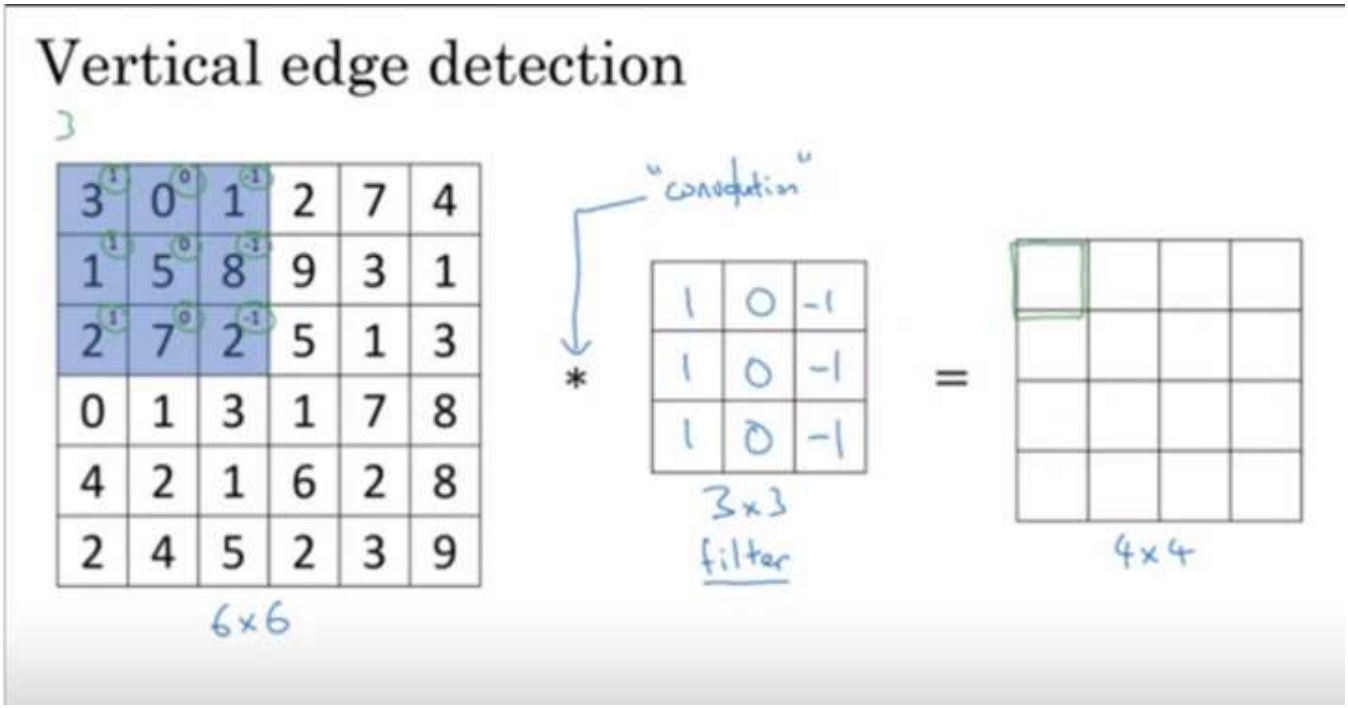
Ramit Agarwal Follow

May 29, 2020 · 5 min read · Lis

Save

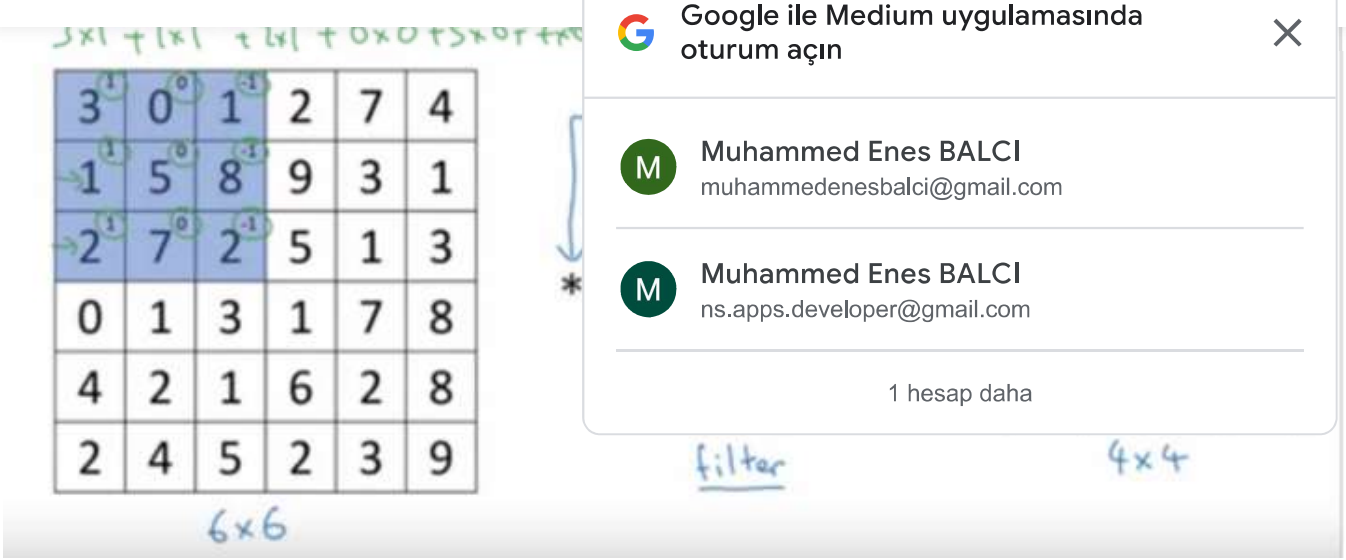
Performing Convolution On a Matrix

> *In this blog, we will be discussing about performing convolution on a 2D image matrix based on the intution from the deeplearning.ai CNN course*



We have “6x6” matrix which represents our image and “3 x3” matrices represents the kernel

So to perform convolution we overlap the kernel on the image matrix and multiply its every element with the element of the image matrix



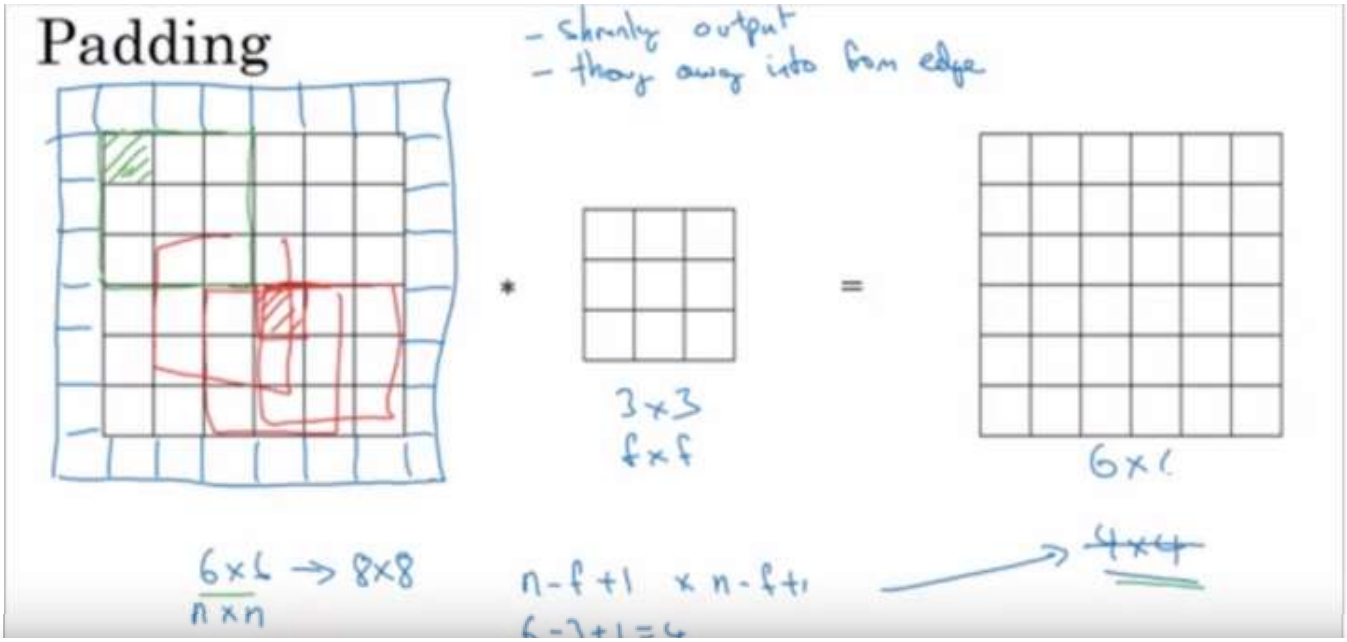
3x1 + 0x0 + 1x-1 + 1x1 + 5x0 + 8 x -1 + 2x1+ 7x0 + 2x-1 = -5

Similarly, we traverse the kernel through the whole image matrix and fill the entries in the output image matrix

But we can observe that the output image matric size is less than the original image matrix size so to deal with that problem we will use the concept of padding

Padding

To perform padding we will add extra cells symmetrically to the original image matrix



Calculating Size of Output Imag

If the input image matrix is of size "r
defined padding as p then

Then the size of the output image ma

If we want the output size as the sam

$n + 2p - f + 1 = n$

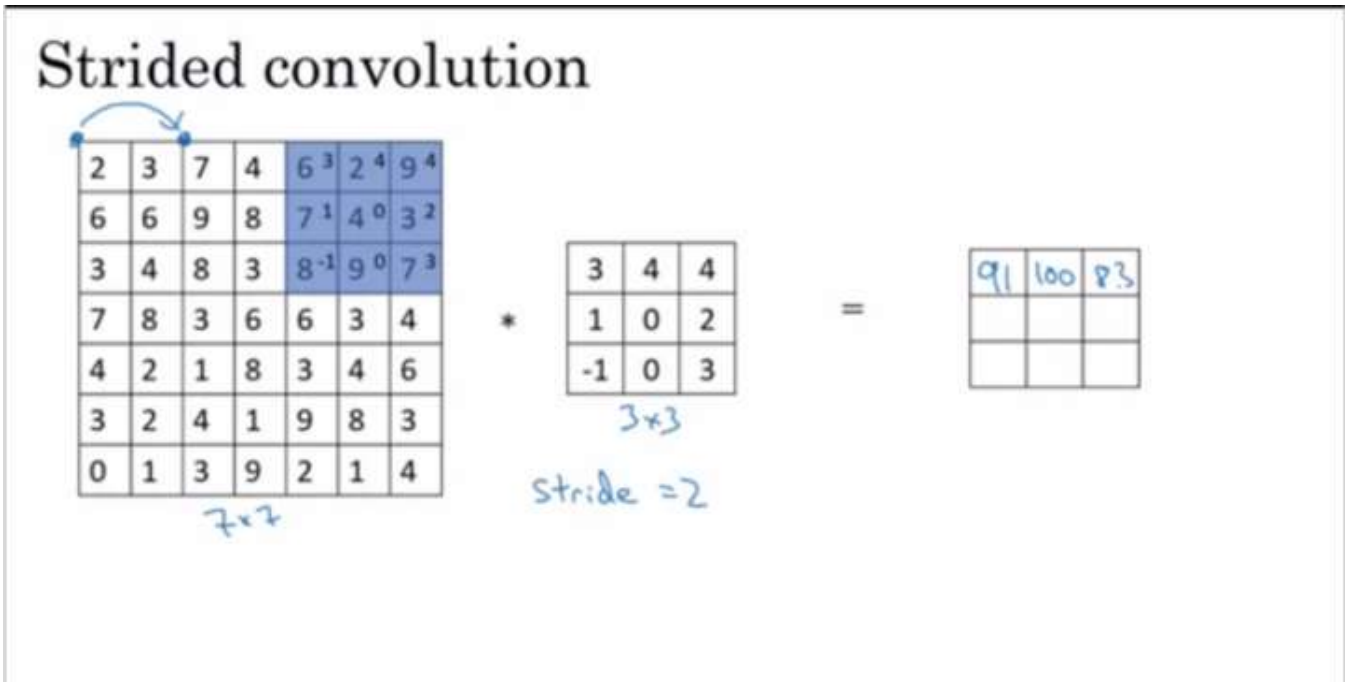
$2p = f - 1$

$p = (f - 1) / 2$

So if we want our output image to have the same shape as input image we will use padding = (f-1)/2 for our input image where f is the size of our kernel

Stride

This is another concept which is associated with performing convolutions

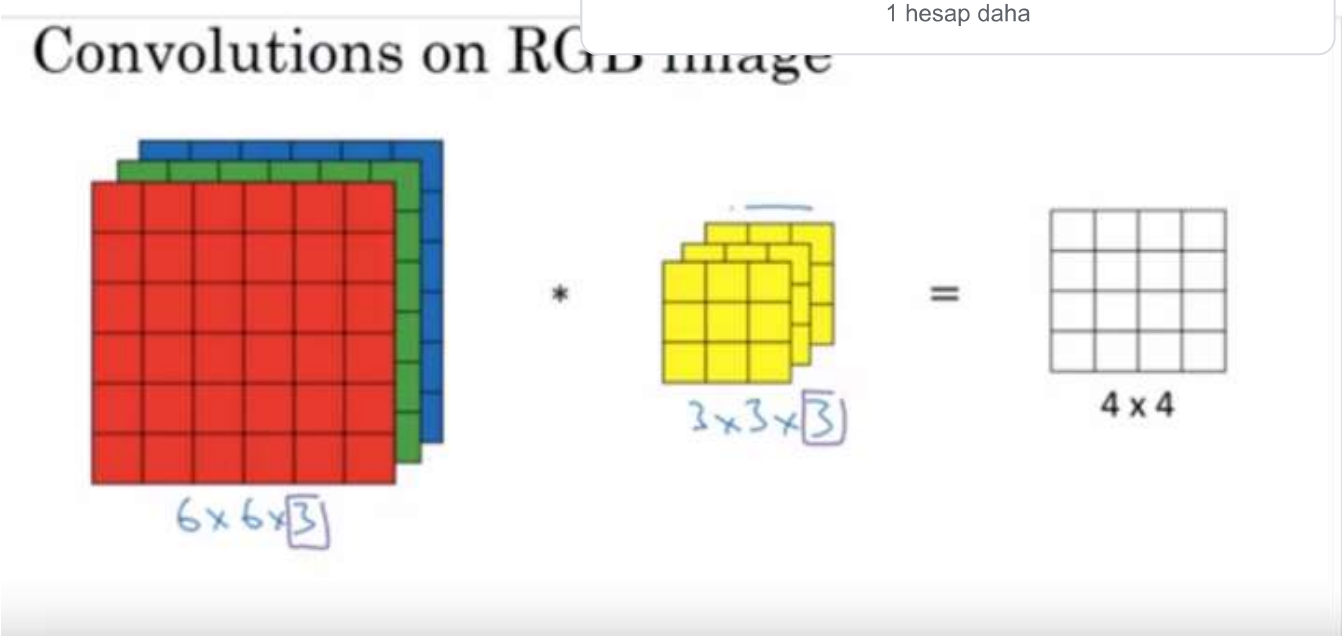


If stride =1 then the kernel shift by one unit to traverse through whole input image

Then the shape of the final output in

math.floor((n + 2p -f)/s +1)

Convolution over volume



Most of the time we will be working with 3D image

The “6 x 6 x 3" of the image can be interpreted as “n_h x n_w x n_c”

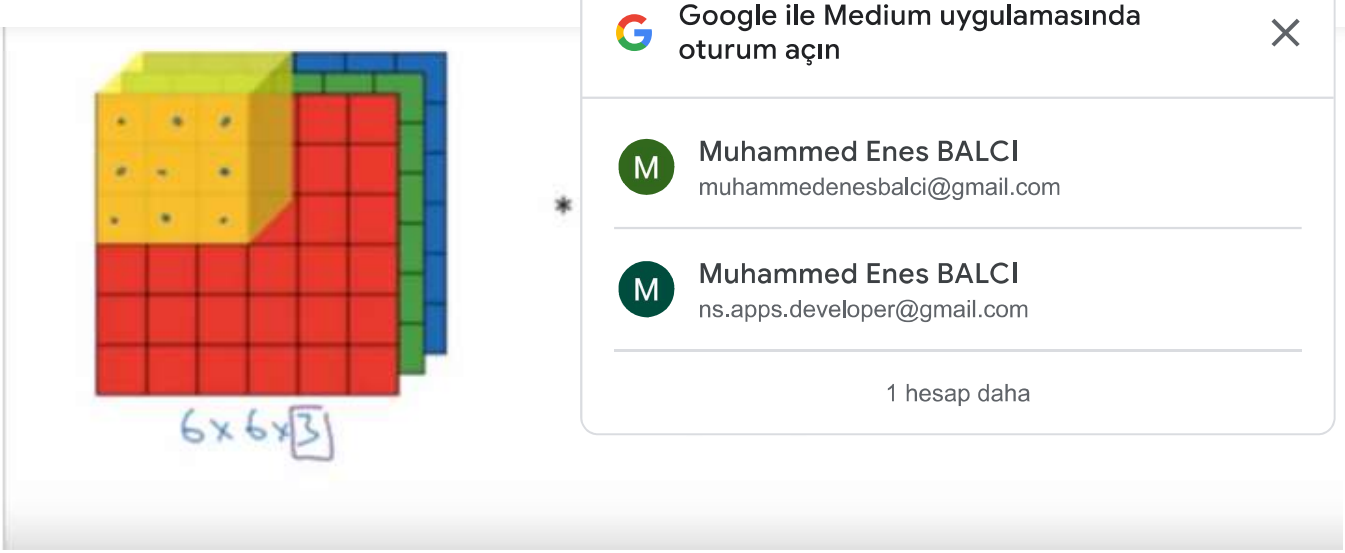
n_h: The height of the image

n_w: the width of the image

n_c: Number of channels of the image

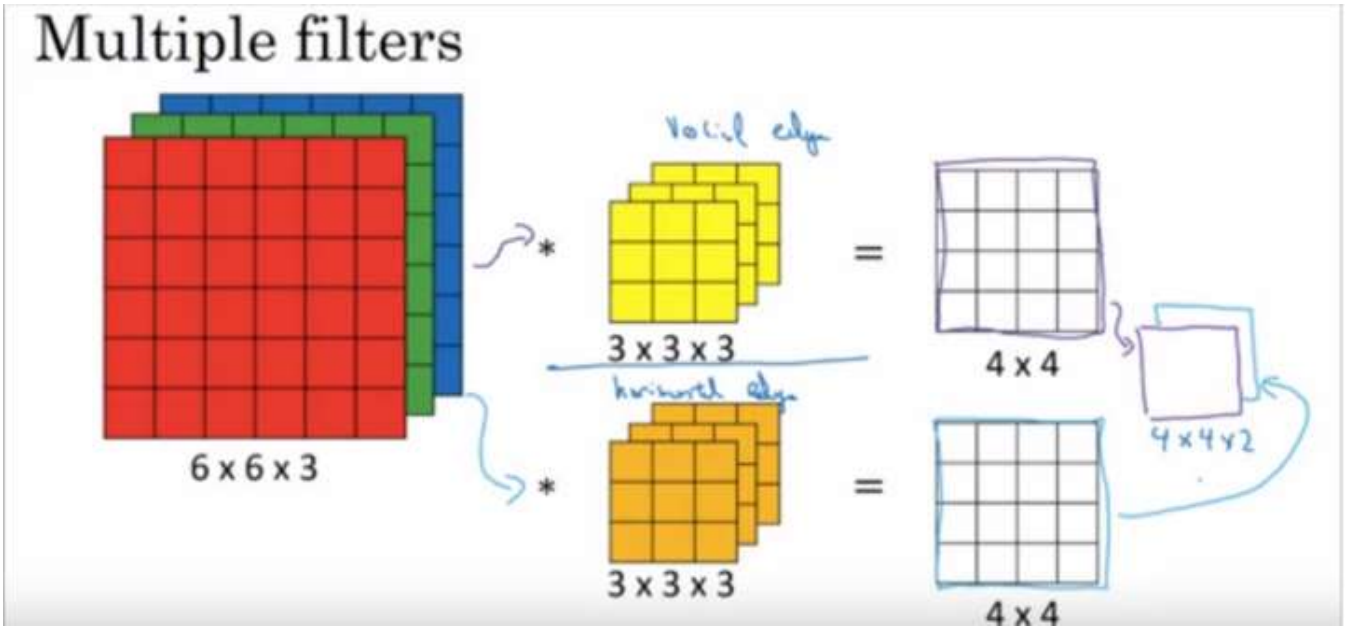
While performing convolution over the image we need to take care that our kernel should have the same number of channels as that of the input image so it can perfectly overlap with the input image

We will get a 2 d matrix as output



If we apply another kernel of the same size as previous it will give a different output layer

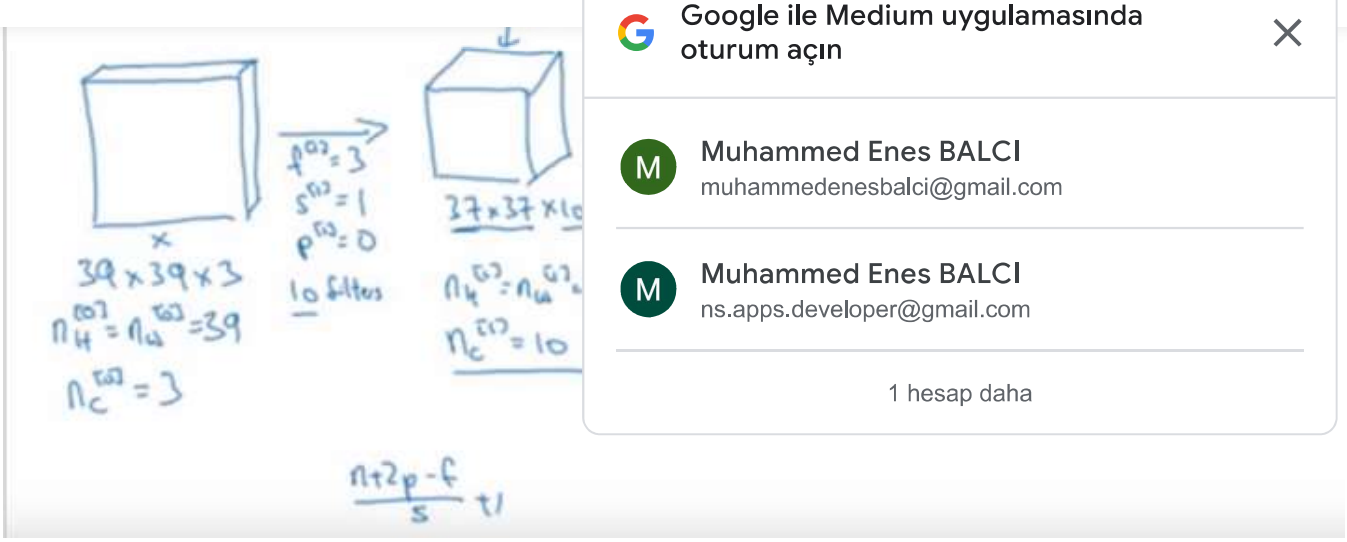
We can concatenate all the layers to form a 3d output image matrix



If input size is n x n x n_c and the size of the kernel is f x f x n_c and if applied K kernels of the same size to the input image with padding of p and stride of s then the output image matrix will be

(n -f +2p +1/s) x (n-f +2p +1/s) x K

How a Convolutional Layer looks like



We have taken input image as “39 x 39 x 3” matrix then we applied the convolution using a kernel of size “3 x 3” with stride as 1 and padding as 1

We have applied such 10 filters on the image

Applying the formula

(n +2p -f)/s + 1

(39 + 2(1)- 3)/1 +1 = 37

The shape of the output image equal to “37 x 37 x 10”

37: we get from the above calculation

10: as we applied the filter 10 times

Similarly, we can do this for the image obtained having a shape as“37x 37x10"

We will use the kernel of size as “5 x 5” with padding equal to 0 and stride equal to 2

Now we will again do the calculation using the formula

math.floor(n +2p -f)/s + 1

math.floor((n +2p -f)/s + 1) = 17

20: as we applied the filter 10 times

This way we can apply convolution a
purposes

Slide snippet has been taken from th