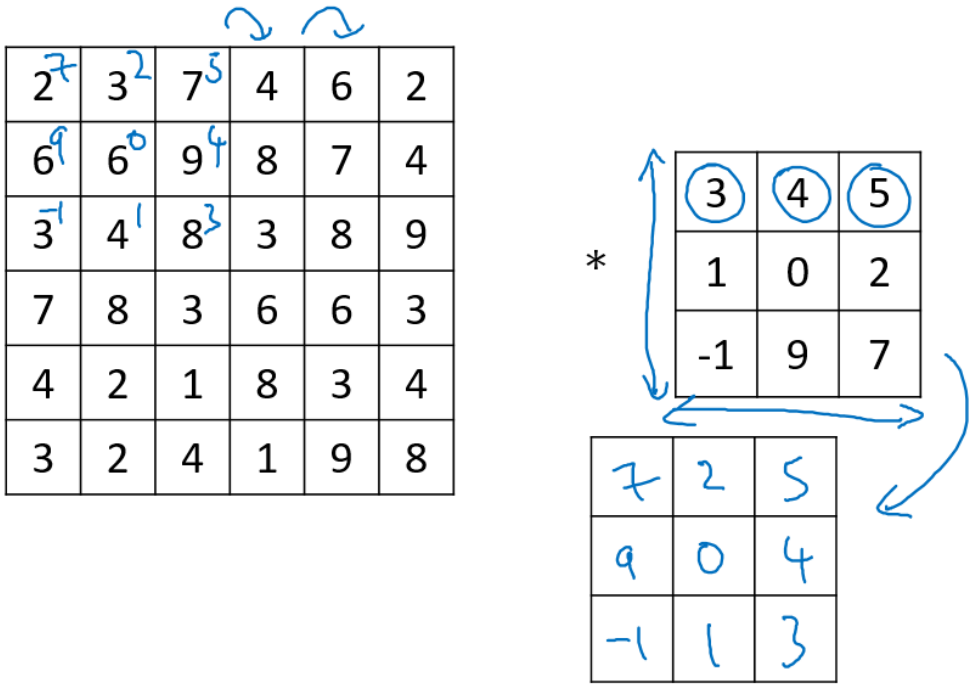
对于复杂的图片，我们可以直接将 **filter** 中的数字直接看作是需要学习的参数，其可以学习到对于图片检测相比上面filter更好的更复杂的 **filter** ，如相对于水平和垂直检测器，我们训练的 filter 参数也许可以知道不同角度的边缘。

通过卷积运算，在卷积神经网络中通过反向传播算法，可以学习到相应于目标结果的 **filter**，将其应用于整个图片，输出其提取到的所有有用的特征。



卷积和互相关：

在数学定义上，矩阵的**卷积**（convolution）操作为首先将卷积核同时在水平和垂直方向上进行翻转，构成一个卷积核的镜像，然后使用该镜像再和前面的矩阵进行移动相乘求和操作。如下面例子所示：



在深度学习中，我们称为的卷积运算实则没有卷积核变换为镜像的这一步操作，因为在权重学习的角度，变换是没有必要的。深度学习的卷积操作在数学上准确度来说称为**互相关**（cross-correlation）。

3. Padding

没有Padding的缺点：

- 每次卷积操作，图片会缩小；

就前面的例子来说，6×6 大小的图片，经过 3×3 大小的 filter，缩小成了 4×4 大小

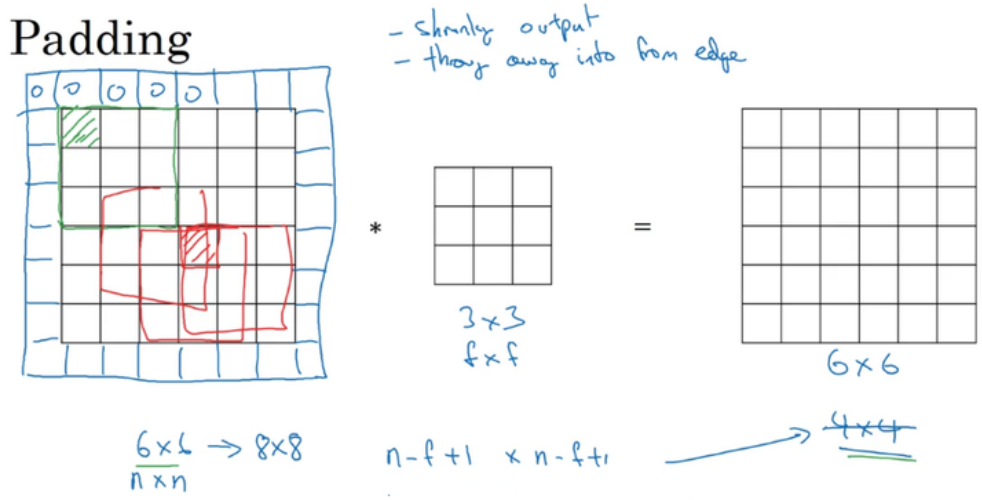
图片：$n \times n --> (n-f+1) \times (n-f+1)$

- 角落和边缘位置的像素进行卷积运算的次数少，可能会丢失有用信息。

其中，**n** 表示图片的长或宽的大小，**f** 表示filter的长或宽的大小。

加Padding：

为了解决上面的两个缺点，我们在进行卷积运算前为图片加padding，包围角落和边缘的像素，使得通过filter的卷积运算后，图片大小不变，也不会丢失角落和边沿的信息。



以**p**表示 Padding 的值，则输入 $n \times n$ 大小的图片，最终得到的图片大小为 $(n+2p-f+1) \times (n+2p-f+1)$，为使图片大小保持不变，需根据filter的大小调整p的值。

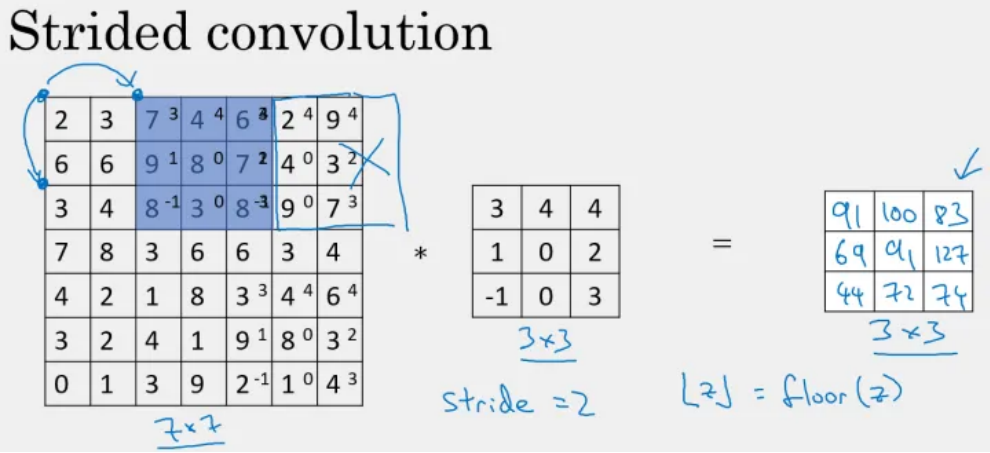
Valid / Same 卷积：

- **Valid**：no padding；（$n \times n --> (n-f+1) \times (n-f+1)$）
- **Same**：padding，输出与输入图片大小相同，（$p=(f-1)/2$）。在计算机视觉中，一般来说padding的值为奇数（因为filter一般为奇数）

4. 卷积步长（stride）

卷积的步长是构建卷积神经网络的一个基本的操作。

如前面的例子中，我们使用的 stride=1，每次的卷积运算以1个步长进行移动。下面是 stride=2 时对图片进行卷积的结果：



以**s**表示stride的大小，那么在进行卷积运算后，图片的变化为：

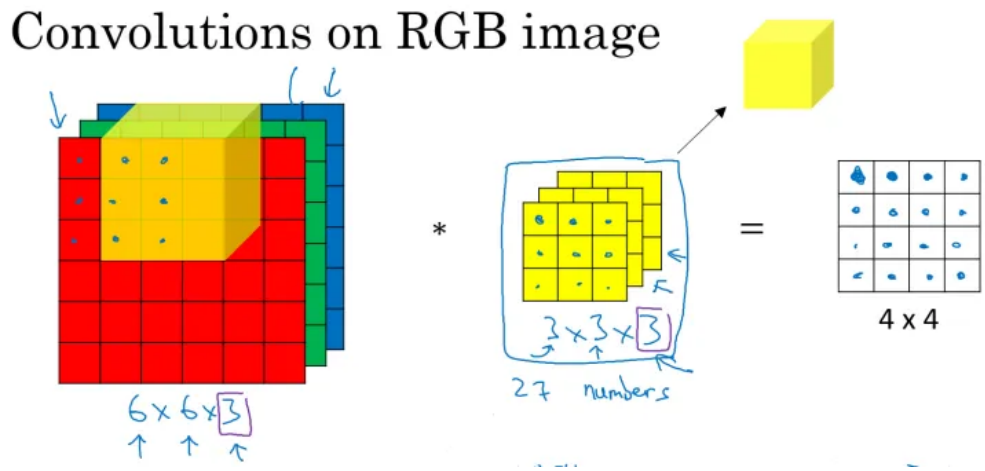
$$n \times n --> \left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor$$

注意，在当 $padding \neq 1$ 时，若移动的窗口落在图片外面，则不要再进行相乘的操作，丢弃边缘的数值信息，所以输出图片的最终维度为**向下取整**。

5. 立体卷积

卷积核的通道数：

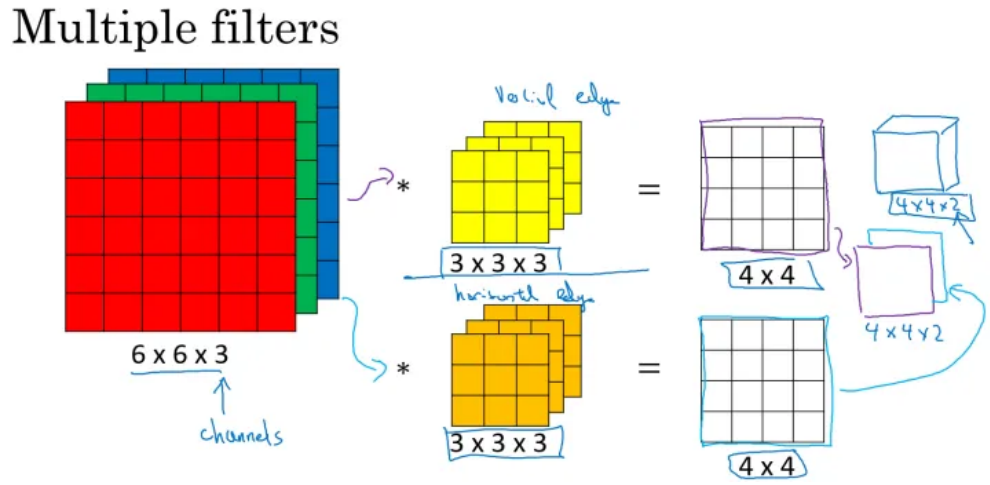
对于灰色图像中，卷积核和图像均是二维的。而应用于彩色图像中，因为图片有R、G、B三个颜色通道，所以此时的卷积核应为三维卷积核。



卷积核的第三个维度需要与进行卷积运算的图片的通道数相同。

多卷积核：

单个卷积核应用于图片时，提取图片特定的特征，不同的卷积核提取不同的特征。如两个大小均为 $3 \times 3 \times 3$ 的卷积核分别提取图片的垂直边缘和水平边缘。



由图可知，最终提取到彩色图片的垂直特征图和水平特征图，得到有2个通道的 4×4 大小的特征图片。

Summary：

图片： $(n \times n \times n_c) * (f \times f \times n_c) \text{——} > (n - f + 1) \times (n - f + 1) \times n'_c$

其中， n_c 表示通道的数量， n'_c 表示下一层的通道数，同时也等于本层卷积核的个数。

6. 简单卷积网络

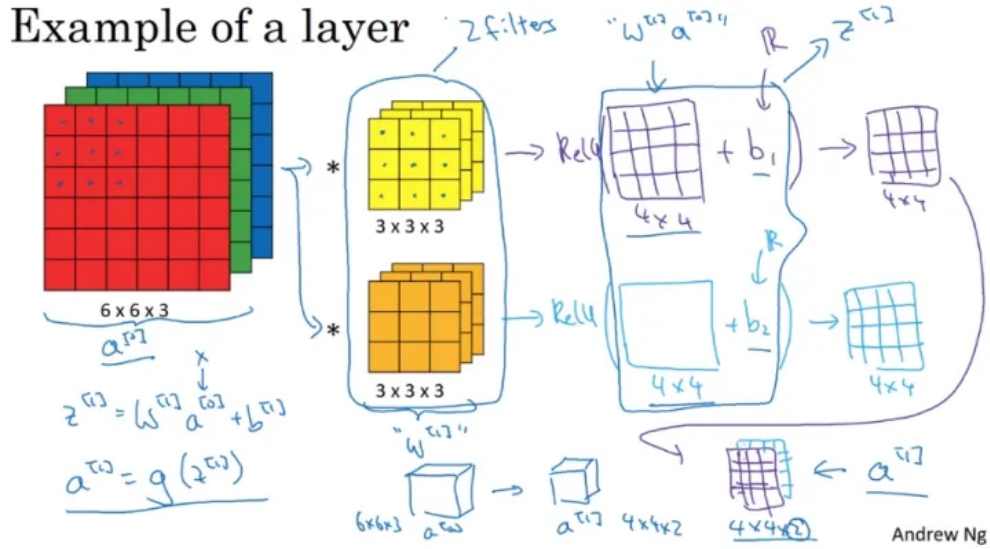
单层卷积网络的例子：

和普通的神经网络单层前向传播的过程类似，卷积神经网络也是一个先由输入和权重及偏置做线性运算，然后得到的结果输入一个激活函数中，得到最终的输出：

$$z^{[1]} = w^{[1]}a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

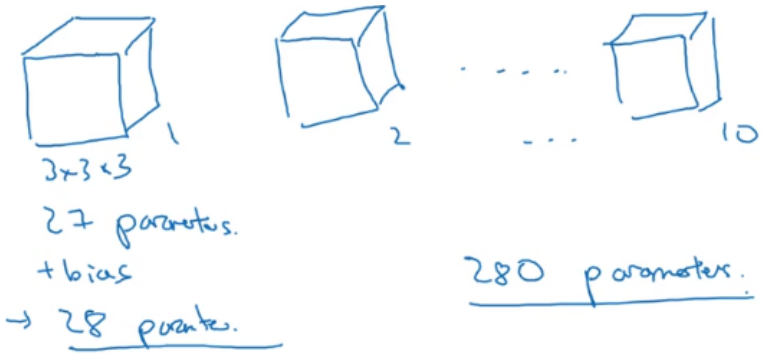
不同点是：在卷积神经网络中，权重和输入进行的是卷积运算。



单层卷积的参数个数：

在一个卷积层中，如果我们有10个 $3 \times 3 \times 3$ 大小的卷积核，那么加上每个卷积核对应的偏置，则对于一个卷积层，我们共有的参数个数为：

$$(3 \times 3 \times 3 + 1) \times 10 = 280$$



无论图片大小是多少，该例子中的卷积层参数个数一直都是280个，相对于普通的神经网络，卷积神经网络的参数个数要少很多。

标记的总结：

如果 l 表示一个卷积层：

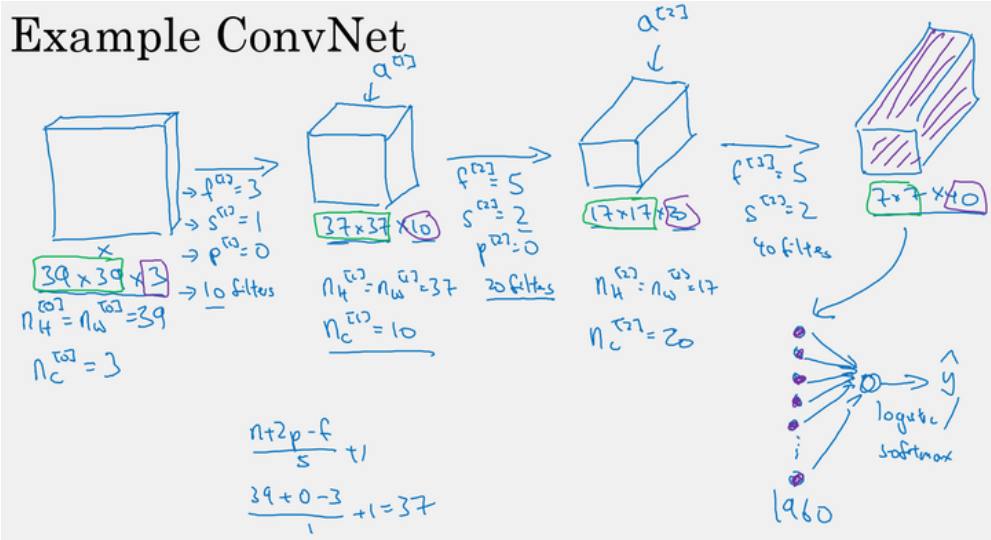
- $f^{[l]}$ ： filter 的大小；
- $p^{[l]}$ ： padding；
- $s^{[l]}$ ： 步长（stride）；
- 卷积核的个数： $n_C^{[l]}$ ；
- filter大小： $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$ ；
- 激活值（Activations）： $a^{[l]} -> n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$；

- 权重（Weights）： $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$ ；
- 偏置（bias）： $n_C^{[l]}——(1,1,1,n_C^{[l]})$

- Input： $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$ ；
- Output： $n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$ ；
- 其中， $n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$ ， $n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$ 。

简单卷积网络示例：

多层卷积构成卷积神经网络，下面是一个卷积神经网络的例子：



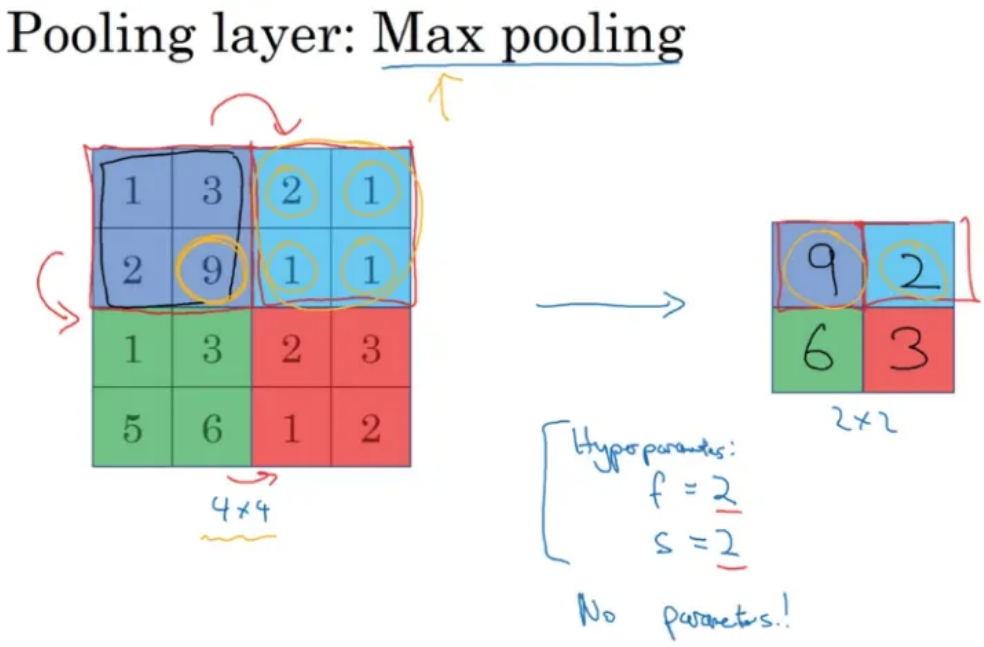
卷积网络层的类型：

- 卷积层（Convolution），Conv；
- 池化层（Pooling），Pool；
- 全连接层（Fully connected）：Fc；

7. 池化层

最大池化（Max pooling）：

最大池化是对前一层得到的特征图进行池化减小，仅由当前小区域内的最大值来代表最终池化后的值。

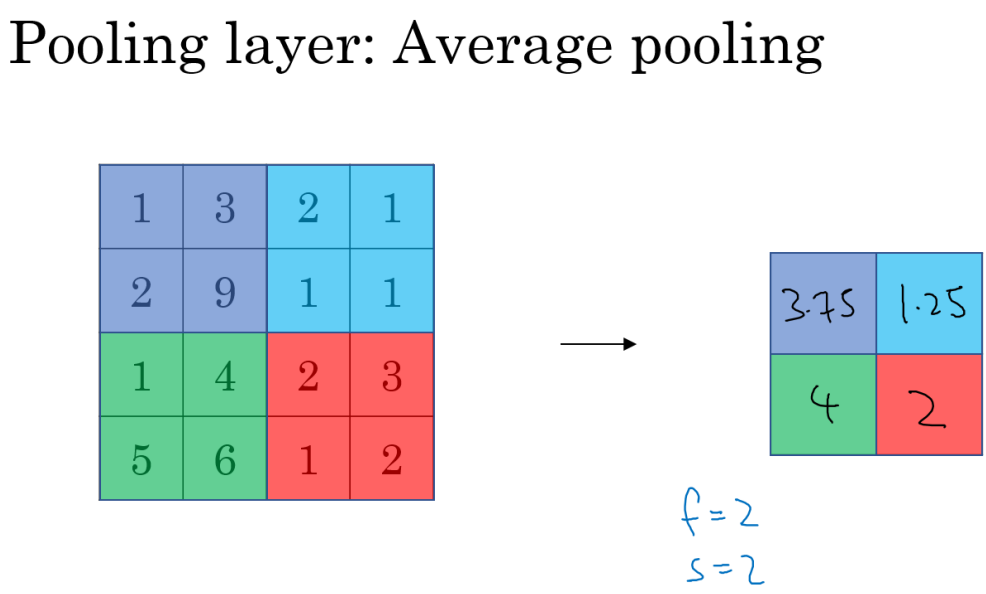


在最大池化中，有一组超参数需要进行调整，其中， f 表示池化的大小， s 表示步长。

- 池化前： $n \times n$ ；
- 池化后： $\left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s}+1 \right\rfloor$ 。

平均池化（Average pooling）：

平均池化与最大池化唯一不同的是其选取的是小区域内的均值来代表该区域内的值。



池化 Summary：

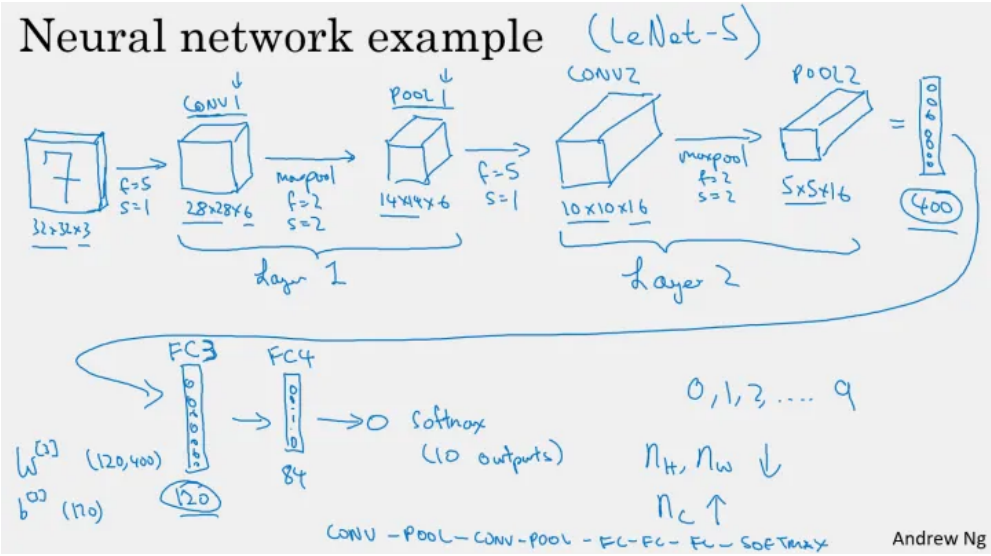
池化层的超参数：

- f ： filter的大小；
- s ： stride大小；
- 最大池化或者平均池化；
- p ： padding，这里要注意，几乎很少使用。

注意，池化层没有需要学习的参数。

8. 卷积神经网络示例

这里以 **LeNet-5** 为例，给出一个完整的卷积神经网络。



构建深度卷积的模式：

- 随着网络的深入，提取的特征图片大小将会逐渐减小，但同时通道数量应随之增加；
- Conv——Pool——Conv——Pool——Fc——Fc——Fc——softmax。

卷积神经网络的参数：

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

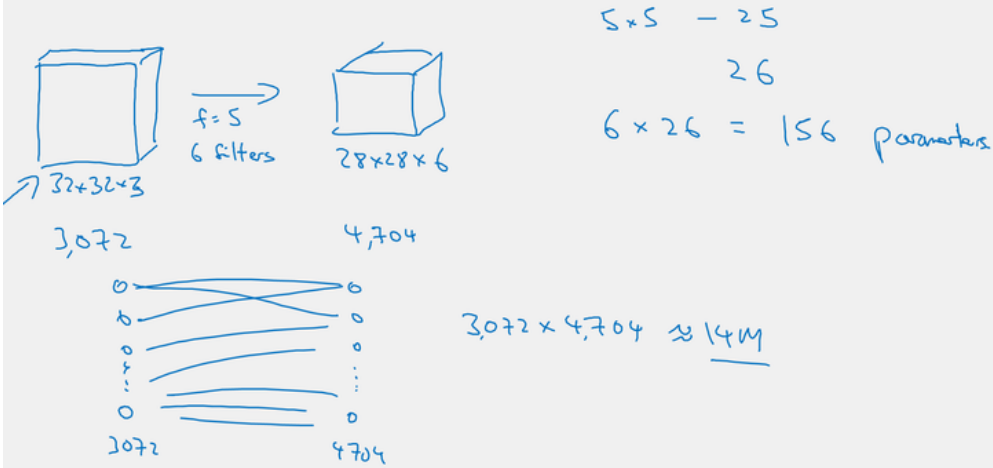
根据上表我们可以看出，对于卷积卷积神经网络的参数：

- 在卷积层，仅有少量的参数；
- 在池化层，没有参数；
- 在全连接层，存在大量的参数。

9. 使用卷积神经网络

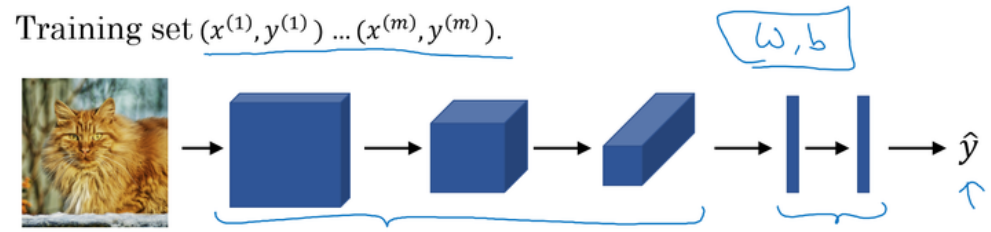
参数少的优势：

与普通的全连接神经网络相比，卷积神经网络的参数更少。如图中的例子，卷积神经网络仅有 $6 \times (5 \times 5 + 1) = 156$ 个参数，而普通的全连接网络有 $3072 \times 4704 \approx 14M$ 个参数。



- **参数共享**：一个特征检测器（filter）对图片的一部分有用的同时也有可能对图片的另外一部分有用。
- **连接的稀疏性**：在每一层中，每个输出值只取决于少量的输入。

训练卷积神经网络：



$$\text{Cost } J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

我们将训练集输入到卷积神经网络中，对网络进行训练。利用梯度下降（Adam、momentum等优化算法）最小化代价函数来寻找网络的最优参数。

编辑于 2017-11-08 08:56

深度学习（Deep Learning）　人工智能　卷积神经网络（CNN）