1 卷积神经网络

1.1 卷积基础知识

1.1.1 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机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：

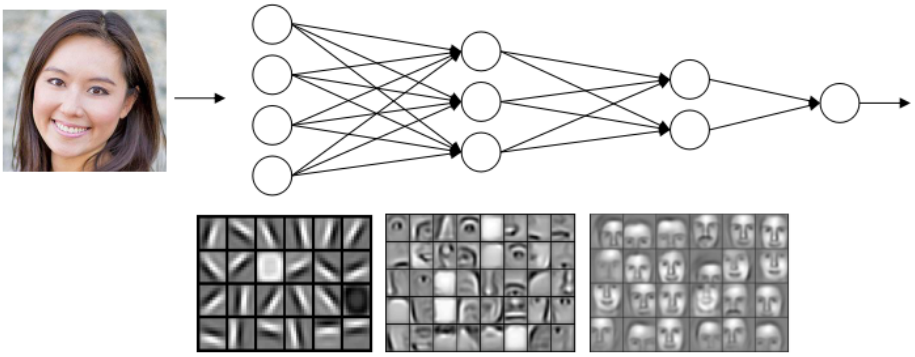
- 图片分类（Image Classification）；
- 目标检测（Object detection）；
- 神经风格转换（Neural Style Transfer）。

应用计算机视觉时要面临的一个挑战是数据的输入可能会非常大。例如一张 1000x1000x3 的图片，神经网络输入层的维度将高达三百万，使得网络权重 W 非常庞大。这样会造成两个后果：

- 神经网络结构复杂，数据量相对较少，容易出现过拟合；
- 所需内存和计算量巨大。

因此，一般的神经网络很难处理蕴含着大量数据的图像。解决这一问题的方法就是使用**卷积神经网络（Convolutional Neural Network, CNN）**。

1.1.2 基本概念



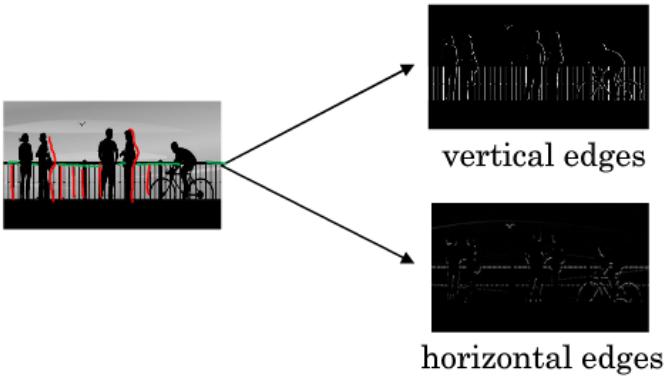
前面在神经网络中提到过，构建一个深度神经网络来进行人脸识别时，深度神经网络的前面一层可以用来进行**边缘探测**，其次一层用来探测照片中组成面部的各个特征部分，到后面的一层就可以根据前面获得的特征识别不同的脸型等等。其中的这些工作，都是依托CNN实现的。

1.1.2.1 卷积运算

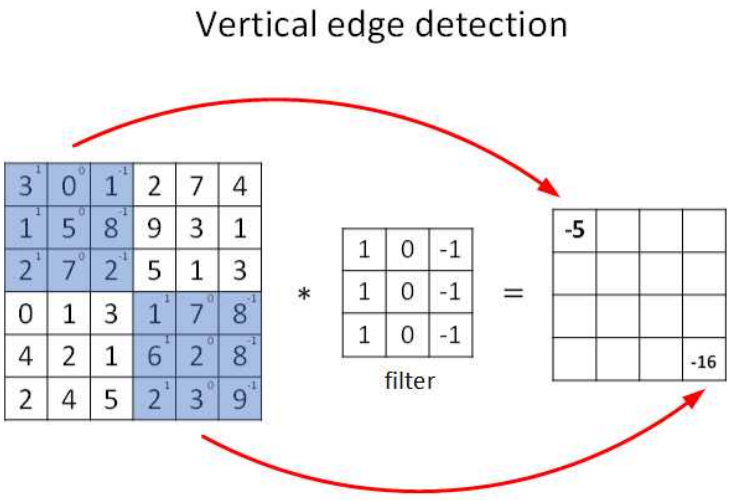
卷积运算（Convolutional Operation）是卷积神经网络最基本的组成部分。我们以边缘检测为例，来解释卷积是怎样运算的。

边缘检测

图片最常做的边缘检测有两类：垂直边缘（Vertical Edges）检测和水平边缘（Horizontal Edges）检测。

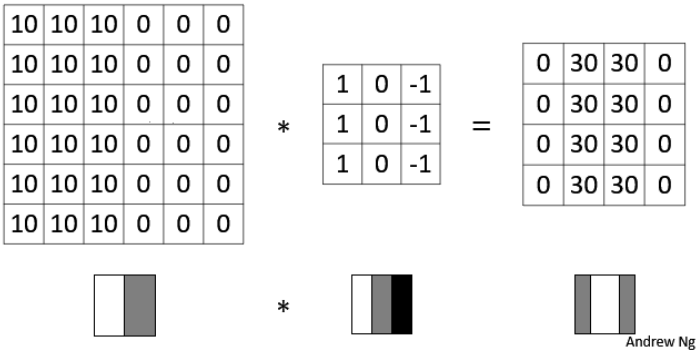


图片的边缘检测可以通过与相应滤波器进行卷积来实现。以垂直边缘检测为例，原始图片尺寸为 6x6，中间的矩阵被称作**滤波器（filter）**，尺寸为 3x3，卷积后得到的图片尺寸为 4x4，得到结果如下（数值表示灰度，以左上角和右下角的值为例）：



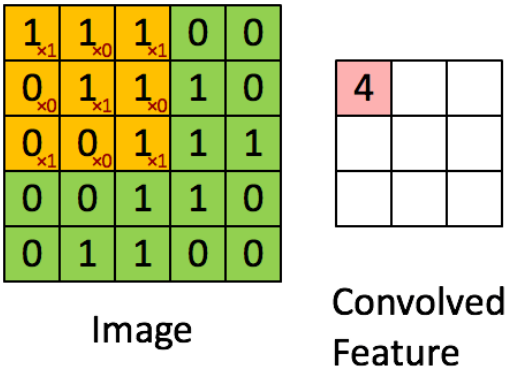
可以看到，卷积运算的求解过程是从左到右，由上到下，每次在原始图片矩阵中取与滤波器同等大小的一部分，每一部分中的值与滤波器中的值对应相乘后求和，将结果组成一个矩阵。

下图对应一个垂直边缘检测的例子：



如果将最右边的矩阵当作图像，那么中间一段亮一些的区域对应最左边的图像中间的垂直边缘。

深度学习里面所称的卷积运算，和泛函分析中的卷积运算有所不同，它的求解过程只是简单将图像矩阵中，从左到右，由上到下，取与滤波器同等大小的一部分，每一部分中的值与滤波器中的值对应相乘后求和，最后的结果组成一个矩阵，其中没有经过翻转、反褶等过程。这里是一个卷积运算的动态的例子，方便理解：

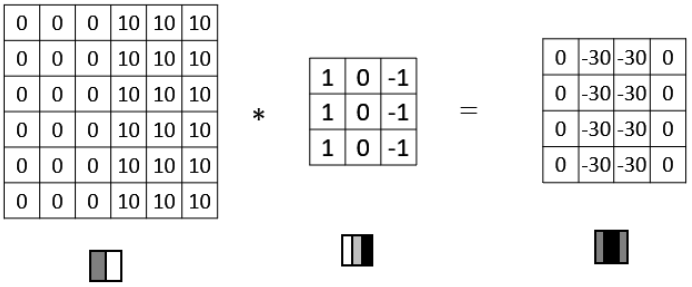


图中的*表示卷积运算符号。在计算机中这个符号表示一般的乘法，而在不同的深度学习框架中，卷积操作的 API 定义可能不同：

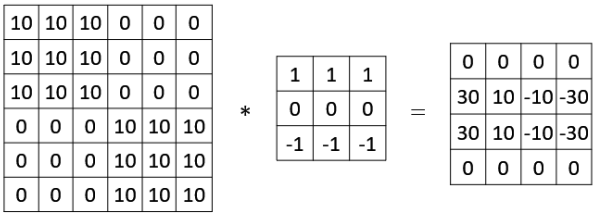
- 在 Python 中，卷积用 conv_forward() 表示；
- 在 Tensorflow 中，卷积用 tf.nn.conv2d() 表示；
- 在 keras 中，卷积用 Conv2D() 表示。

更多边缘检测

如果将灰度图左右的颜色进行翻转，再与之前的滤波器进行卷积，得到的结果也有区别。实际应用中，这反映了**由明变暗和由暗变明**的两种渐变方式。可以对输出图片取绝对值操作，以得到同样的结果。



上面的这个滤波器 $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ 可以用来探测垂直方向的边缘，那么只要把这个滤波器翻转下，变成 $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$，则这个新的滤波器就可以用来探测水平方向的边缘。如下图：



不同的滤波器有着不同的作用。滤波器矩阵的大小和其中的值也都不是固定不变的，可以根据需求来选择。

垂直边缘检测和水平边缘检测的滤波器如下所示：

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。

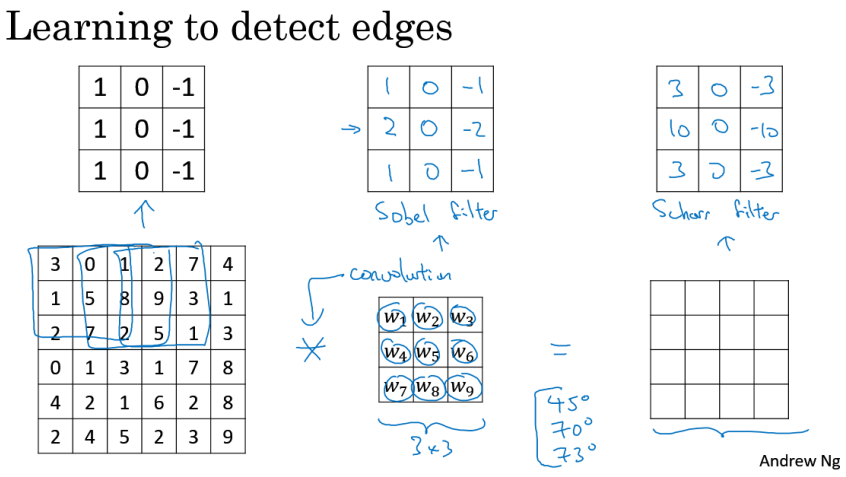
1	0	-1
2	0	-2
1	0	-1

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。



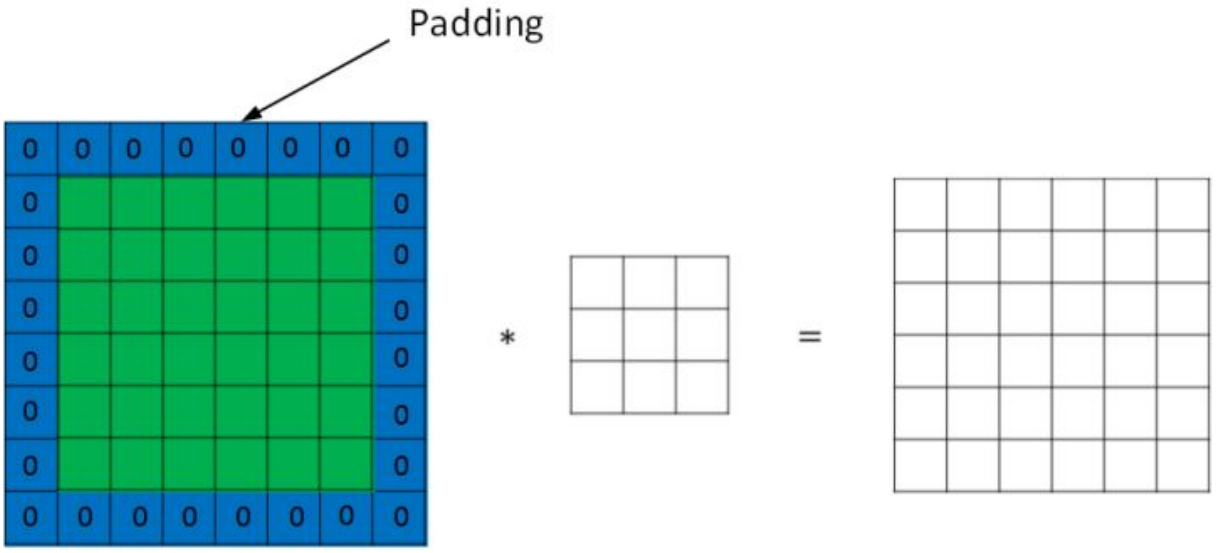
1.1.2.2 填充

假设输入图片的大小为 $n \times n$，而滤波器的大小为 $f \times f$，则卷积后的输出图片大小为 $(n-f+1) \times (n-f+1)$。

这样就有两个问题：

- 每次卷积运算后，输出图片的尺寸缩小；
- 原始图片的角落、边缘区像素点在输出中采用较少，输出图片丢失边缘位置的很多信息。

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 $(n+2p-f+1)\times(n+2p-f+1)$。

因此，在进行卷积运算时，我们有两种选择：

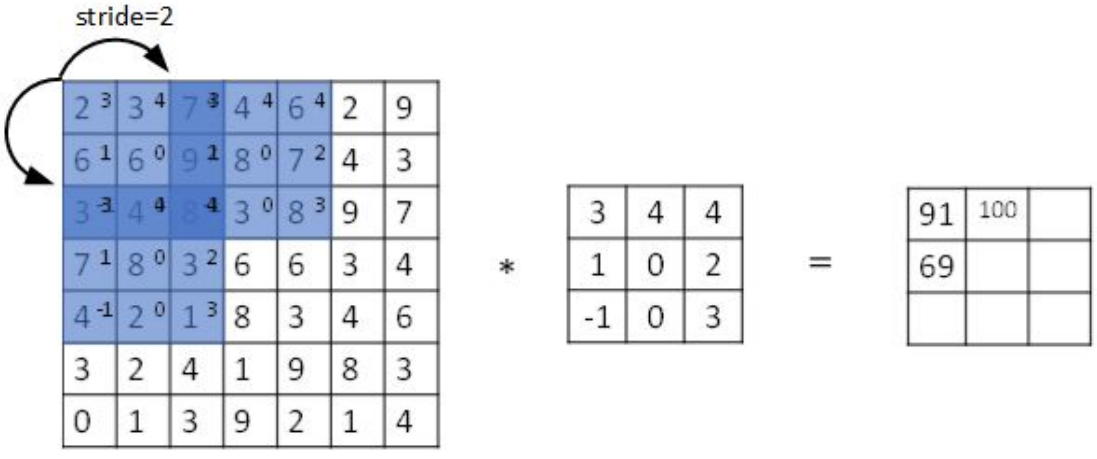
- **Valid 卷积：**不填充，直接卷积。结果大小为 $(n-f+1)\times(n-f+1)$；
- **Same 卷积：**进行填充，并使得卷积后结果大小与输入一致，这样 $p=\frac{f-1}{2}$。

在计算机视觉领域，f通常为奇数。原因包括 Same 卷积中 $p=\frac{f-1}{2}$ 能得到自然数结果，并且滤波器有一个便于表示其所在位置的中心点。

1.1.2.3 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：



设步长为 s，填充长度为 p，输入图片大小为 $n\times n$，滤波器大小为 $f\times f$，则卷积后图片的尺寸为：

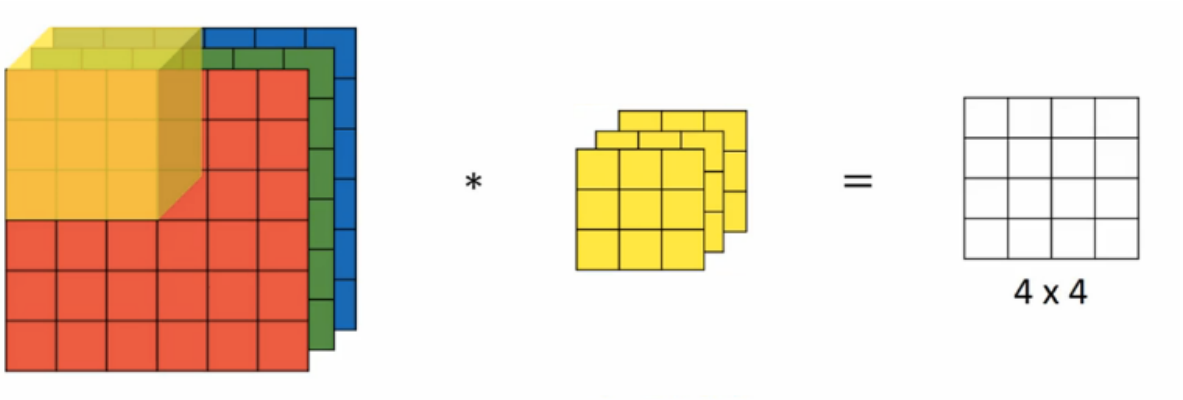
$$\left\lfloor\frac{n+2p-f}{s}+1\right\rfloor\times\left\lfloor\frac{n+2p-f}{s}+1\right\rfloor \tag{1}$$

注意公式中有一个向下取整的符号，用于处理商不为整数的情况。向下取整反映着当取原始矩阵的图示蓝框完全包括在图像内部时，才对它进行运算。

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。

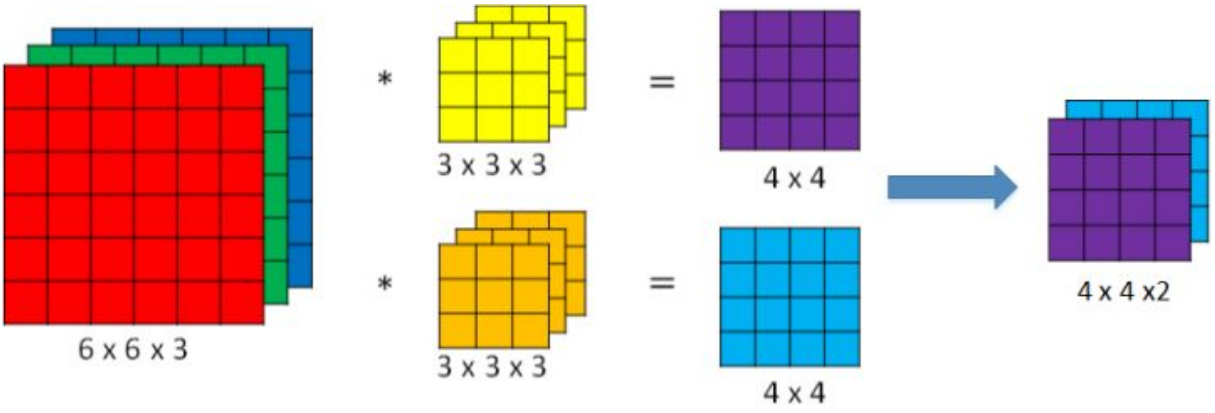
1.1.3 高维卷积

如果我们想要对三通道的 RGB 图片进行卷积运算，那么其对应的滤波器组也同样是三通道的。过程是将每个单通道（R，G，B）与对应的滤波器进行卷积运算求和，然后再将三个通道的和相加，将 27 个乘积的和作为输出图片的一个像素值。

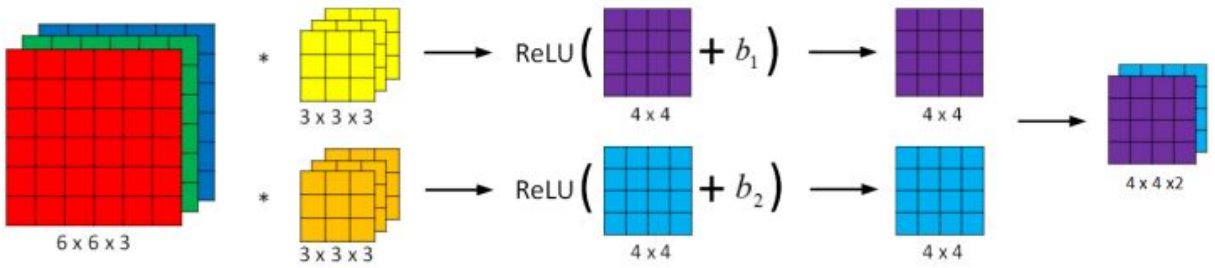


不同通道的滤波器可以不相同。例如只检测 R 通道的垂直边缘，G 通道和 B 通道不进行边缘检测，则 G 通道和 B 通道的滤波器全部置零。当输入有特定的高、宽和通道数时，滤波器可以有不同的高和宽，但通道数必须和输入一致。

如果想同时检测垂直和水平边缘，或者更多的边缘检测，可以增加更多的滤波器组。例如设置第一个滤波器组实现垂直边缘检测，第二个滤波器组实现水平边缘检测。设输入图片的尺寸为 $n \times n \times n_c$（n_c 为通道数），滤波器尺寸为 $f \times f \times n_c$，则卷积后的输出图片尺寸为 $(n-f+1) \times (n-f+1) \times n'_c$，$n'_c$ 为滤波器组的个数。



1.1.4 单层卷积网络



与之前的卷积过程相比较，卷积神经网络的单层结构多了激活函数和偏移量；而与标准神经网络：

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b$$
$$A^{[l]} = g^{[l]}(Z^{[l]}) \tag{2}$$

相比，滤波器的数值对应着权重 $W^{[l]}$，卷积运算对应着 $W^{[l]}$ 与 $A^{[l-1]}$ 的乘积运算，所选的激活函数变为 ReLU。

对于一个 3x3x3 的滤波器，包括偏移量 b 在内共有 28 个参数。不论输入的图片有多大，用这一个滤波器来提取特征时，参数始终都是 28 个，固定不变。即**选定滤波器组后，参数的数目与输入图片的尺寸无关**。因此，卷积神经网络的参数相较于标准神经网络来说要少得多。这是 CNN 的优点之一。

1.1.5 符号约定

设 l 层为卷积层：

- $f^{[l]}$：滤波器的高（或宽）
- $p^{[l]}$：填充长度
- $s^{[l]}$：步长
- $n_c^{[l]}$：滤波器组的数量
- 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，因此加上下标予以区分。
- 输出维度：$n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$。其中

$$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$
$$n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \tag{3}$$

- 每个滤波器组的维度：$f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$。其中 $n_c^{[l-1]}$ 为输入图片通道数（也称深度）。
- 权重维度：$f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
- 偏置维度：$1 \times 1 \times 1 \times n_c^{[l]}$

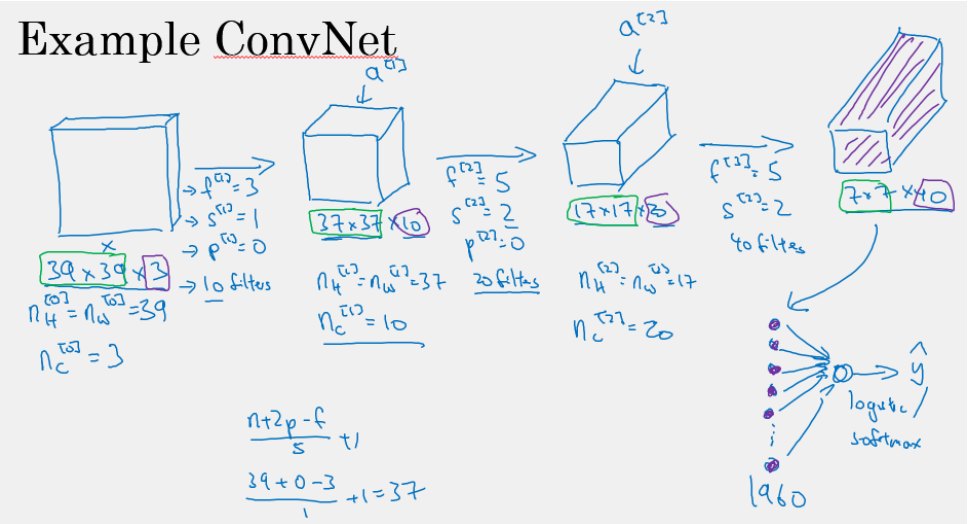
由于深度学习的相关文献并未对卷积标示法达成一致，因此不同的资料关于高度、宽度和通道数的顺序可能不同。有些作者会将通道数放在首位，需要根据标示自行分辨。

1.1.6 网络的结构

通常一个卷积神经网络是由**卷积层（Convolution）**、**池化层（Pooling）**、**全连接层（Fully Connected）**组成。

在输入层输入原始数据，卷积层中进行的是前面所述的卷积过程，用它来进行提取特征。全连接层就是将识别到的所有特征全部连接起来，并输出到分类器（如Softmax）。

一个简单的 CNN 如下所示：



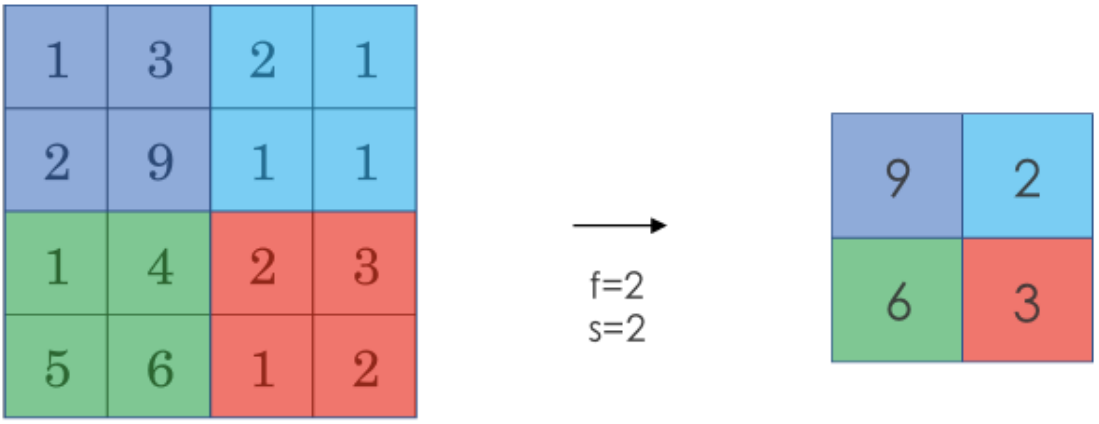
其中，$a^{[3]}$ 的维度为 7x7x40，将 1960 个特征平滑展开成 1960 个单元的一列，然后连接最后一级的输出层。输出层可以是一个神经元，即二元分类（logistic）；也可以是多个神经元，即多元分类（softmax）。最后得到预测输出 $\hat{y}$。

随着神经网络计算深度不断加深，图片的高度和宽度 $n_H^{[l]}$、$n_W^{[l]}$ 一般逐渐减小，而 $n_c^{[l]}$ 在增加

1.1.6.1 池化层

池化层的作用是缩减模型的大小，提高计算速度，同时减小噪声提高所提取特征的稳健性。

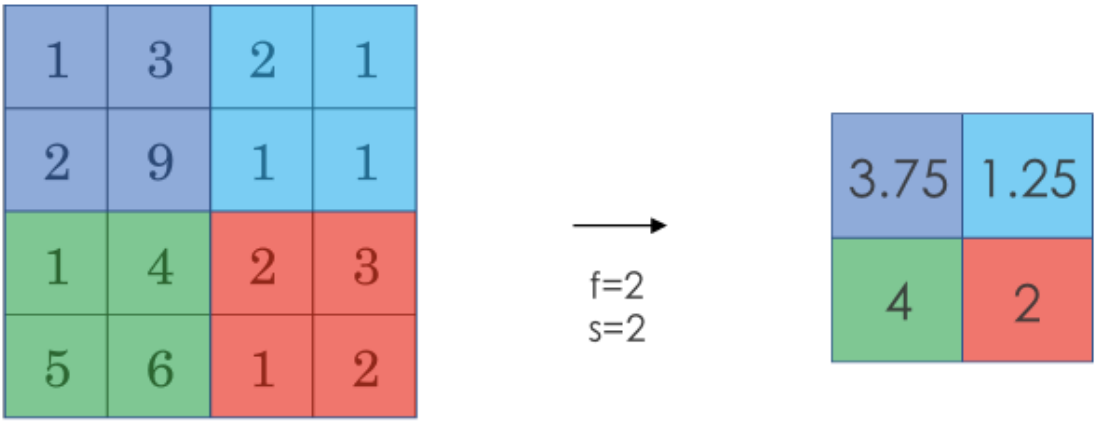
采用较多的一种池化过程叫**最大池化（Max Pooling）**，将输入拆分成不同的区域，输出的每个元素都是对应区域中元素的最大值，如下图所示：



池化过程类似于卷积过程，上图所示的池化过程中相当于使用了一个大小 $f=2$ 的滤波器，且池化步长 $s=2$。卷积过程中的几个计算大小的公式也都适用于池化过程。如果有多个通道，那么就对每个通道分别执行计算过程。

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。

另一种池化过程是**平均池化（Average Pooling）**，就是从取某个区域的最大值改为求这个区域的平均值：



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：

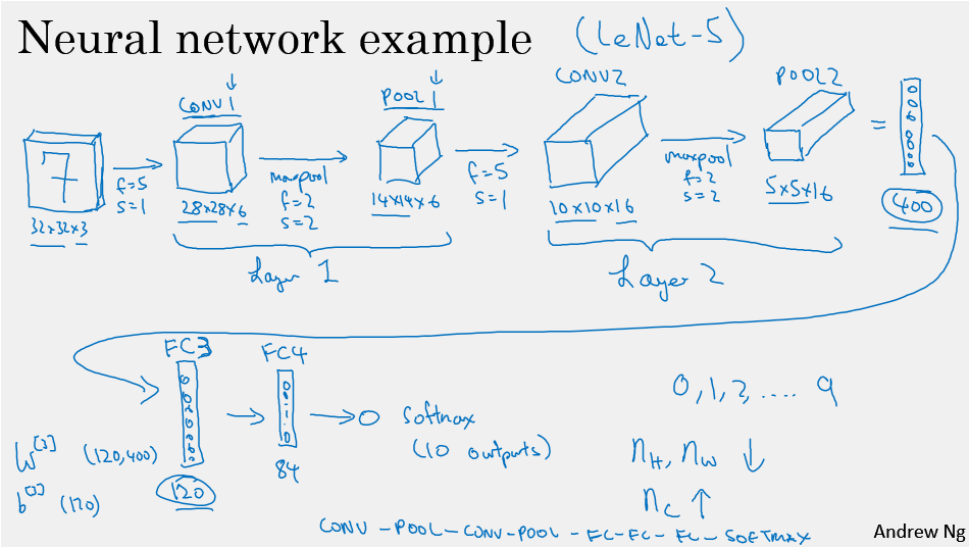
$$n_H \times n_W \times n_c \tag{4}$$

输出维度为：

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_c \tag{5}$$

1.1.6.2 示例结构

一种典型卷积网络结构是 LeNet-5，它是由用 LeCun 在上世纪九十年代提出的用来识别数字的卷积网络，下图的卷积网络结构与它类似：



其中，一个卷积层和一个池化层组成整个卷积神经网络中的一层，图中所示整个过程为：
$Input->Conv1->Pool1->Conv2->Pool2->FC->FC3->FC4->Softmax$

下面各层中的数据：

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	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

可以看出，激活的大小随着向卷积网络深层的递进而减小，参数的数量在不断增加后在几个全连接过程后将逐渐减少。

1.1.7 卷积使用解释

对于大量的输入数据，卷积过程有效地减少了参数数量，而这主要归功于以下两点：

- **参数共享(Parameter Sharing)**：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，一个特征探测器也就前面所说的滤波器就能对整个输入的特征进行探测。
- **局部感知(Local Perception)**：在每一层中，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，通过降维而减少了运算量。

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。

整合训练这些层