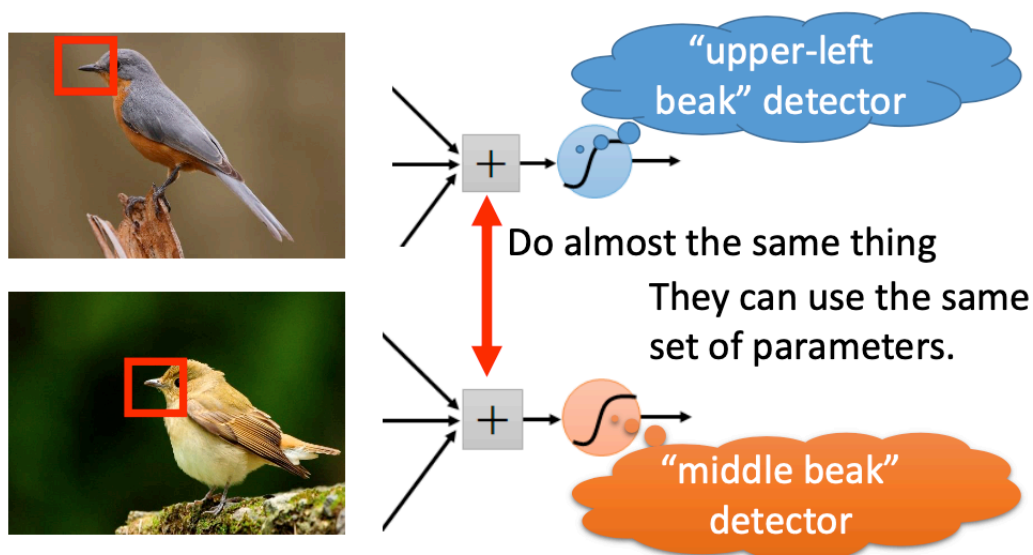


Why CNN for Image

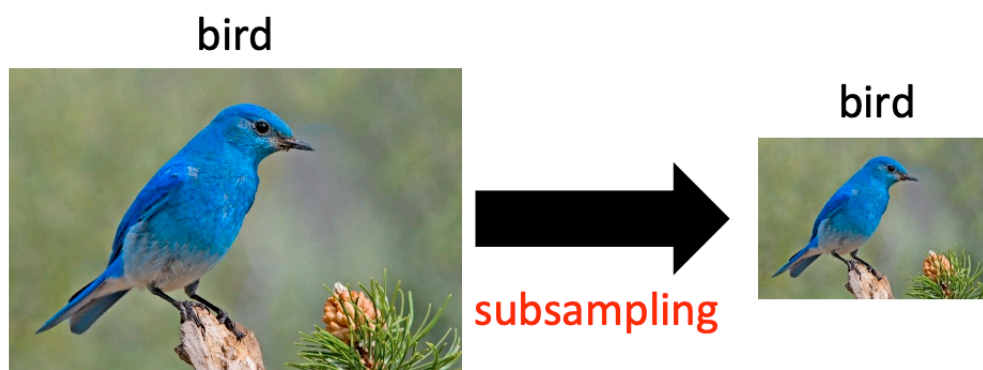
可以使用相同的参数来提取不同图像中的同一种特征，这样就可以降低网络需要学习的参数

- The same patterns appear in different regions.



CNN还有另外一种操作可以减少网络需要训练的参数，即下采样（subsampling），subsampling可以减少图像的大小。采样层就是使用pooling的技术来实现的，可以用max pooling或average pooling，获取某个像素点及其周围区域的最大值或平均值，将这些像素都用一个像素来表示，就可以缩小图像的大小。

- Subsampling the pixels will not change the object

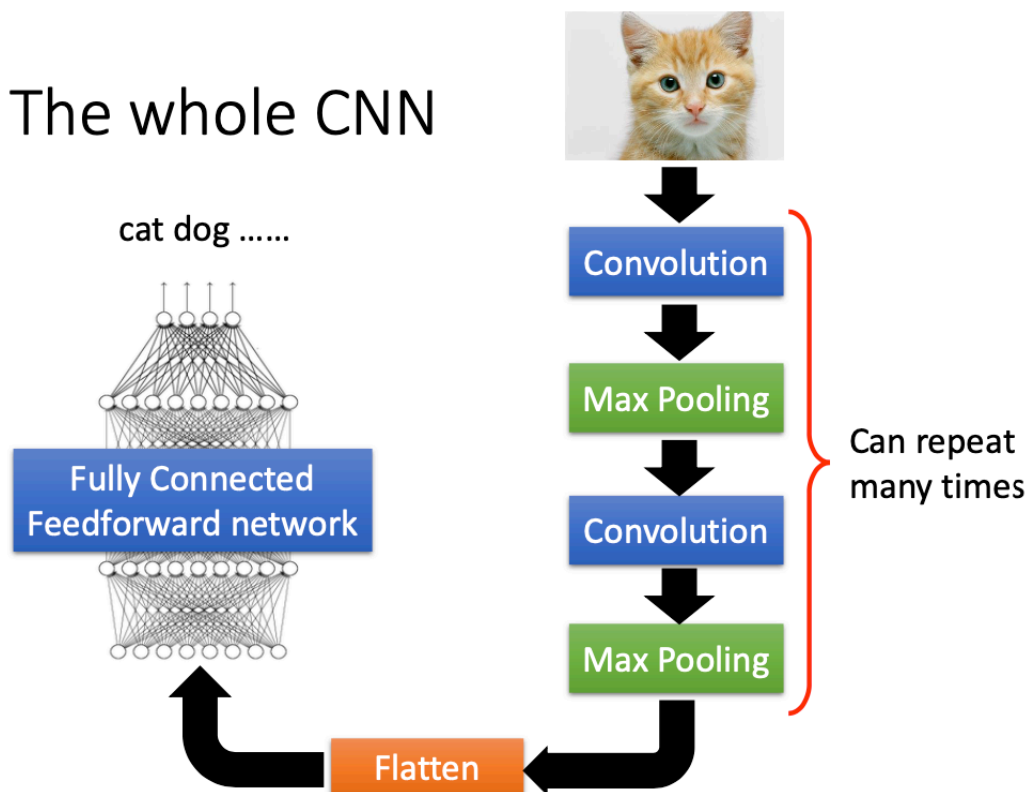


We can subsample the pixels to make image smaller

➡ Less parameters for the network to process the image

一个完整的CNN网络结构，由多个convolution和pooling层、以及全连接层组成。输入图像先经过多次的convolution、pooling，提取图像中的特征，再把这些特征flatten成一个一维的向量，即全连接层，最后再得出分类结果 cat or dog

The whole CNN



convolution

对于我们想要提取图像中的两个特征，我们使用filter1和filter2这两个过滤器，

CNN – Convolution

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1
Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2
Matrix

⋮

Property 1

Each filter detects a small pattern (3 x 3).

CNN – Convolution

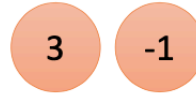
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



CNN – Convolution

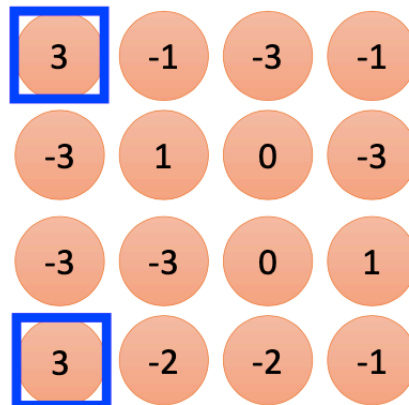
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

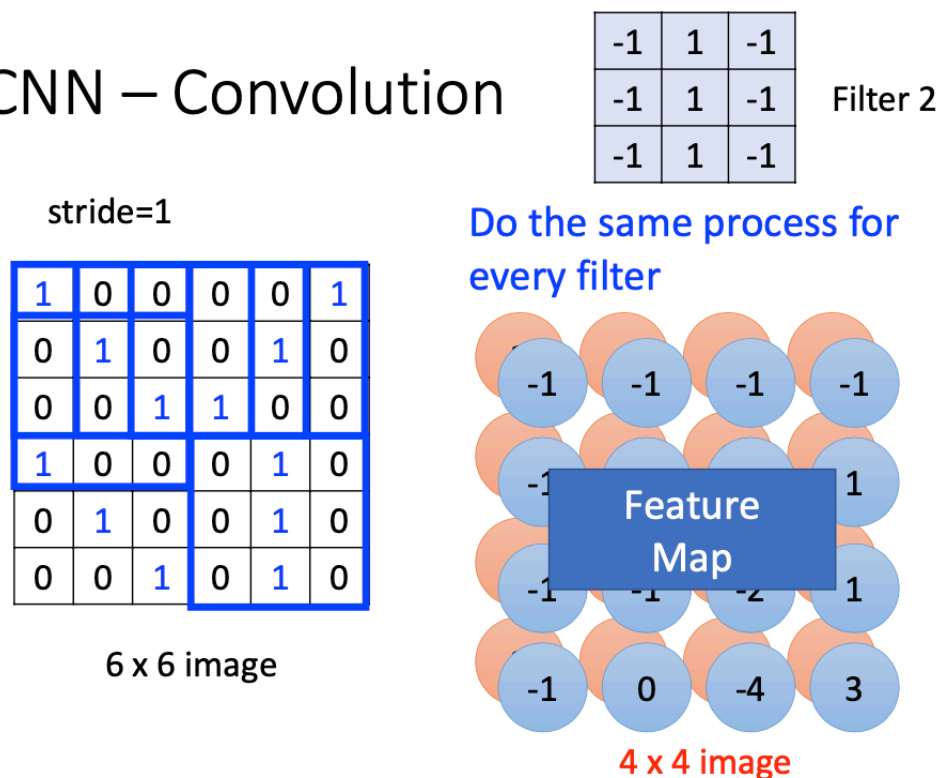
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



Property 2

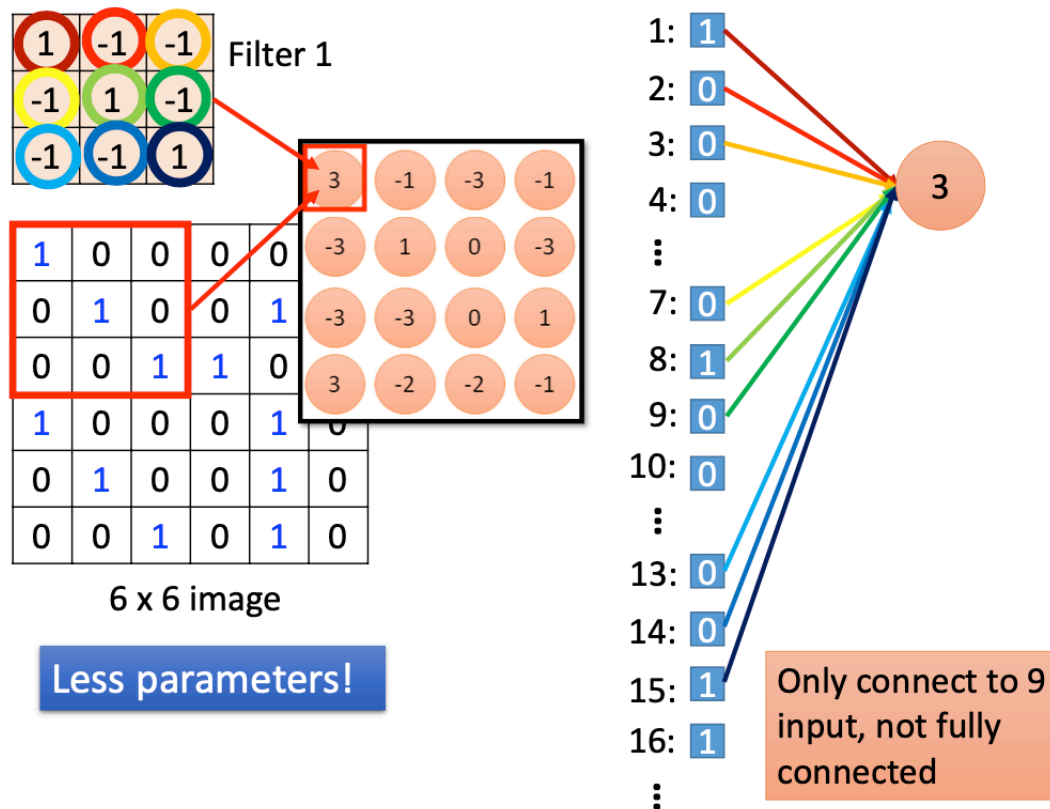
CNN – Convolution



特性

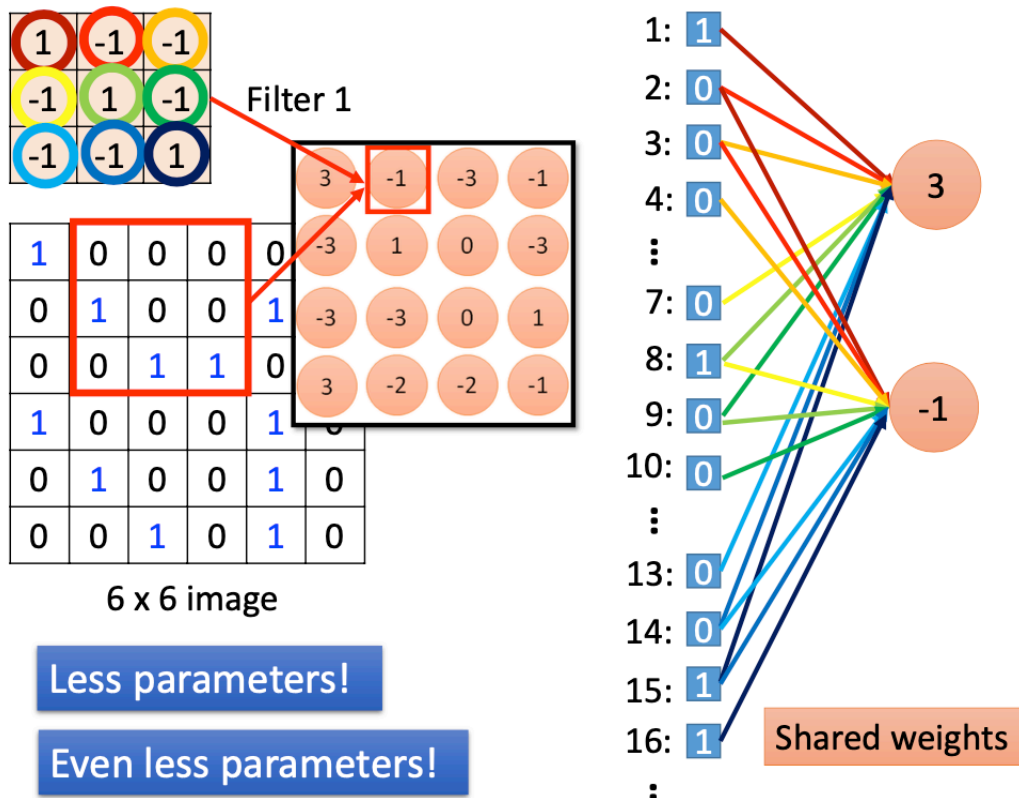
local field

下一层的neural 3, 只与前层的9个neural相连接, 而不是和前层的全部neural连接



weight sharing

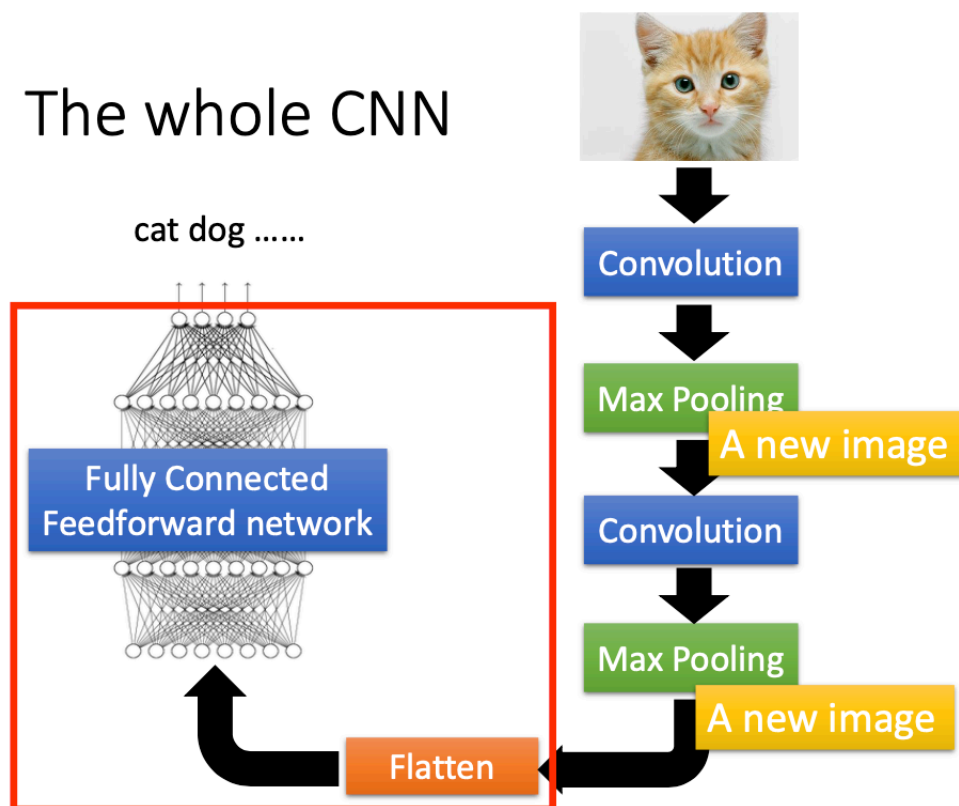
对于图像的同一特征, 只需要一个过滤器filter即可, 这样就可以进一步减少图像中的参数



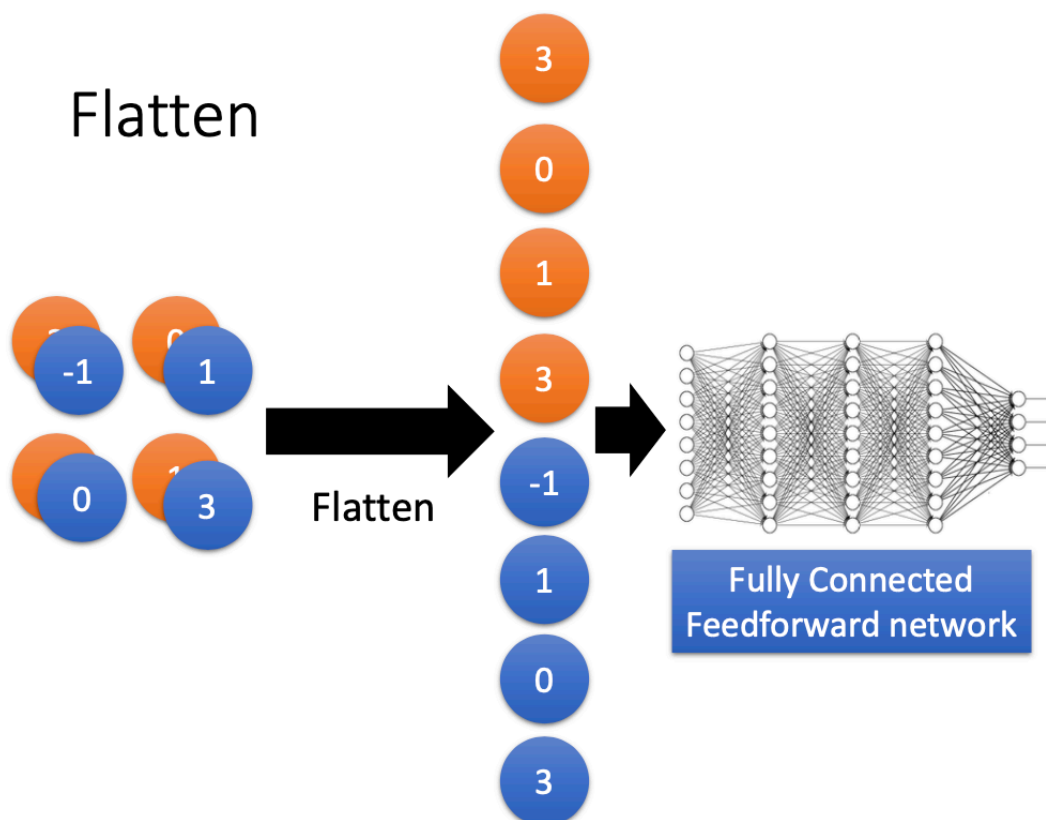
Flatten

每经过一次convolution和pooling操作，都会生成一张新的图片，这张新图片的大小比原图要小很多，减少了需要训练的参数

The whole CNN



再把经过多次convolution和pooling的图像flatten，展开成一维的向量，再输入全连接层



CNN in Keras

首先先介绍一下convolution和pooling对图像大小变化的公式，设输入图像的宽度和高度分别为w和h，卷积核大小为 $F \times F$ ，步长stride大小为S，如果再加入Padding操作，经过卷积或池化操作后图像的大小为W和H

$$H/W = \frac{(h/w - F + 2P)}{S} + 1$$

这里引入了keras，先介绍几个主要函数的参数。

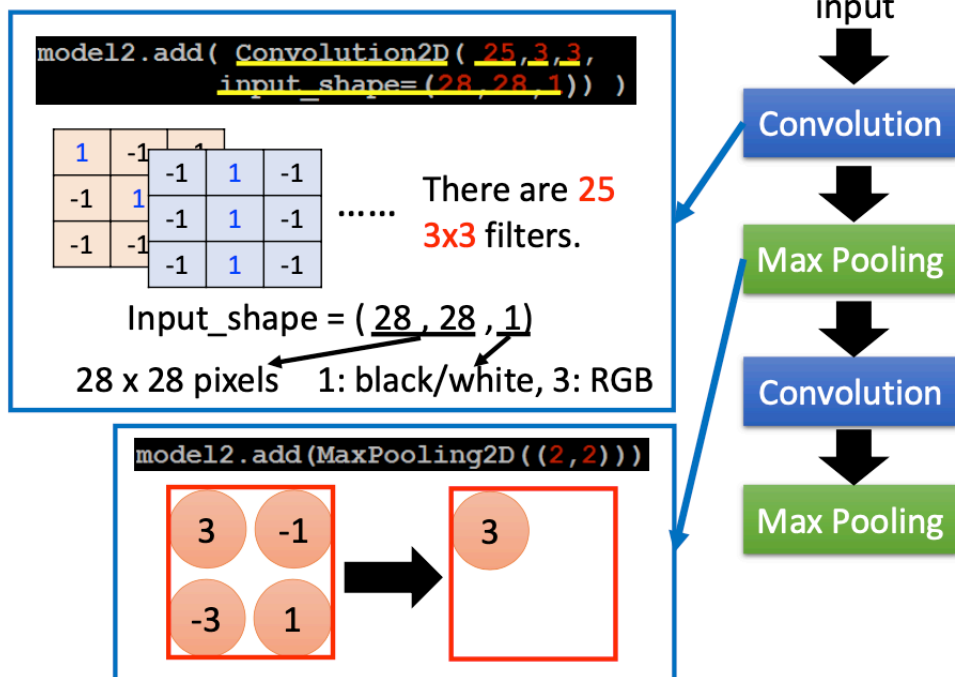
我们可以先通过以下函数来生成一个convolution layer

```
model2.add(Convolution2D(25,3,3, input_shape=(28,28,1)))
```

其中25表示filter的数量， 3×3 表示filter的大小，`model2.add()` 其中的 `Convolution2D` 还有一个参数 `input_shape`，表示函数输入的图像大小，例子中表示一个单通道的 28×28 图像，如果需要输入彩色图像，则是 $28 \times 28 \times 3$

CNN in Keras

Only modified the *network structure* and *input format (vector -> 3-D tensor)*



下面我们将继续介绍通过convolution和pooling操作后参数数量的变化。对于下图中的例子，我们先考虑输入 28×28 大小的图像，即 $w = 28, h = 28$ ，这里默认步长为1，padding为0，经过 3×3 卷积核大小的过滤，图像的大小就变为 26×26

$$\frac{28 - 3 + 2 * 0}{1} + 1 = 26$$

由于输入的图像只有1个channel，卷积核大小为 3×3 ，因此第一层只有9个参数

由于前一层有25个filter，经过一次convolution操作后，图像大小为 26×26 ，本层的neural个数为 $25 \times 26 \times 26$ ；还需要再进行一次pooling操作，经过pooling操作之后图像大小为 13×13 ，max pooling操作没有filter，不算参数

```
model2.add(MaxPooling2D((2,2)))
```

pooling操作后图像宽度的大小为，

$$\frac{26 - 2 + 2 * 0}{2} + 1 = 13$$

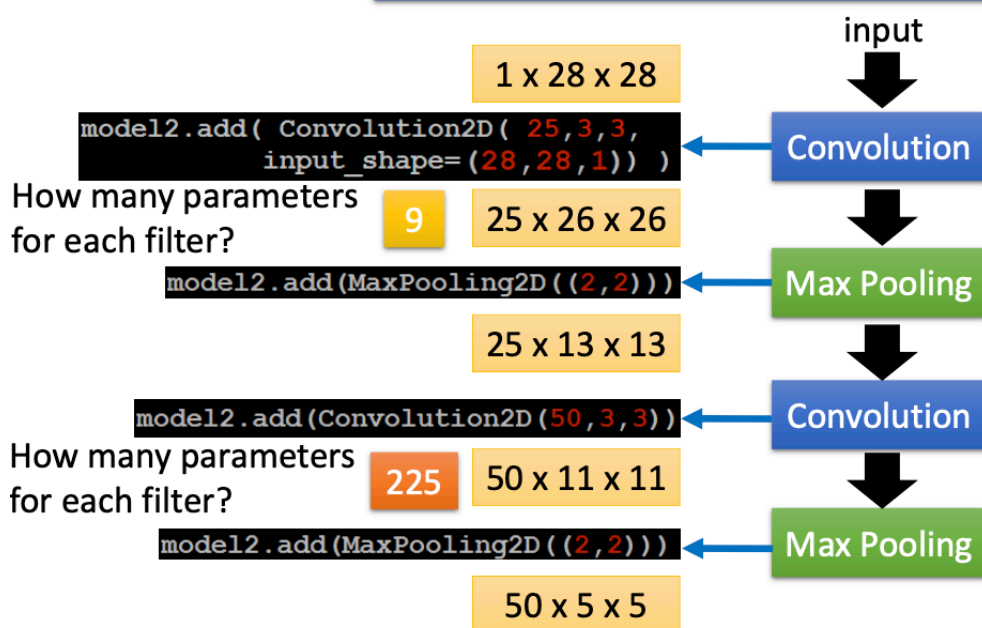
经过一次convolution和pooling操作之后，neural个数为 $25 \times 13 \times 13$ ；

对于第二次的convolution操作，有50个大小为 3×3 的filter，输出的neural数量为 $50 \times 11 \times 11$ ；一次pooling操作所包含的区域大小为 2×2 ，输出的neural数量为 $50 \times 5 \times 5$

$$\frac{13 - 3 + 2 * 0}{1} + 1 = 11$$
$$\frac{11 - 2 + 2 * 0}{2} + 1 = 5$$

CNN in Keras

Only modified the *network structure* and *input format (vector -> 3-D tensor)*



经过两次的convolution和pooling操作之后，neural的数量为 $50 \times 5 \times 5$ ，下一步操作就是将这些神经元flatten，展开成一维向量；再输入激活函数和全连接层

