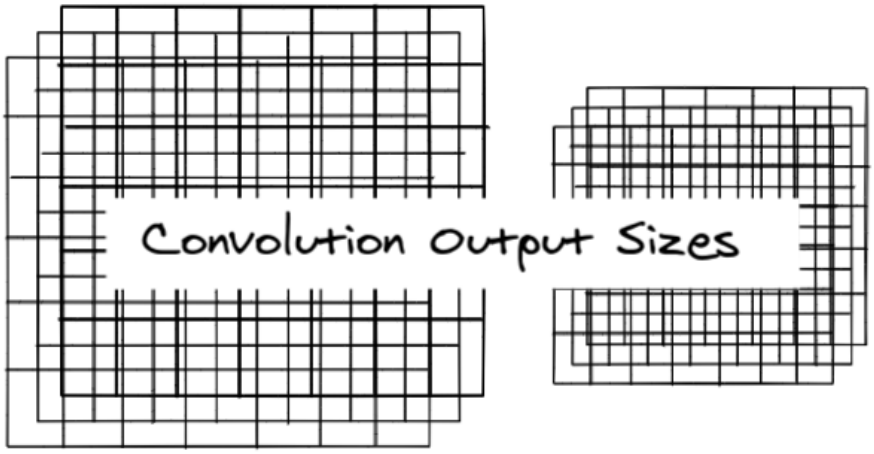
Twitter Thread by Prashant

Prashant
@capeandcode

Calculating Convolution sizes is something that I found particularly hard after understanding convolutions for the first time.

I couldn't remember the formula because I didn't understand its working exactly.

So here's my attempt to get some intuition behind the calculation.■■



BTW if you haven't read the thread ■ on 1D, 2D, 3D CNN, you may want to check it out once.

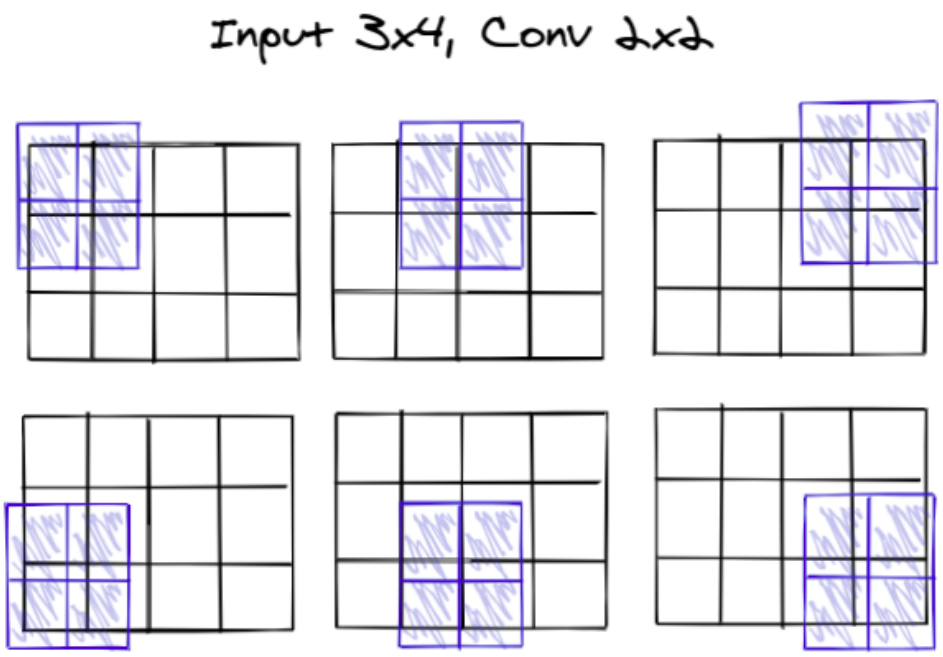
https://t.co/kxZ6ZeHCrk

> Convolutions! 1D! 2D! 3D!\U0001f532
>
> I've had a lot of trouble understanding different convolutions
> What do different convolutions do anyway\u2753
>
> Without the correct intuition, I found defining any CNN architecture very unenjoyable.

So, here's my little understanding (with pictures)\U0001f5bc\U0001f447 pic.twitter.com/dCu70j6Ep6

— Prashant (@capeandcode) April 14, 2021

First, observe the picture below■



The 2 x 2 filter slides over the
3 rows, 2 times and,
4 columns, 3 times

So, let's try subtracting the filter size first
3 - 2 = 1
4 - 2 = 2

Looks short, we'll need to compensate the 1 in both.
3 - 2 + 1 = 2
4 - 2 + 1 = 3

hence the formula so far becomes:

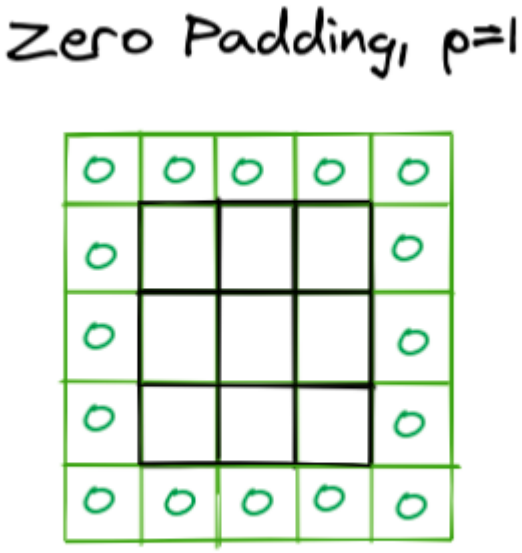
$$=> W - k$$

$$=> W - k + 1$$

Now let's discuss padding0■

Zero padding makes it possible to get output equal to the input by adding extra columns.

It provides extra space for the sliding, making up for the lost space



Padding of p would mean increasing the input size by adding p to both sides

Considering width, there will be padding for left and for right, both equal, same for the height.

The modified formula becomes

$$=> (W + p_l + p_r - k) + 1$$

$$=> (W + 2P - k) + 1$$

All of our calculation so far assumes that we are taking one step at a time during sliding, a stride of 1

What if we take more than that?■

We will be cutting our distance short by increasing the size of our leap. So to account for this we will divide with stride size.

$$=> \left[\left(W + 2P - k\right) \Big/ S\right] + 1$$

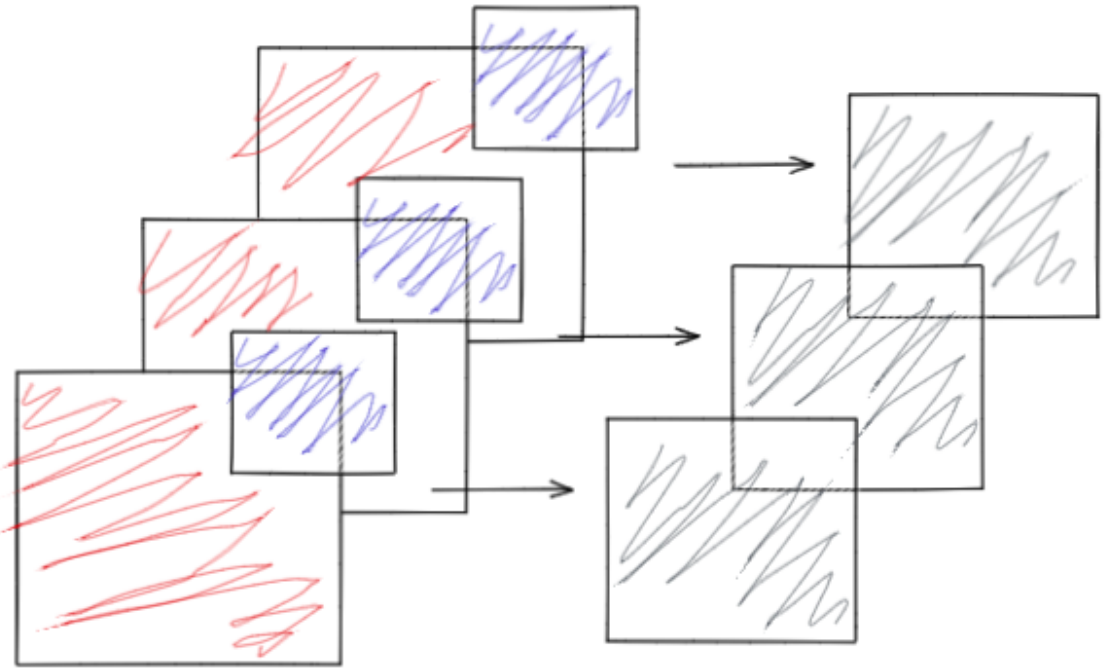
$$=> \left[\left(W - k + 2P\right) \Big/ S\right] + 1$$

Keep in mind that we should make sure that the calculation doesn't go into decimals.

We generally select our values in such a way that the calculations result in an integer.

Now as we may remember from the last thread, that one filter leads to 1 output, be it 1D, 2D...

So the depth of the output will be equal to the number of filters applied.



With all that in mind, let's try to solve a simple question below:

.

Take input size= $(64, 64, 3)$, kernel size= $(3, 3)$ and 40 filters
When omitted we'll assume no padding and stride = 1

This is 3D input for 2D convolution. The depth of kernel is matched with input depth so kernel would become $(3, 3, 3)$

Output Height = Output Width :

$$(64 - 3) + 1 = 62$$

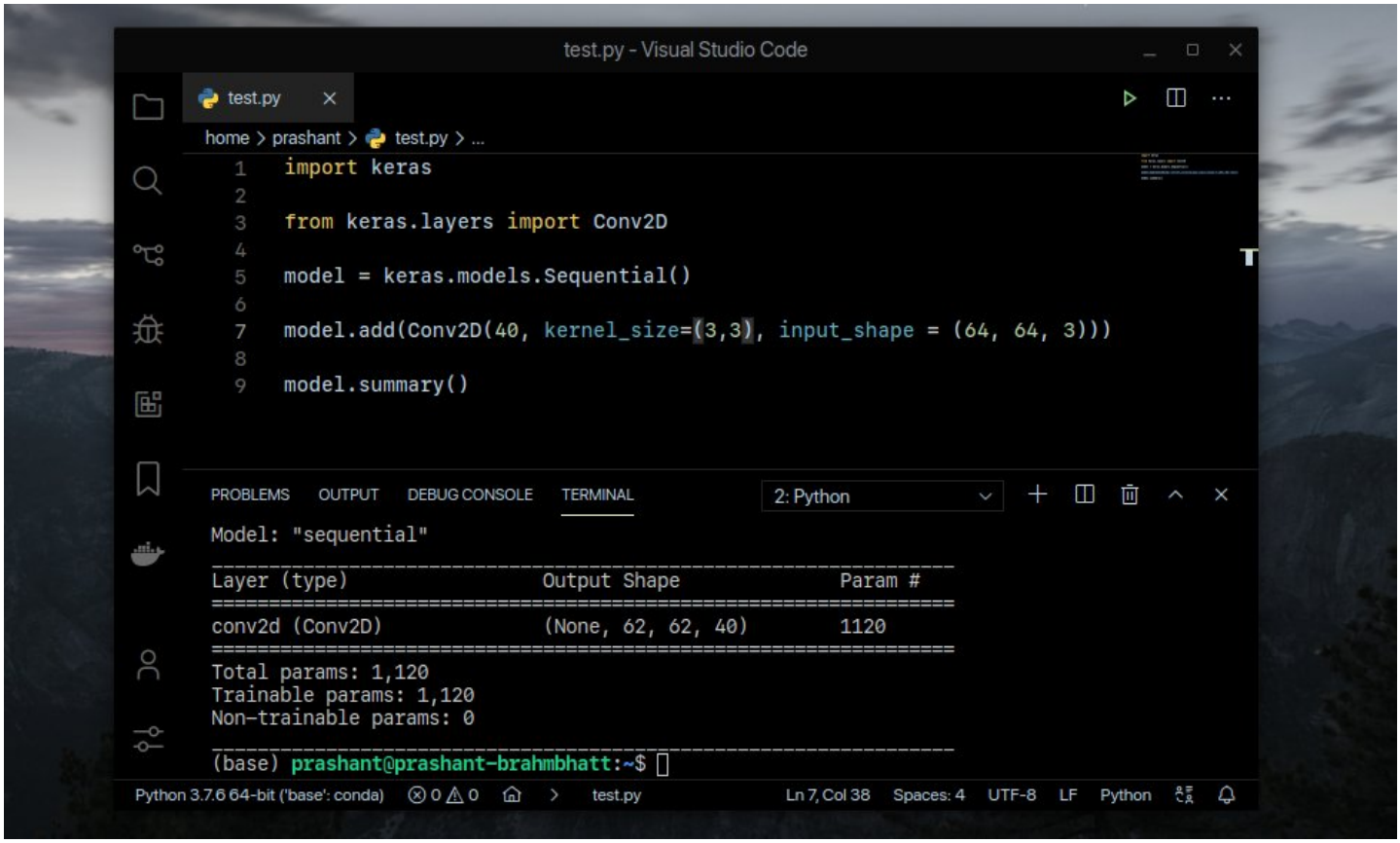
Actually,

$$[(64 - 3 + 2 * 0)/1] + 1 = 62$$

.

Hence final output size is $(62, 62, 40)$

We can try the same using Keras and its functions.



This website is a ConvNet shape calculator with which you can play around a little bit for better understanding.

https://t.co/8rU5BOAkts

Now if you feel like you can calculate correctly, try to pick any network and calculate its output sizes, validate using the model summary.

Or pick 3D convolutions and calculate its outputs, the principle remains the same.

All the above points helped me to be able to understand the CNN architectures better and not be bothered by the output summary.

Hope this helps you too! ■


That's all Folks!