

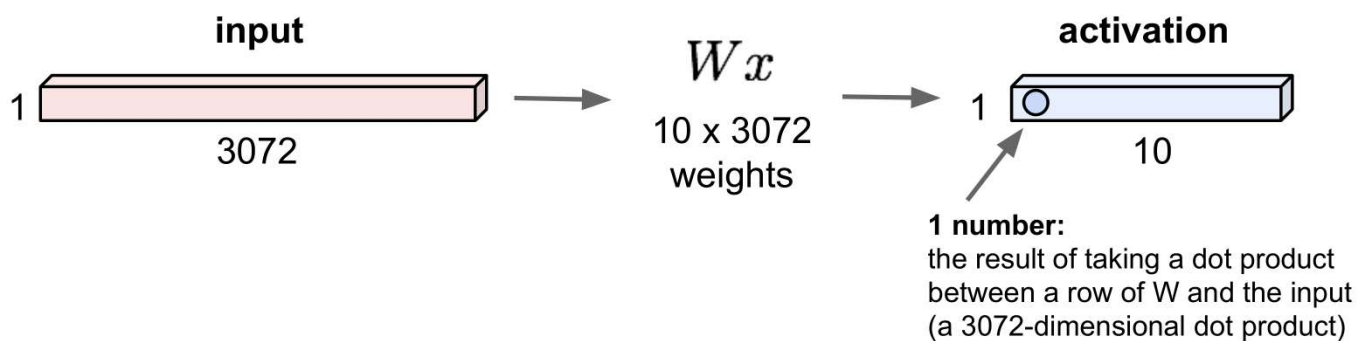
Lecture 5 Convolutional Neural Network

一、卷积神经网络的历史(略)

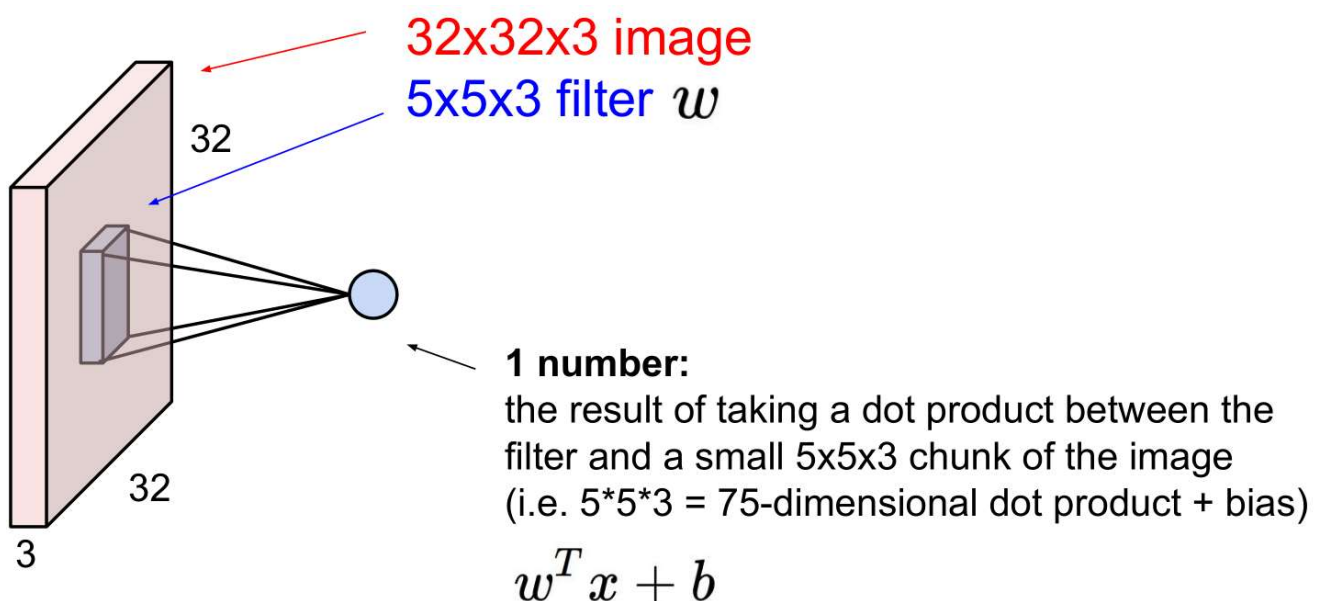
二、Convolutional Neural Network(卷积神经网络)

Fully-Connected Layers

32x32x3 image -> stretch to 3072 x 1

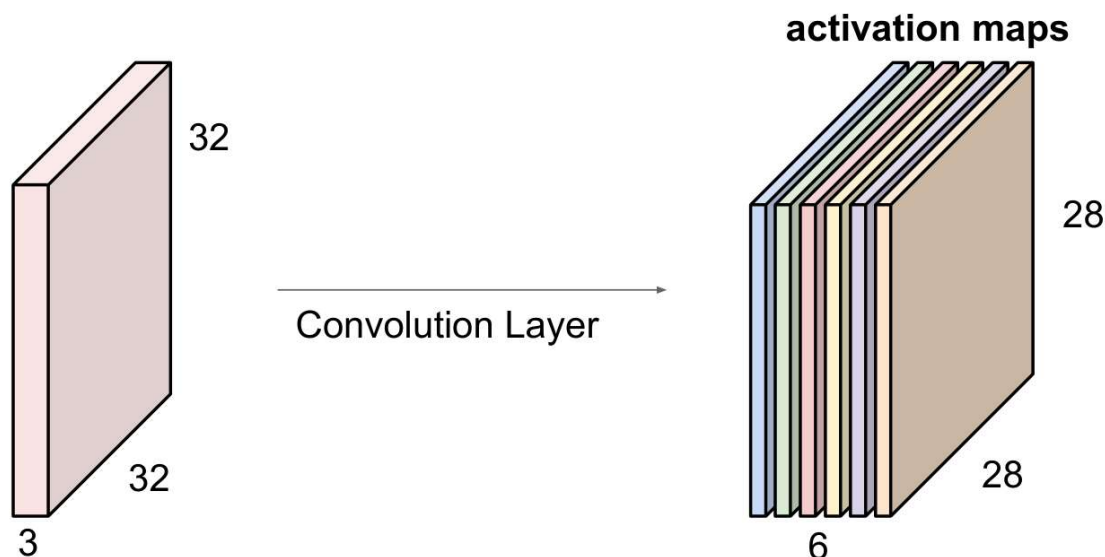


Convolution Layer



每一个卷积核在一次卷积中只会产生一个1x1x1的数，不管这个卷积核的第三维度是多少。

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

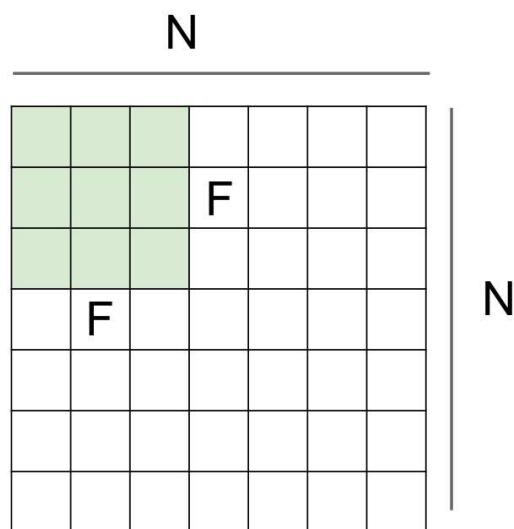


We stack these up to get a “new image” of size 28x28x6!

但是不同的卷积核是生成不同的**activation map**的，因为有6个卷积核，所以这一层产生的activation maps的维度是**28x28x6**

卷积计算规则

这里的卷积仅仅是卷积核与输入的相对应位置的数相乘然后整体相加得到最终的output,其中，输出的维度计算就如下图：



Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7, F = 3$:

stride 1 $\Rightarrow (7 - 3) / 1 + 1 = 5$

stride 2 $\Rightarrow (7 - 3) / 2 + 1 = 3$

stride 3 $\Rightarrow (7 - 3) / 3 + 1 = 2.33 \therefore \backslash$

因为存在着想上面步长为3的时候的不匹配的问题，我们就用了另一种方式解决这种不匹配——**padding**

In practice: Common to zero pad the border

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

7x7 output!

in general, common to see CONV layers with stride 1, filters of size $F \times F$, and zero-padding with $(F-1)/2$. (will preserve size spatially)

e.g. $F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

下面是卷积层的一些小总结：

Summary. To summarize, the Conv Layer:

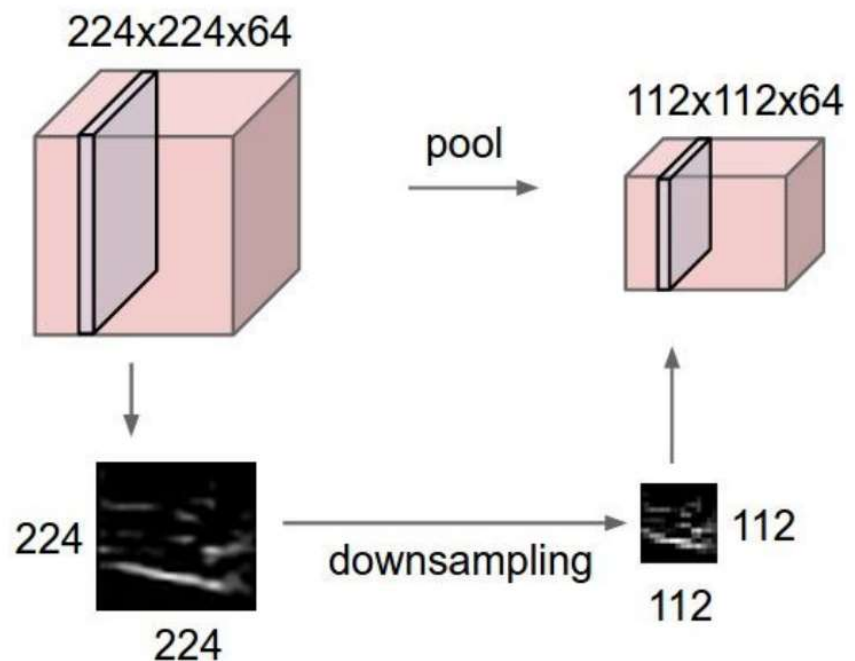
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters K ,
 - their spatial extent F ,
 - the stride S ,
 - the amount of zero padding P .
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F + 2P)/S + 1$
 - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d -th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d -th filter over the input volume with a stride of S , and then offset by d -th bias.

池化计算规则

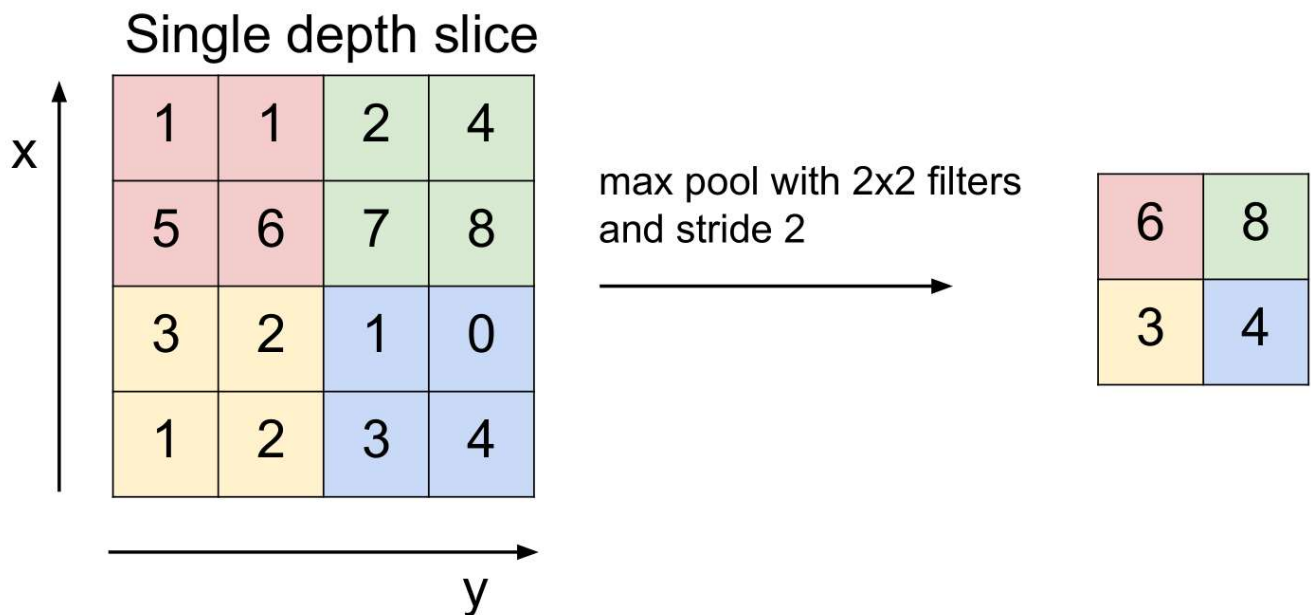
池化层实际上是一种降采样的方式：

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



常用的池化层类型是——**Max Pooling**



- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
 - their spatial extent F ,
 - the stride S ,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F)/S + 1$
 - $H_2 = (H_1 - F)/S + 1$
 - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

三、Summary(总结)

- ConvNets stack CONV, POOL, FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like
 $[(\text{CONV-RELU})^N \text{-POOL?}]^M \text{-(FC-RELU)}^K \text{, SOFTMAX}$
 where N is usually up to ~ 5 , M is large, $0 \leq K \leq 2$.
 - but recent advances such as ResNet/GoogLeNet challenge this paradigm