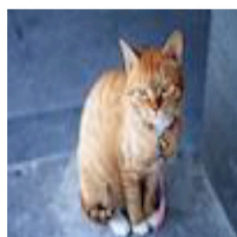


1. 计算机视觉

计算机视觉（**Computer Vision**）的高速发展标志着新型应用产生的可能，例如自动驾驶、人脸识别、创造新的艺术风格。人们对于计算机视觉的研究也催生了很多计算机视觉与其他领域的交叉成果。一般的计算机视觉问题包括以下几类：

Image Classification



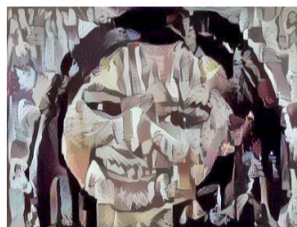
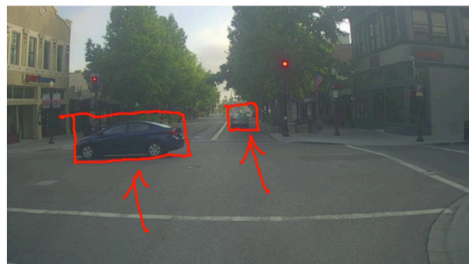
64x64

→ Cat? (0/1)

Neural Style Transfer



Object detection



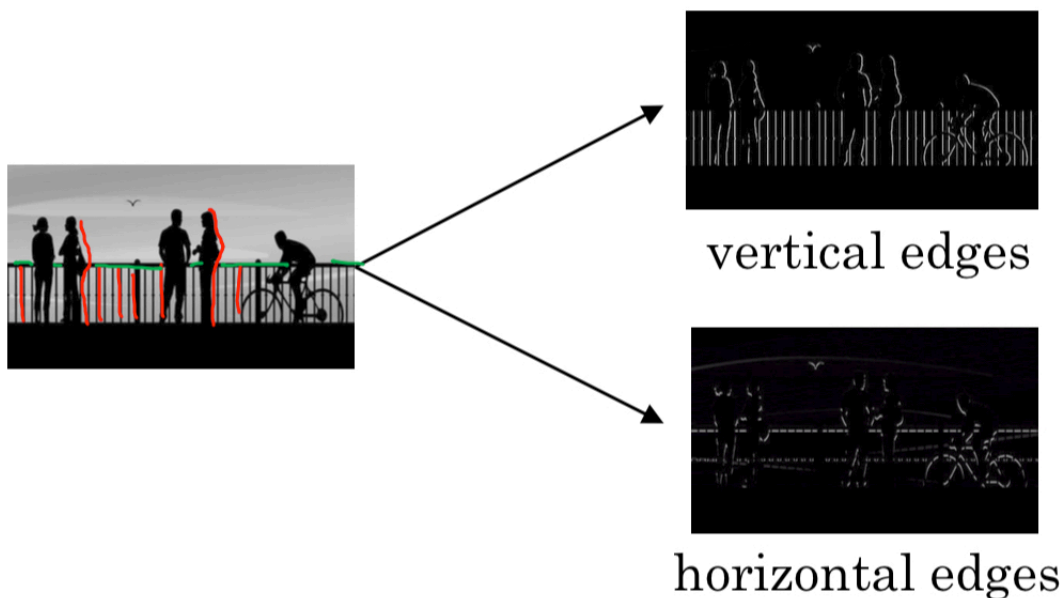
对于小尺寸图片，用深度学习可以较为简单地解决这一问题。但是当图片尺寸变大，比如一个 $1000 \times 1000 \times 3$ 的图片，神经网络的输入层维度高达300万，使得网络参数非常庞大，从而导致下面两个后果：

1. 神经网络结构复杂，而数据量相对较少，容易过拟合（overfitting）；
2. 所需内存和计算量非常巨大。

此时，**卷积神经网络（Convolutional Neural Network）**可以很好解决这一问题。

2. 边缘检测示例

卷积运算是卷积神经网络最基本的组成部分，使用边缘检测作为入门样例。图片最常做的边缘检测有两类：垂直边缘检测和水平边缘检测：



图片的边缘检测可以通过与中间的**滤波器(filter)**做卷积来实现。以垂直边缘检测为例：

假设原图片大小是 6×6 ，滤波器(又称卷积核，filter)的大小为 3×3 ，经过卷积运算后得到的图片大小为 4×4 。

示例： $-5 = 3 \times 1 + 0 \times 0 + 1 \times (-1) + 1 \times 1 + 5 \times 0 + 8 \times (-1) + 2 \times 1 + 7 \times 0 + 2 \times (-1)$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6×6

*

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

3×3
filter
kernel

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

下图对应一个垂直边缘检测的例子：

Vertical edge detection


10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

 $*$

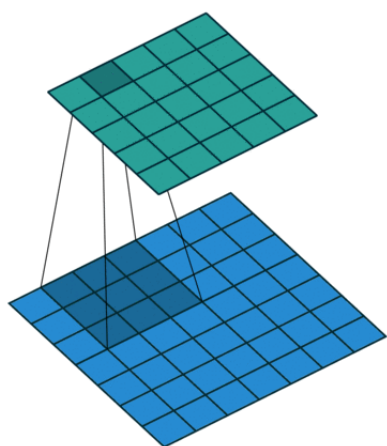
1	0	-1
1	0	-1
1	0	-1

 $=$

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



下面两张动态图可以直观理解卷积运算的过程：



1	1 _{x1}	1 _{x0}	0 _{x1}	0
0	1 _{x0}	1 _{x1}	1 _{x0}	0
0	0 _{x1}	1 _{x0}	1 _{x1}	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	

Convolved
Feature

Note: 图中的 * 表示的是卷积运算，但该符号在计算机中一般表示乘法。在不同的深度学习框架中，卷积操作的 API 定义可能不同：

- 在 Python 中，卷积用 `conv_forward()` 表示；
- 在 Tensorflow 中，卷积用 `tf.nn.conv2d()` 表示；
- 在 keras 中，卷积用 `Conv2D()` 表示。

3. 更多边缘检测

垂直和水平边缘检测过滤器：

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

Vertical

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

Horizontal

其他常用的滤波器还有 Sobel 滤波器和 Scharr 滤波器。它们增加了中间行的权重，以提高结果的稳健性：

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

Sobel filter

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

Scharr filter

滤波器中的值还可以设置为参数，通过模型训练来得到。这样，神经网络使用反向传播算法可以学习到一些低级特征，从而实现对图片所有边缘特征的检测，而不仅限于垂直边缘和水平边缘：

Q:如何同时做到水平和垂直边缘检测?

多卷积核

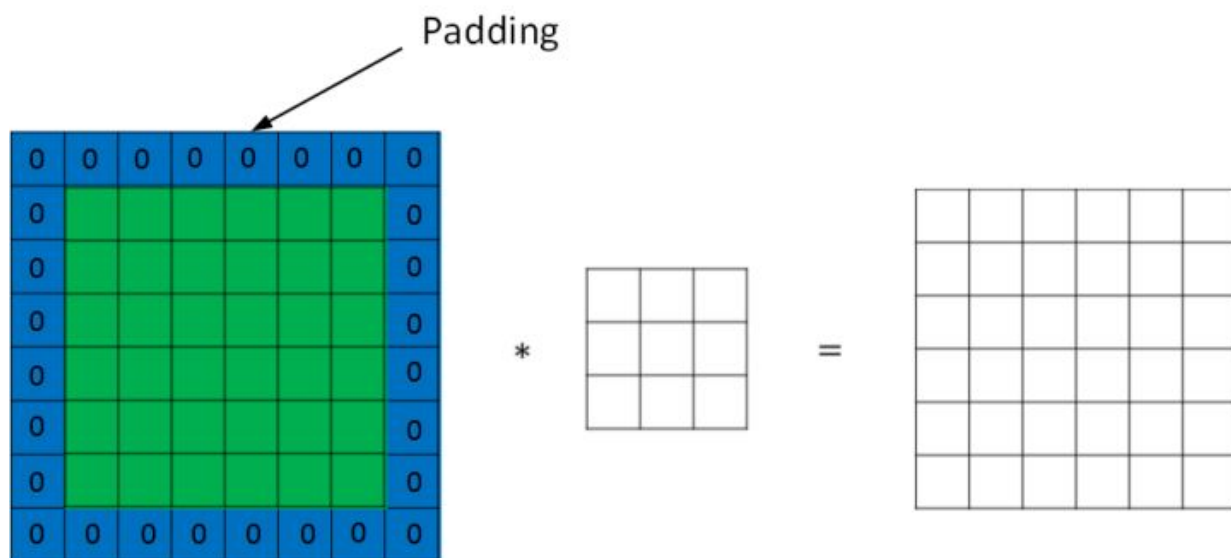
4. Padding

假设图片的大小为 $n \times n$ ，滤波器的大小为 $f \times f$ ，则卷积后输出的图片大小为 $(n - f + 1) \times (n - f + 1)$

这样就有存在两个问题：

- 每次卷积操作后，图片的尺寸都缩小；
- 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。

为了解决这些问题，可以在进行卷积操作前，对原始图片在边界上进行填充（**Padding**），以增加矩阵的大小。通常将 0 作为填充值。



设每个方向扩展像素点数量为 p ，则填充后原始图片的大小为 $(n + 2p) \times (n + 2p)$ ，滤波器大小保持 $f \times f$ 不变，则输出图片大小为 $(n + 2p - f + 1) \times (n + 2p - f + 1)$ 。

两种padding方式：

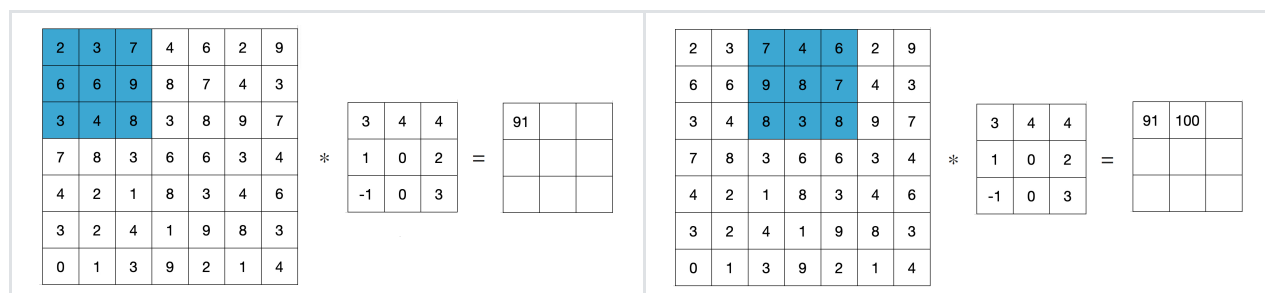
- **Valid**：不填充($p = 0$)，直接进行卷积；
- **Same**：进行填充，使得卷积后的结果与输入维度一致，则 $p = \frac{f-1}{2}$

在计算机视觉任务中， f 一般为奇数，一来**same** 卷积的 $p = \frac{f-1}{2}$ 能被整除；二是这样的卷积核有中心点便于指示位置。

5. 卷积步长

卷积过程中，有时需要通过填充来避免信息损失，有时也需要通过设置步长（**Stride**）来压缩一部分信息。

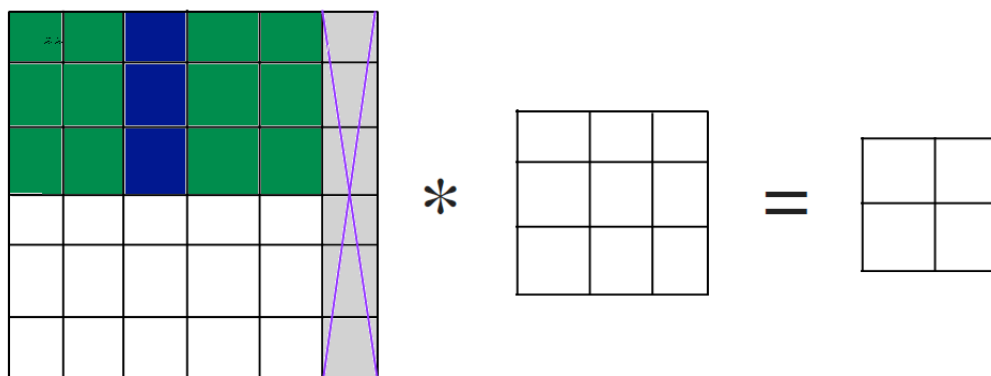
步长表示滤波器在原始图片的水平方向和垂直方向上每次移动的距离。之前，步长被默认为 1。而如果我们设置步长为 2，则卷积过程如下图所示：



设输入图片大小为 $n \times n$ ，滤波器大小为 $f \times f$ ，填充长度为 p ，步长为 s ，则卷积后的图片大小为：

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

Note: 公式中有向下取整运算符，当原始图片中有不能完全被滤波器覆盖的部分，则舍弃：



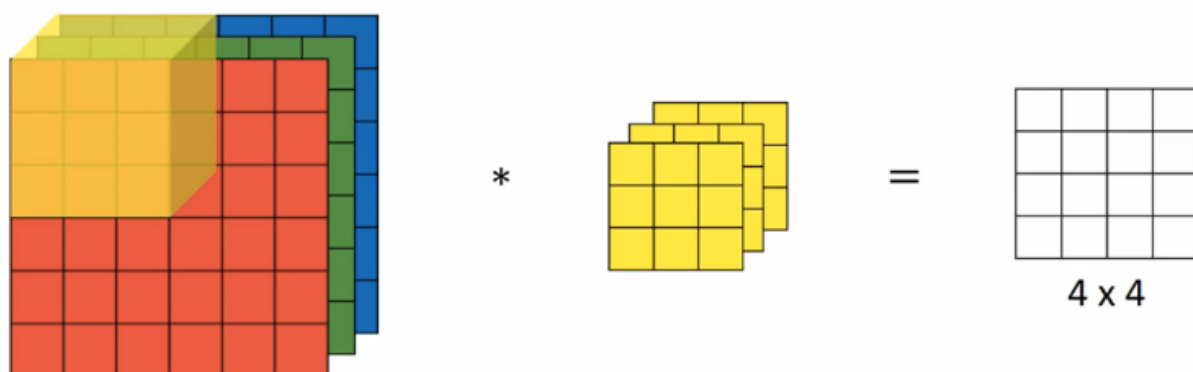
stride=2 padding=0

目前为止我们学习的“卷积”实际上被称为**互相关（cross-correlation）**，而非数学意义上的卷积。真正的卷积操作在做元素乘积求和之前，要将滤波器沿水平和垂直轴翻转（相当于旋转 180 度）。因为这种翻转对一般为水平或垂直对称的滤波器影响不大，按照机器学习的惯例，我们通常不进行翻转操作，在简化代码的同时使神经网络能够正常工作。

6. 高维卷积

6.1 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：



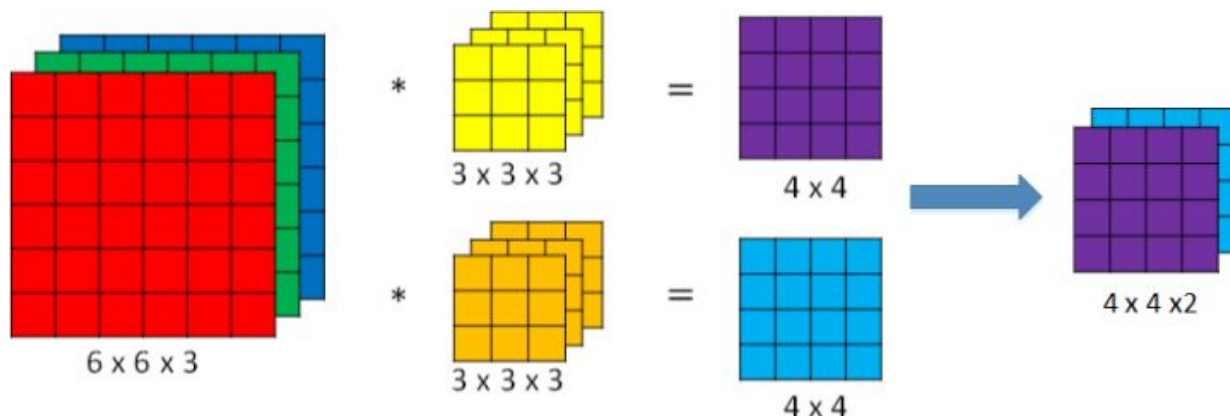
Note: 卷积核的通道数需与输入图像的通道数相等！

Q: 3通道的卷积核，每个通道上的权值是否相等？

不相等，需要学习

6.2 多核卷积

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。

6.3 Summary

设输入图片的尺寸为 $n \times n \times n_c$ (n_c 为通道数)，

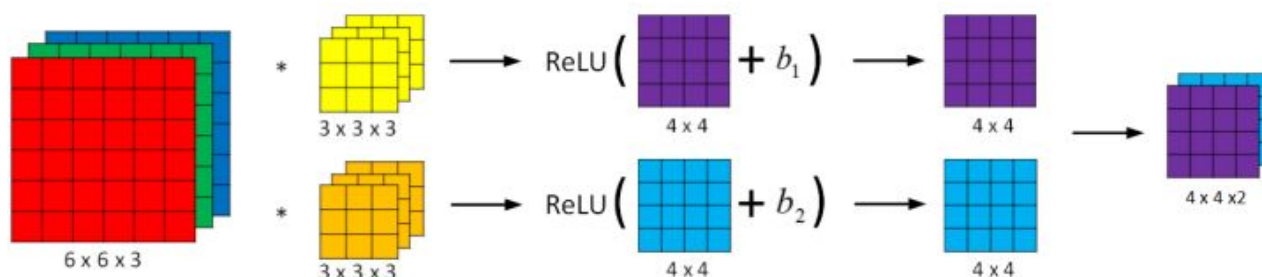
卷积核的尺寸为 $f \times f \times n_c$ ，

则输出图片的尺寸为： $(n - f + 1) \times (n - f + 1) \times n'_c$ (n'_c 为卷积核的个数)

7. 单层卷积网络

7.1 卷积层计算过程

这一小节主要讲如何构建CNN中的卷积层，构建过程如下图所示：



与之前的卷积过程相比，CNN的卷积层多了激活函数和偏置项，其前向传播可表示为：

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b$$

$$A^{[l]} = g^{[l]}(Z^{[l]})$$

其中 $W^{[l]}$ 对应滤波器的数值，所选激活函数是Relu。

Q: WA这里应该是卷积运算，代码如何实现上应不是np.dot?

7.2 卷积层参数

Q: 假设一个CNN的某个卷积层有10个 $3 \times 3 \times 3$ 的卷积核，请问一共有多少个参数？

1个卷积核，包括偏置 b 在内，有28个参数，10个卷积核对应280个参数。

因此，确定卷积核组后，参数的数目与输入图片的大小无关！因此，CNN相较于标准的神经网络，参数要小得多！这也是CNN的一个特征，叫作“避免过拟合”

7.3 符号说明

设第 l 层是一个卷积层：

- $f^{[l]}$ ：滤波器的高（或宽）
- $p^{[l]}$ ：padding
- $s^{[l]}$ ：步长
- $n_c^{[l]}$ ：卷积核的个数

- 输入维度：

$$n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$$

其中， $n_H^{[l-1]}$ 表示输入图片的高， $n_W^{[l-1]}$ 表示输入图片的宽，之前的示例中输入图片的高和宽都相同，但是实际中也可能不同，因此加上标予以区分。

- 输出维度： $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ ，其中

$$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$$n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

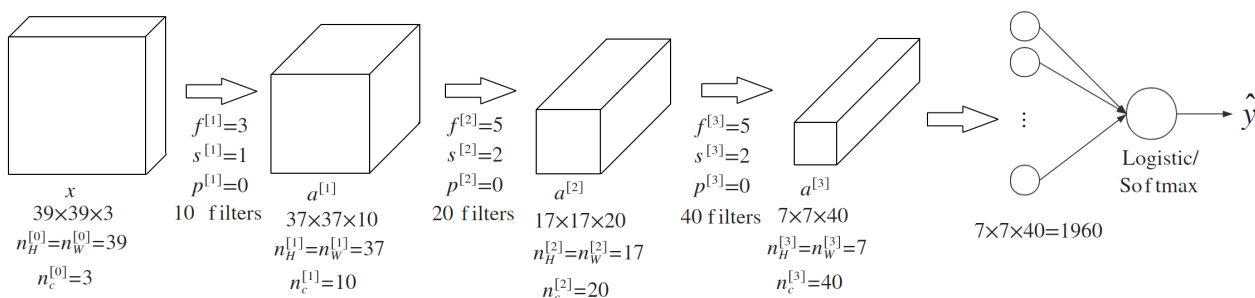
- 每个卷积核组的维度： $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$ ，其中 $n_c^{[l-1]}$ 是输入图片的通道数（或称为深度）
- 权重 $W^{[l]}$ 的维度： $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

- 偏置 b 的维度： $1 \times 1 \times 1 \times n_c^{[l]}$ ，自动广播到输入维度进行计算

Note: 由于深度学习的相关文献并未对卷积标示法达成一致，因此不同的资料关于高度、宽度和通道数的顺序可能不同。有些作者会将通道数放在首位，需要根据标示自行分辨。

8. 简单卷积网络示例

一个简单的 CNN 模型如下图所示：



其中， $a^{[3]}$ 的维度为 $7 \times 7 \times 40$ ，将 1960个特征平滑展开成 1960 个单元的一列，然后连接最后一级的输出层。输出层可以是一个神经元，即二元分类（logistic）；也可以是多个神经元，即多元分类（softmax）。最后得到预测输出 \hat{y} 。

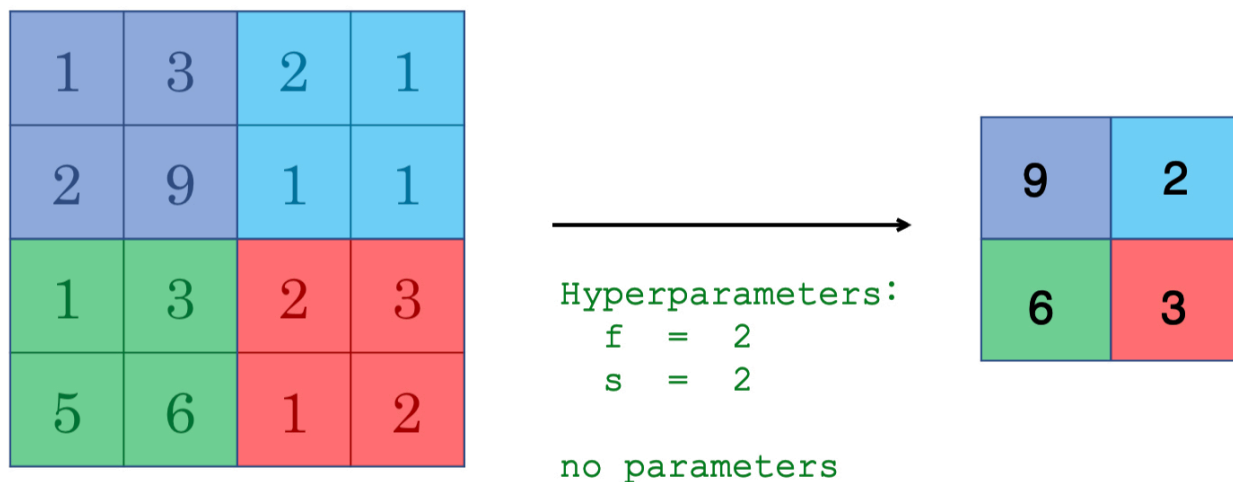
随着神经网络深度的增加，图片的宽和高 $n_H^{[l]}$ 、 $n_W^{[l]}$ 在不断变小，而 $n_c^{[l]}$ 不断变大。

一个典型的卷积神经网络通常包含有三种层：**卷积层（Convolution layer）**、**池化层（Pooling layer）**、**全连接层（Fully Connected layer）**。仅用卷积层也有可能构建出很好的神经网络，但大部分神经网络还是会添加池化层和全连接层，它们更容易设计。

9. 池化层

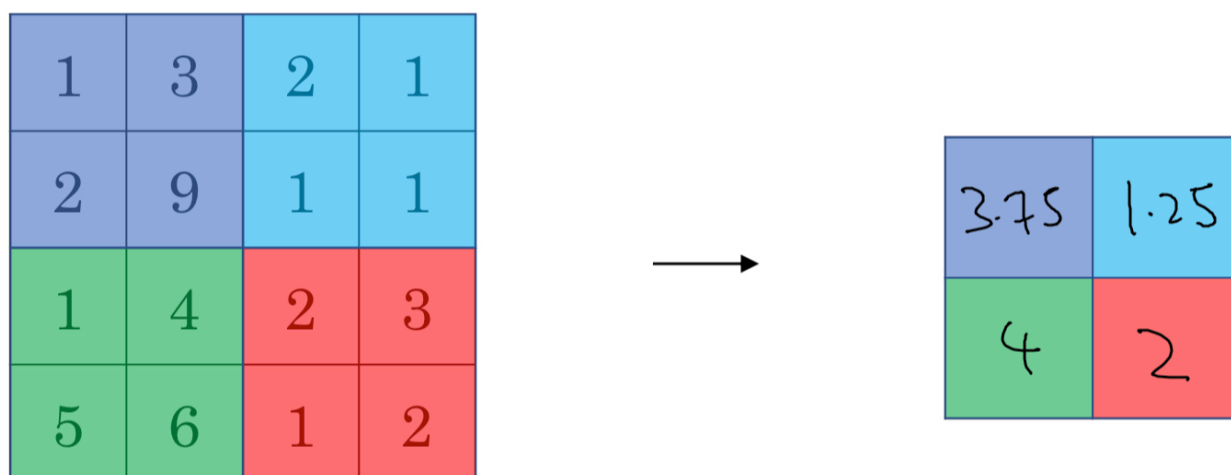
池化层的作用：缩减模型大小，提高计算速度；同时减少噪声，提高模型提取特征的鲁棒性。

9.1 Max Pooling



对最大池化的一种直观解释是，元素值较大可能意味着池化过程之前的卷积过程提取到了某些特定的特征，池化过程中的最大化操作使得只要在一个区域内提取到某个特征，它都会保留在最大池化的输出中。但是，没有足够的证据证明这种直观解释的正确性，而最大池化被使用的主要原因是它在很多实验中的效果都很好。

9.2 Average Pooling



池化过程的特点之一是，它有一组超参数，但是并没有参数需要学习。池化过程的超参数包括滤波器的大小 f 、步长 s ，以及选用最大池化还是平均池化。而填充 p 则很少用到。

池化过程的输入维度为：

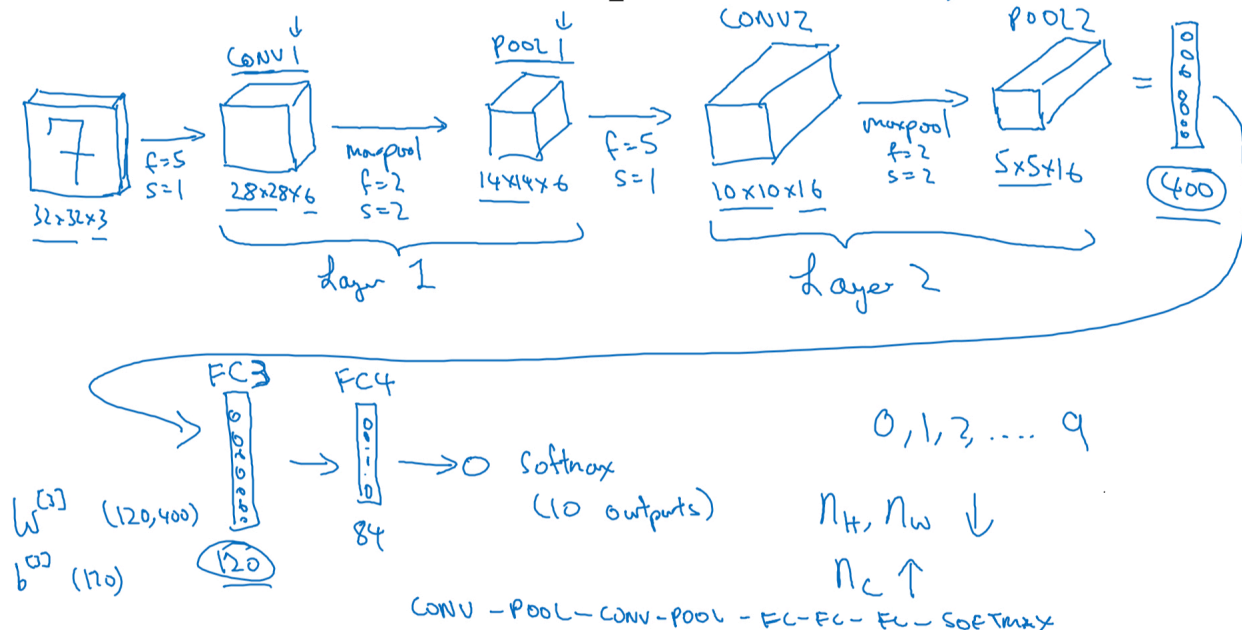
$$n_H \times n_W \times n_c$$

则池化操作输出的维度为：

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_c$$

10. CNN示例

Neural network example (LeNet-5)



在计算神经网络的层数时，通常只统计具有权重和参数的层，因此池化层通常和之前的卷积层共同计为一层。

CNN网络的参数：

	Activation shape	Activation Size	#parameters
Input	(32, 32, 3)	3072	0
CONV1(f=5, s=1)	(28, 28, 6)	4704	456
POOL1	(14, 14, 6)	1176	0
CONV2(f=5, s=1)	(10, 10, 16)	1600	2416
POOL2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48120
FC4	(84, 1)	84	10164
Softmax	(10, 1)	10	850

CONV1层参数个数计算示例：

$$\#params = (f^{[1]} \times f^{[1]} \times n_c^{[0]} + 1) \times n_c^{[1]} = (5 * 5 * 3 + 1) * 6 = 456$$

11. 卷积的作用

相比标准神经网络，对于大量的输入数据，卷积过程有效地减少了CNN的参数数量，原因有以下两点：

- **参数共享**：特征检测如果适用于图片的某个区域，那么它也可能适用于图片的其他区域。即在卷积过程中，不管输入有多大，一个特征探测器（滤波器）就能对整个输入的某一特征进行探测。
- **稀疏连接**：在每一层中，由于滤波器的尺寸限制，输入和输出之间的连接是稀疏的，每个输出值只取决于输入在局部的一小部分值。

池化过程则在卷积后很好地聚合了特征，通过降维来减少运算量。

由于 CNN 参数数量较小，所需的训练样本就相对较少，因此在一定程度上不容易发生过拟合现象。并且 CNN 比较擅长捕捉区域位置偏移。即进行物体检测时，不太受物体在图片中位置的影响，增加检测的准确性和系统的健壮性。