Convolution, Padding, Stride, and Pooling in CNN

Convolution operation

The convolution is a mathematical operation used to extract features from an image. The convolution is defined by an image kernel. The image kernel is nothing more than a small matrix. Most of the time, a 3x3 kernel matrix is very common.

In the below fig, the green matrix is the original image and the yellow moving matrix is called kernel, which is used to learn the different features of the original image. The kernel first moves horizontally, then shift down and again moves horizontally. The sum of the dot product of the image pixel value and kernel pixel value gives the output matrix. Initially, the kernel value initializes randomly, and its a learning parameter.

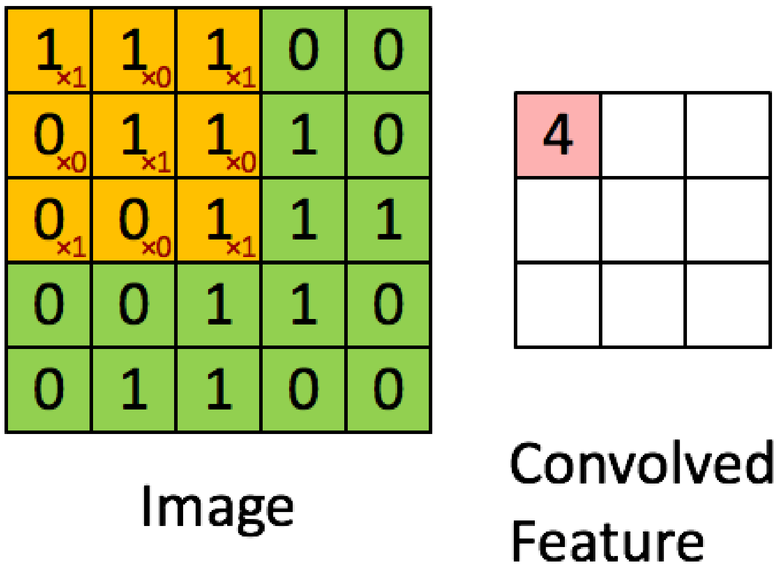
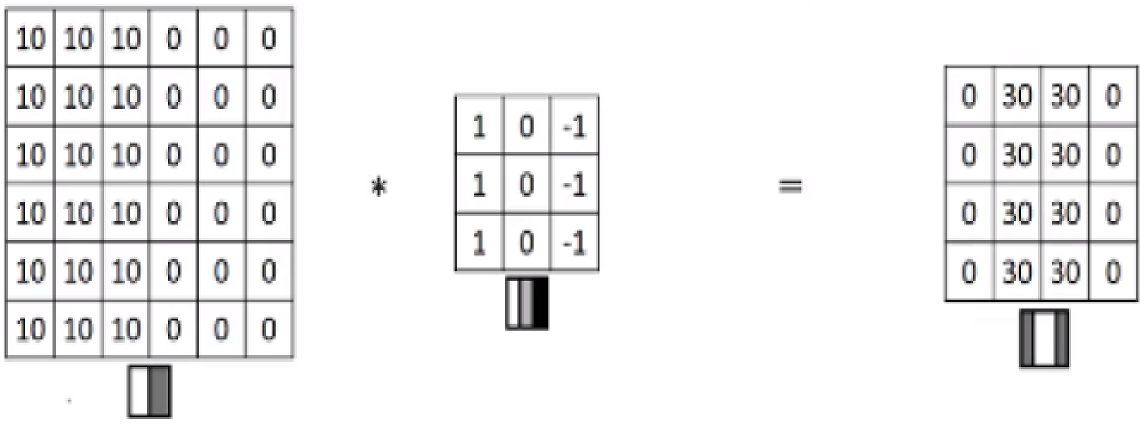


Illustration of the convolution operation



$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

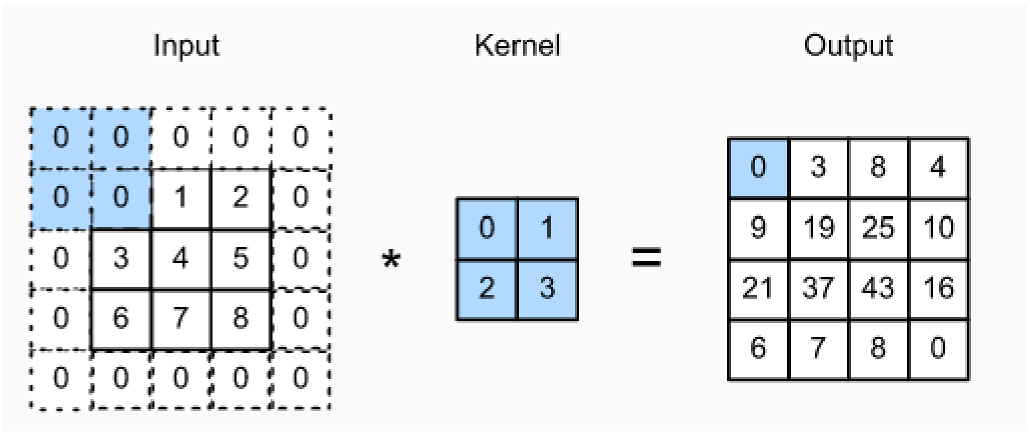
So if a 6*6 matrix convolved with a 3*3 matrix output is a 4*4 matrix. To generalize this if a $m * m$ image convolved with $n * n$ kernel, the output image is of size $\mathbf{(m - n + 1) * (m - n + 1)}$.

Padding

There are two problems arises with convolution:

1. Every time after convolution operation, original image size getting shrinks, as we have seen in above example six by six down to four by four and in image classification task there are multiple convolution layers so after multiple convolution operation, our original image will really get small but we don’t want the image to shrink every time.
2. The second issue is that, when kernel moves over original images, it touches the edge of the image less number of times and touches the middle of the image more number of times and it overlaps also in the middle. So, the corner features of any image or on the edges aren’t used much in the output.

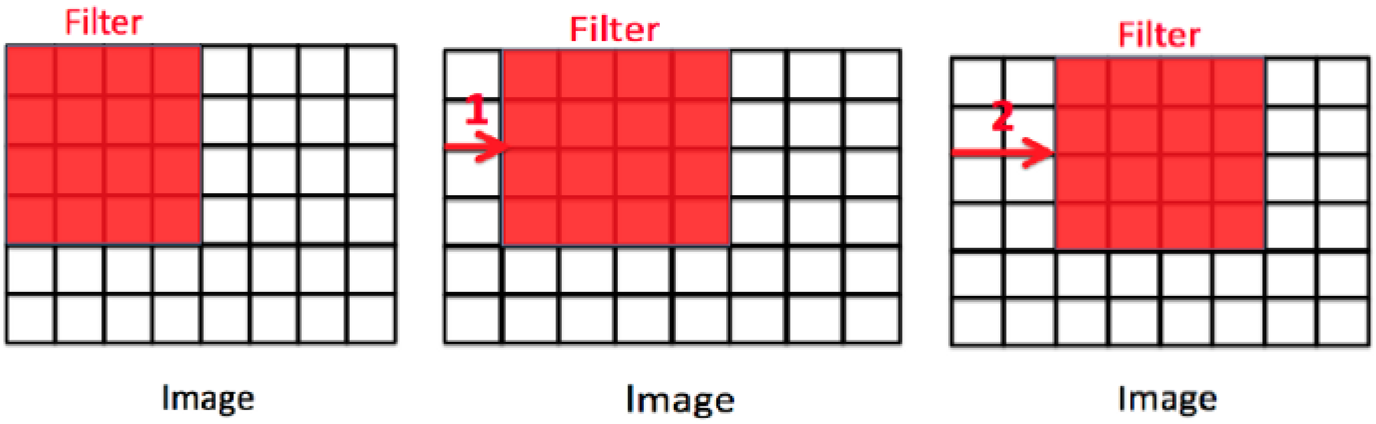
So, in order to solve these two issues, a new concept is introduced called **padding.** Padding preserves the size of the original image.



Padded image convolved with 2*2 kernel

So if a $n*n$ matrix convolved with an f*f matrix the with padding p then the size of the output image will be (n + 2p — f + 1) * (n + 2p — f + 1) where p =1 in this case.

Stride



left image: stride =0, middle image: stride = 1, right image: stride =2

Stride is the number of pixels shifts over the input matrix. For padding p, filter size $f*f$ and input image size $n * n$ and stride 's' our output image dimension will be $[\ \{(n + 2p - f + 1) / s\} + 1] * [\ \{(n + 2p - f + 1) / s\} + 1]$.

Pooling

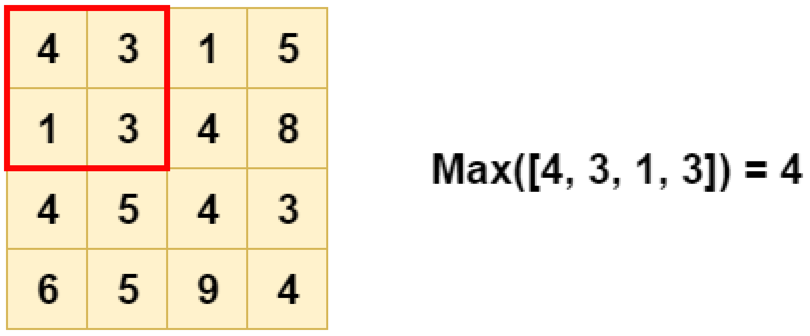
A pooling layer is another building block of a CNN**.** Pooling Its function is to progressively reduce the spatial size of the representation to reduce the network complexity and computational cost.

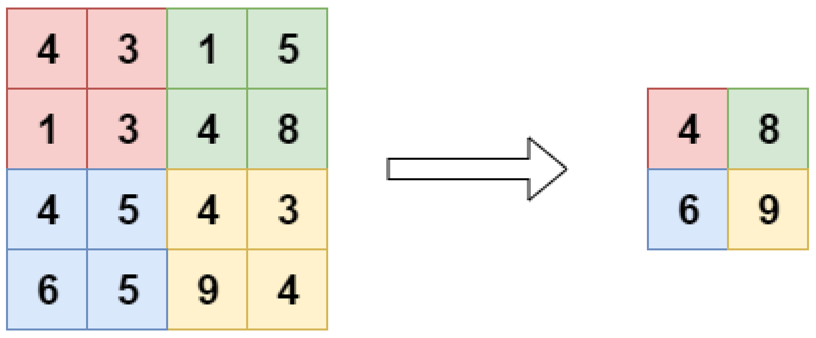
There are two types of widely used pooling in CNN layer:

1. Max Pooling
2. Average Pooling

Max Pooling

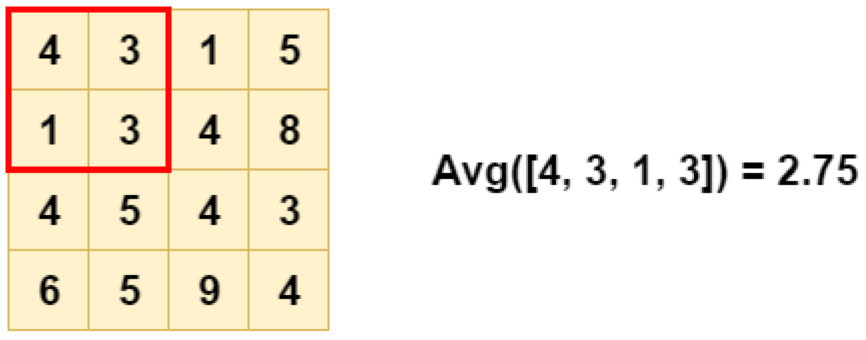
Max pooling is simply a rule to take the maximum of a region and it helps to proceed with the most important features from the image. Max pooling selects the brighter pixels from the image. It is useful when the background of the image is dark and we are interested in only the lighter pixels of the image.

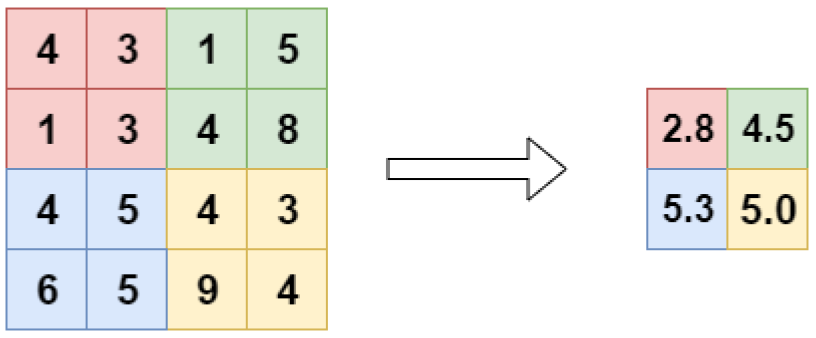




Average Pooling

Average Pooling is different from Max Pooling in the sense that it retains much information about the "less important" elements of a block, or pool. Whereas Max Pooling simply throws them away by picking the maximum value, Average Pooling blends them in. This can be useful in a variety of situations, where such information is useful.


4
3
1
5
1
3
4
8
4
5
4
3
6
5
9
4
Avg([4, 3, 1, 3]) = 2.75


4
3
1
5
1
3
4
8
4
5
4
3
6
5
9
4
2.8
4.5
5.3
5.0