Convolutional Neural Network

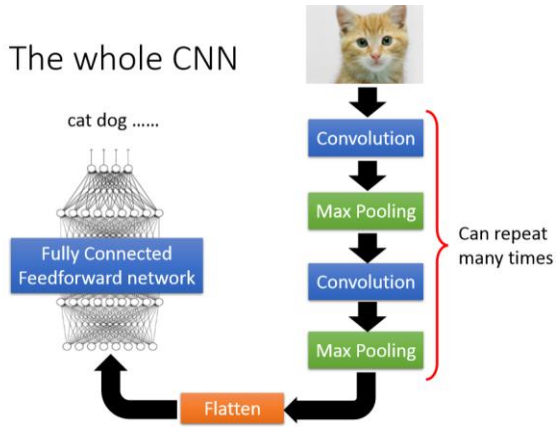
Why CNN for Image?

CNN vs DNN

我们当然可以用一般 neural network 来做影像处理，不一定要用 CNN，比如说，你想要做图像的分类，那你就去 train 一个 neural network，它的 input 是一张图片，你就用里面的 pixel 来表示这张图片，也就是一个很长的 vector，而 output 则是由图像类别组成的 vector，假设你有 1000 个类别，那 output 就有 1000 个 dimension

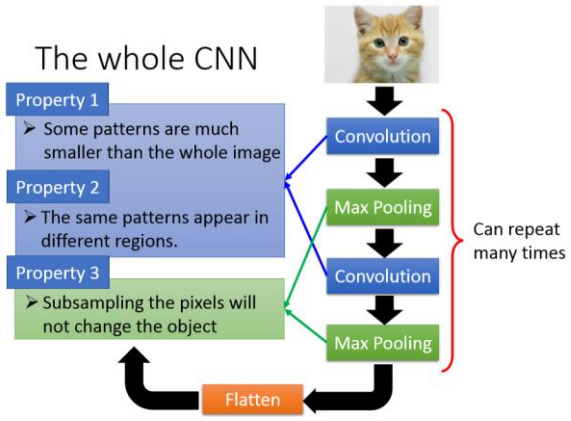
The whole CNN structures

首先，input 一张 image 以后，它会先通过 convolution 的 layer，接下来做 Max pooling 这件事，然后再去做 Convolution，再做 Max pooling…..,这个 process 可以反复进行多次(重复次数需要事先决定)，这就是 network 的架构，就好像 network 有几层一样，你要做几次 convolution，做几次 Max pooling，再定这个 network 的架构就要事先决定好



我们基于之前提到的三个对图像处理的观察，设计了 CNN 这样的架构，第一个是要侦测一个 pattern，你不需要看整张 image，只要看 image 的一个小部分，第二个是同样的 pattern 会出现在一张图片的不同区域；第三个是我们可以对整张 image 做 subsampling

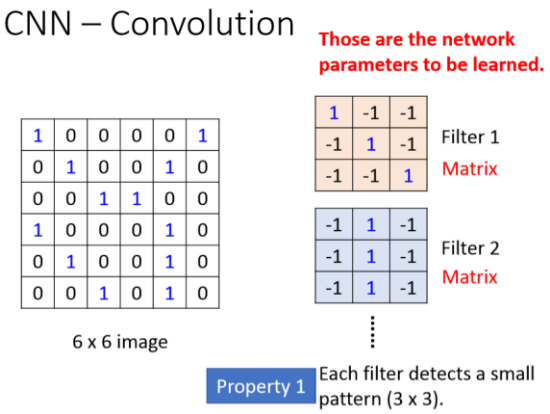
那前面这两个 property，是用 convolution 的 layer 来处理的；最后这个 property，是 max pooling 来处理的



Convolution

假设现在我们 network 的 input 是一张 6*6 的 image，图像是黑白的，因此每个 pixel 只需要用一个 value 来表示，而在 convolution layer 里面，有许多 Filter，这边的每一个 Filter，其实就等于是 Fully connected layer 里的一个 neuron

Property 1

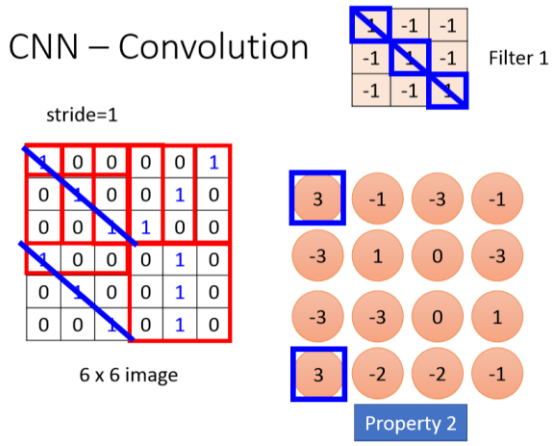


每一个 Filter 其实就是一个 matrix，这个 matrix 里面每一个 element 的值，就跟那边 neuron 的 weight 和 bias 一样，是 network 的 parameter，它们具体的值都是通过 Training data 学出来的，而不是人去设计的

所以，每个 Filter 里面的值是什么，要做什么事情，都是自动学习出来的，上图中每一个 filter 是 3*3 的 size，意味着它就是在侦测一个 3*3 的 pattern，当它侦测的时候，并不会去看整张 image，它只看一个 3*3 范围内的 pixel，就可以判断某一个 pattern 有没有出现，这就考虑了 property1

Property 2

这个 filter 是从 image 的走上角开始，做一个 slide window，每次向右挪动一定的距离，这个距离就叫做 stride，有你自己设定，每次 filter 停下的时候就跟 image 中对应的 3*3 的 matrix 做一个内积(相同位置的值相乘并累计求和)，这里假设 stride=1，那么我们的 filter 每次移动一格，当它碰到 image 最右边的时候，就从下一行的最左边开始重复进行上述操作，经过一整个 convolution 的额 process，最终得到下图所示的红色的 4*4matrix

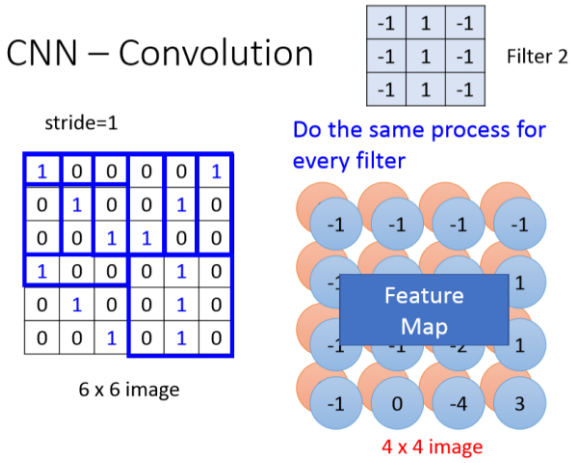


观察上图中的 Filter1，它斜对角的地方是 1,1,1，所以它的工作就是 detect 有没有连续的从左上角到右下角的 1,1,1 出现这个 image 里面，检测到的结果已在上图中用蓝线标识出来，此时 filter 得到的卷积结果的左上角和走下得到了最大的值，这就是说，该 filter 所要侦测的 pattern 出现在 image 的左上角和左下角

同一个 pattern 出现在 image 左上角的位置和左下角的位置，并不需要用到不同的 filter，我们用 filter1 就可以侦测出来，这就考虑了 property2

Feature Map

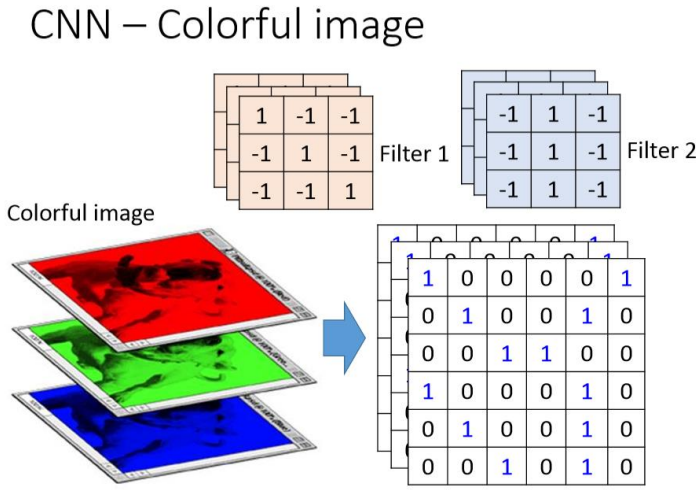
在一个 convolution 的 layer 里面，有许多的 filter，不一样的 filter 会有不一样的参数，但是这些 filter 做卷积的过程是一模一样的，你把 filter2 做完 convolution 以后，你就会得到另外一个蓝色的 4*4matrix，那这个蓝色的 4*4matrix 与之前红色的 4*4matrix 合起来。就叫做 Feature Map(特征映射)，有多少个 filter，对应就有多少个映射后的 image



CNN 对不同 scale 的相同 pattern 的处理上存在一定的困难，由于现在每一个 filter size 都是一样的，这意味着，如果你今天有同一个 pattern，它有不同的 size，有大的鸟嘴，也有小的鸟嘴，CNN 并不能够自动处理这个问题；DeepMind 曾经发过一篇 paper，上面提到了当你 input 一张 image 的时候，它在 CNN 前面，再接另一个 network，这个 network 做的事情是，它会 output 一些 scalar，告诉你说，它要把这个 image 的里面的哪些位置做旋转、缩放、然后，在丢到 CNN 里面，这样你其实会得到比较好的 performance

Colorful image

刚才举的例子是黑白的 image，所以你 input 的是一个 matrix，如果今天是彩色的 image 会怎么样呢？我们知道彩色的 image 就是由 RGB 组成的，所以一个彩色的 image，它就是好几个 matrix 叠在一起，是一个立方体，如果我今天要处理彩色的 image，要怎么做呢?



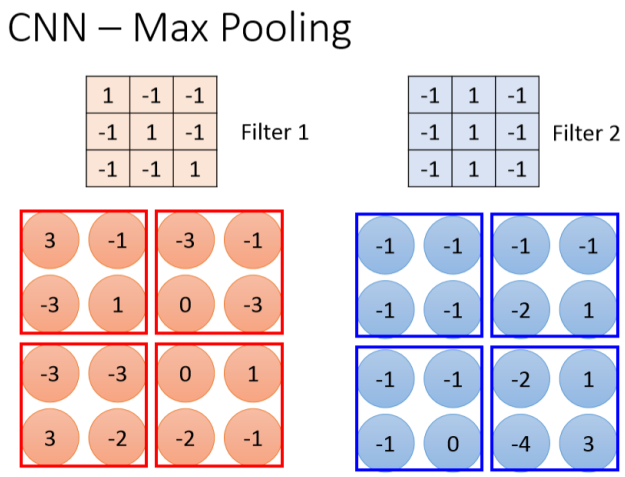
这个时候你的 filter 就不再是一个 matrix 了，它也会是一个立方体，如果你今天是 RGB 这三个颜色来表示一个 pixel 的话，那你 input 就是 3*6*6，你的 filter 就是 3*3*3，你的 filter 的高就是 3，你在做 convolution 的时候，就是把这个 filter 的 9 个值跟这个 image 里面的 9 个值做内积，可以想象成 filter 的每一层都分别跟 image 的三层做内积，得到的也是一个三层的 output，每一个 filter 同时就考虑了不同颜色所代表的 channel

Max Pooling

Operation of max pooling

相较于 convolution，max pooling 是比较简单的。它就是做 subsampling，根据 filter1，我们得到一个 4*4 的 matrix，根据 filter2，你得到另外一个 4*4 的 matrix，接下来，我们要做什么事呢?

我们把 output 四个分为一组，每一组里面通过选取平均值或最大值的而方式，把原来 4 个 value 合成一个 value，这件事情相当于在 image 每组相邻的四开区域内都挑出一块来检测，这种 subsampling 的方式就可以让你的 image 缩小



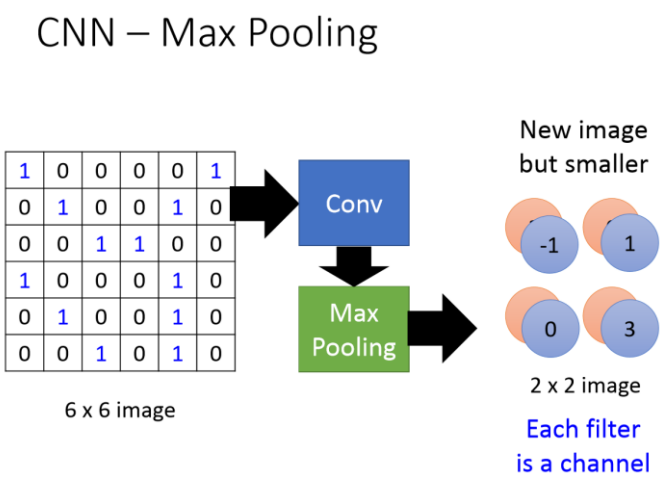
讲到这里你可能会有一个问题，如果取 Maximum 放到 network 里面，不就没法微分了，max 这个东西，感觉是没有办法对它微分的啊，其实是可以的，后面的章节会讲到 Maxout network，会告诉你怎么用微分的方式来处理它

Convolution + Max Pooling

所以，结论是这样的：

做完一次 convolution 加一次 max pooling，我们就把原来 6*6 的 image，变成了一个 2*2 的 image；至于这个 2*2 的 image，它每一个 pixel 的深度，也就是每一个 pixel 用几个 value 来表示，就取决于你有几个 filter，如果你有 50 个 filter，就是 50 维，像下图中是两个 filter，对应的深度就是两维

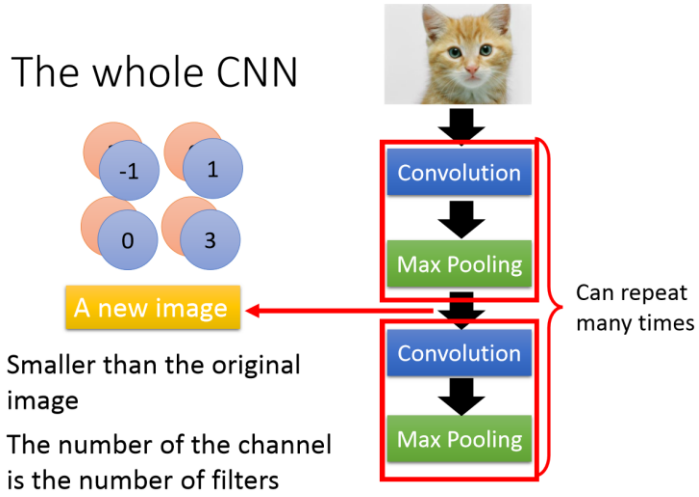
所以，这就是一个新的比较小的 image，它表示的是不同区域上提取到的特征，实际上不同的 filter 检测的是该 image 同一区域上的不同特征属性，所以每一层 channel(通道)代表的是一种属性，一块区域有几种不同的属性，就有几层不同的 channel，对应的就会有几个不同的 filter 对其进行 convolution 操作



这件事情可以 repeat 很多次，你就可以把得到的这个比较小的 image，再次进行 convolution 和 max pooling 的操作，得到一个更小的 image，依次类推

有这样一个问题：假设我第一个 convolution 有 25 个 filter，通过这些 filter 得到 25 个 feature map，然后 repeat 的时候第二个 convolution 也有 25 个 filter，那这样做完，我是不是会得到 25^2 个 feature map?

其实不是这样的，你这边做完一次 convolution，得到 25 个 feature map 之后再做一次 convolution，还是会得到 25 个 feature map，因为 convolution 在考虑 input 的时候，是会考虑深度的，它并不是每一个 channel 分开考虑，而是一次考虑所有的 channel，所以，你 convolution 这边有多少个 filter，再次 output 的时候就会有多少个 channel，因此你这边有 25 个 channel，经过含有 25 个 filter 的 convolution 之后 output 还会是 25 个 channel，只是这边的每一个 channel，它都是一个 cubic(立方体)，它的高有 25 个 value 那么高



Flatten

做完 convolution 和 max pooling 之后，就是 flatten 和 fully connected Feedforward network 的部分 flatten 的意思是，把左边的 feature map 拉直，然后把它丢进一个 fully connected Feedforward network，然后就结束了，也就是说，我们之前通过 CNN 提取出了 image 的 feature，它相较于原先一整个 image 的 vector，少了很大一部分内容，因此需要参数大幅度地减少了，但最终，也还是要丢到一个 fully connected 的 network 中去做最后的分类工作