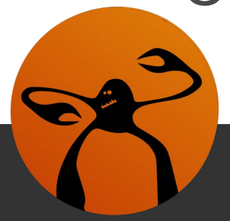
Deep Learning

Big Data & Machine Learning Bootcamp - Keep Coding

Outline

1. Computer Vision
2. Edge detection
3. Padding
4. Strided convolutions
5. One layer of a convolutional network
6. Simple convolutional network
7. Pooling layers
8. Convolutional neural network (CNN) example
9. Why convolutions?



1. Computer Vision

Computer vision is one of the areas that is **rapidly advancing thanks to the improvements made in deep learning.**

Computer vision community have been so creative and inventive coming with **new architectures** that not only they can be applied to computer vision task but also speech recognition.

*It is important to realise that this area covers not only what humans can see but also **what humans cannot see***

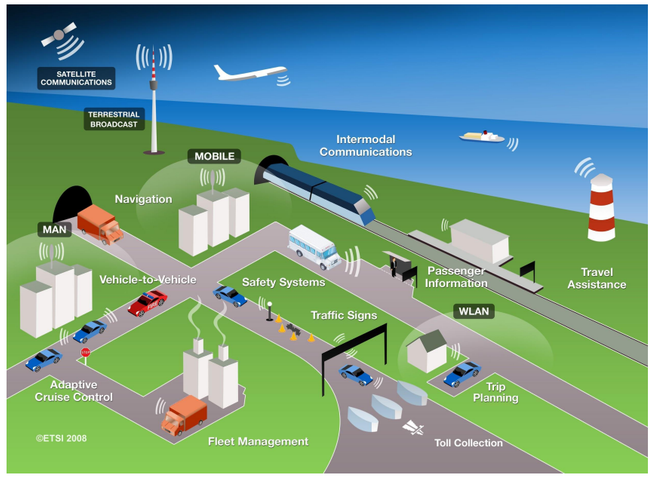
Sources:
- Coursera

1. Computer Vision

Application ranging from modeling objects through observation for **virtual reality applications**. Creation of Virtual Reality and Mixed Reality contents, increasing the reality of the synthesized image in MR

Physics-based vision theories for modeling object appearances and illumination environments. For instance magnetic resonance images, space images, and other images based on infrared or other electromagnetic signals

Recognition and classification of vehicles for Intelligent Transportation Systems (ITS) applications, and 3D modeling of real exterior environments by merging photometric and geometric images.

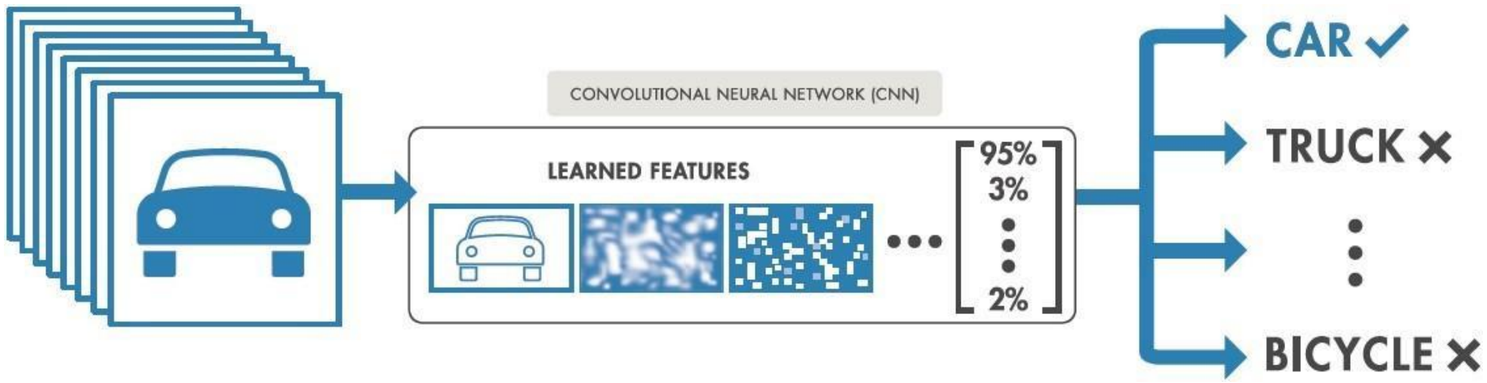


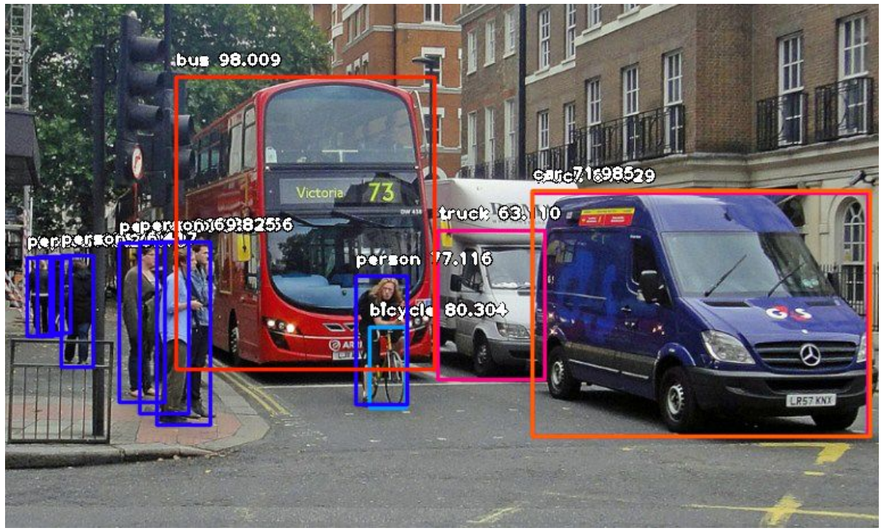
Sources:
- https://mubbisherahmed.wordpress.com/2011/11/29/the-future-of-intelligent-transport-systems-its/
- https://weetracker.com/2019/05/15/tanzania-largest-producer-of-a-life-saving-gas-helium/

1. Computer Vision

Many of previously described application involve these fundamental ones:

- Image classification -> Sometimes also called image recognition. You assigned a label to an image
- Object detection -> Not only recognize the objects but also draw boxes around them





Sources:
- https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606
- https://medium.com/intro-to-artificial-intelligence/simple-image-classification-using-deep-learning-deep-learning-series-2-5e5b89e97926

1. Computer Vision

Many of previously described application involve these fundamental ones:

- Neural style transfer -> In this task you paint an image using a style of different one

Sources:
- Coursera
- https://towardsdatascience.com/real-time-video-neural-style-transfer-9f6f84590832

2. Edge detection

For computer vision applications it is common to work/use large images.

It is better to use **convolutional networks** as using standard neural networks will need thousand or millions of neurons (Width x Height x 3) in the input layer. For instance for 1024 px * 1024 px images:

1024 px * 1024 px * 3 = 3.145.728

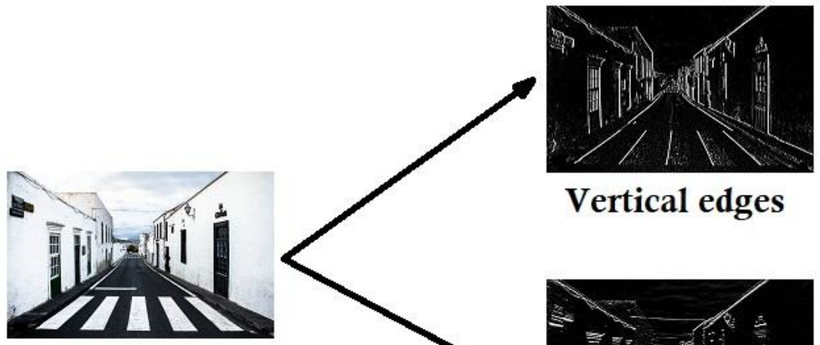
The convolution operation is one of the fundamental blocks of convolutional neural networks. By analyzing the edge detector we'll understand how this works!

Sources:
- Coursera

2. Edge detection

We've previously talked about how the neural networks extract different features at different layers.

But how the CNN detect edges?



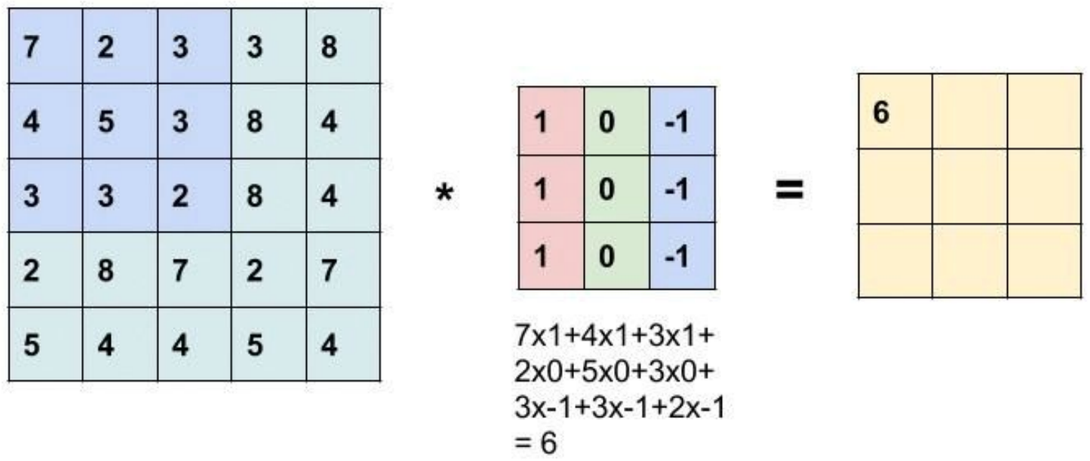
Vertical edges

Horizontal edges

Sources:
- Coursera
- http://datahacker.rs/edge-detection/

2. Edge detection

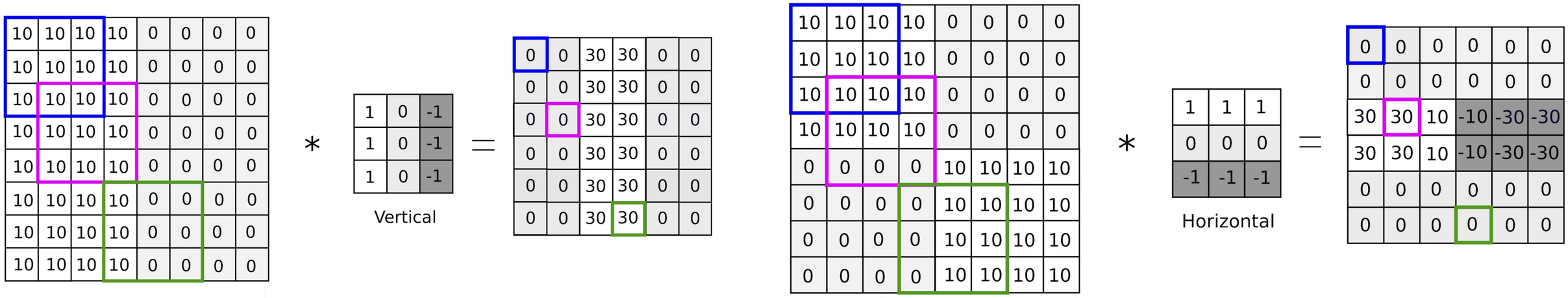
First, how convolution works:



Sources:
- Coursera
- https://medium.com/datadriveninvestor/convolutional-neural-networks-3b241a5da51e

2. Edge detection

For vertical and horizontal edge detection:



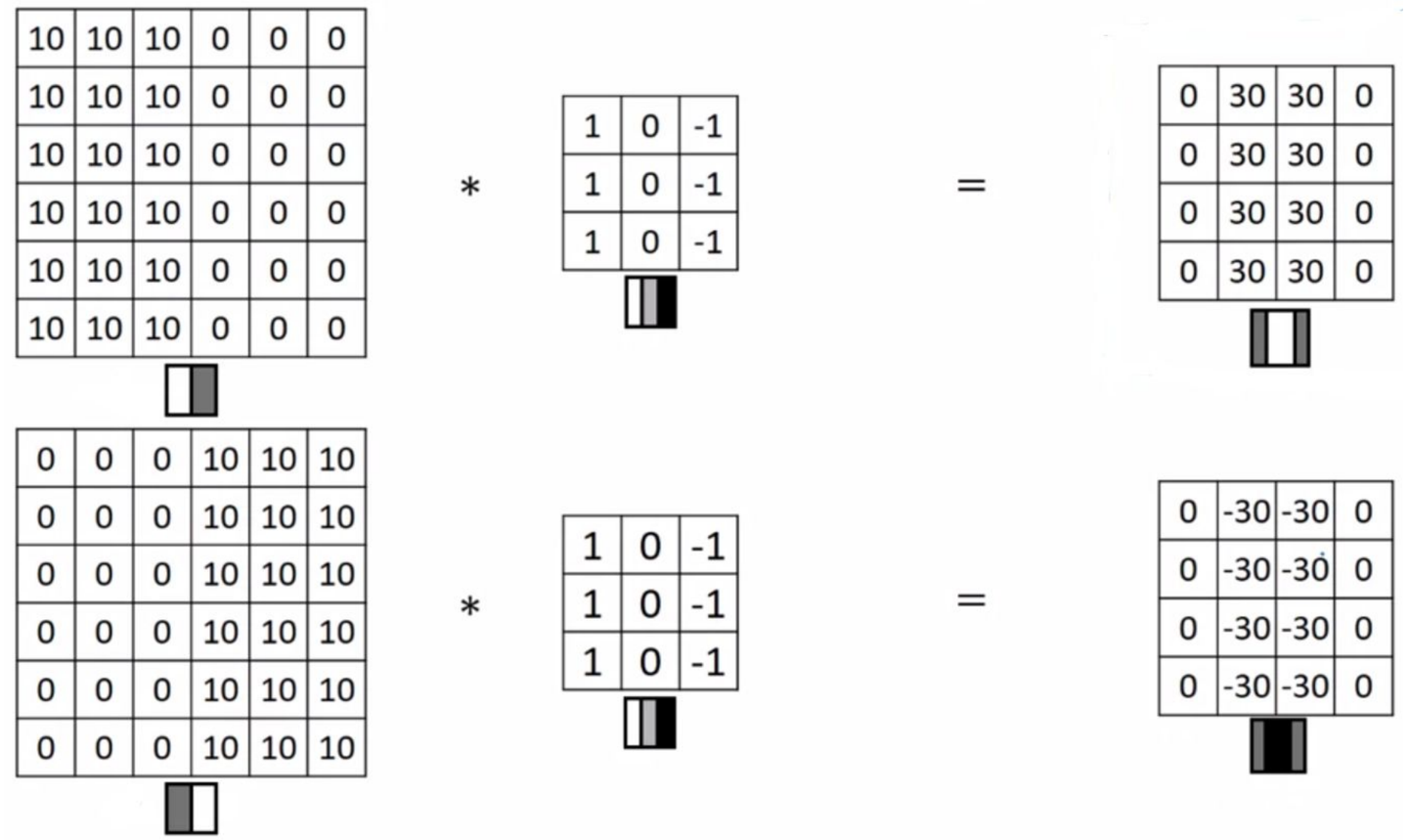
To obtain the images on the right, we perform an element-wise product between the filter/kernel and the input image (left). **The filter only cares about what's on the left and what's on the right. It doesn't care what's in the middle**

Sources:
- Coursera
- https://anhreynolds.com/blogs/cnn.html

2. Edge detection

It doesn't matter whether the transition is from bright to dark or dark to bright. You can take the absolute values.

This input images are small for showing purposes. Think of 1024 px images



Sources:
- Coursera

2. Edge detection

There are other types of filters such as:

- Sobel filter
- Scharr Filter

BUT the idea is to **make the neural network to learn/choose which filter is better**.

Maybe diagonal filters in combination with vertical and horizontal filters work better. Let the network decide that through backpropagation.

The filter/kernel values are parameters that the network learn in order to extract the best features!

THIS is one of the best ideas in computer vision!

Sources:
- Coursera

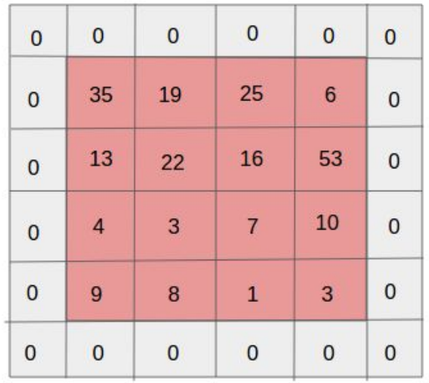
3. Padding

Padding is a modification made to the basic convolutional operation that solves two problems:

- Shrinking output
- Throwing away information from edge

This modification allows us to have **same size of image after convolution** and also **process more than one time the edge pixels** by just adding zeros around the input image.

In this image padding = 1



Sources:
- Coursera
- https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

3. Padding

There are two common choices on how much to pad:

- **Valid convolutions** -> NO PADDING.
  The output image will be (n - f +1) x (n - f + 1) where n is the size on the input image and f the size of the filter/kernel

- **Same convolution** -> Pad so that output size is the same as the input size
  Padding = (filter size - 1) / 2

  As as side note, the size of the filter is usually odd! i.e. 3, 5, 7

Sources:
- Coursera

4. Strided convolutions

Stride is another modification of the basic convolution operation.

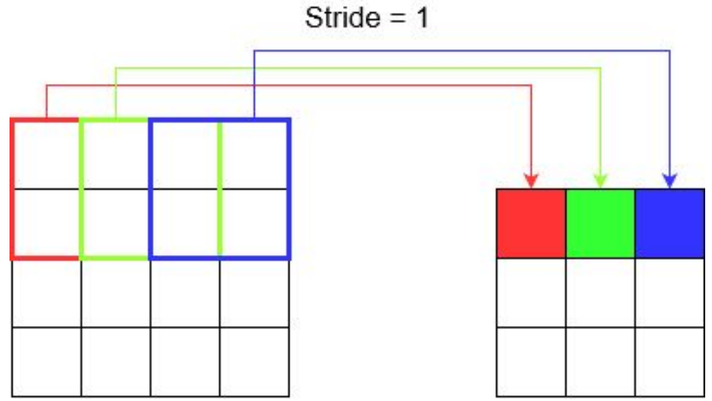
Basically, stride is the **number of rows and columns we traverse per slide** when doing the convolution.

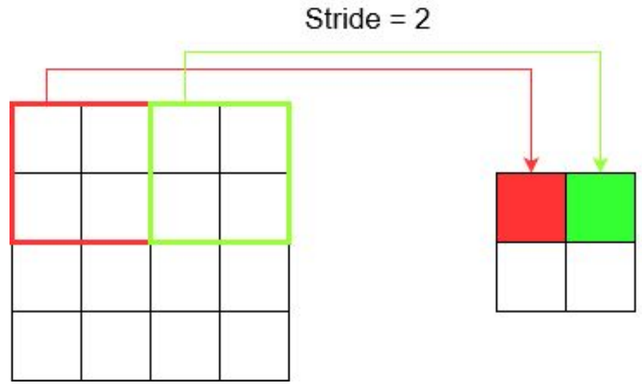
The output width and height are specified by these equations:

$$\text{output width} = \frac{W - F_w + 2P}{S_w} + 1$$

$$\text{output height} = \frac{H - F_h + 2P}{S_h} + 1$$

When the output values are not integers, we take the floor operation. This means we round the values down





Sources:
- Coursera
- https://stackoverflow.com/questions/42883547/intuitive-understanding-of-1d-2d-and-3d-convolutions-in-convolutional-neural-n

4. Strided convolutions

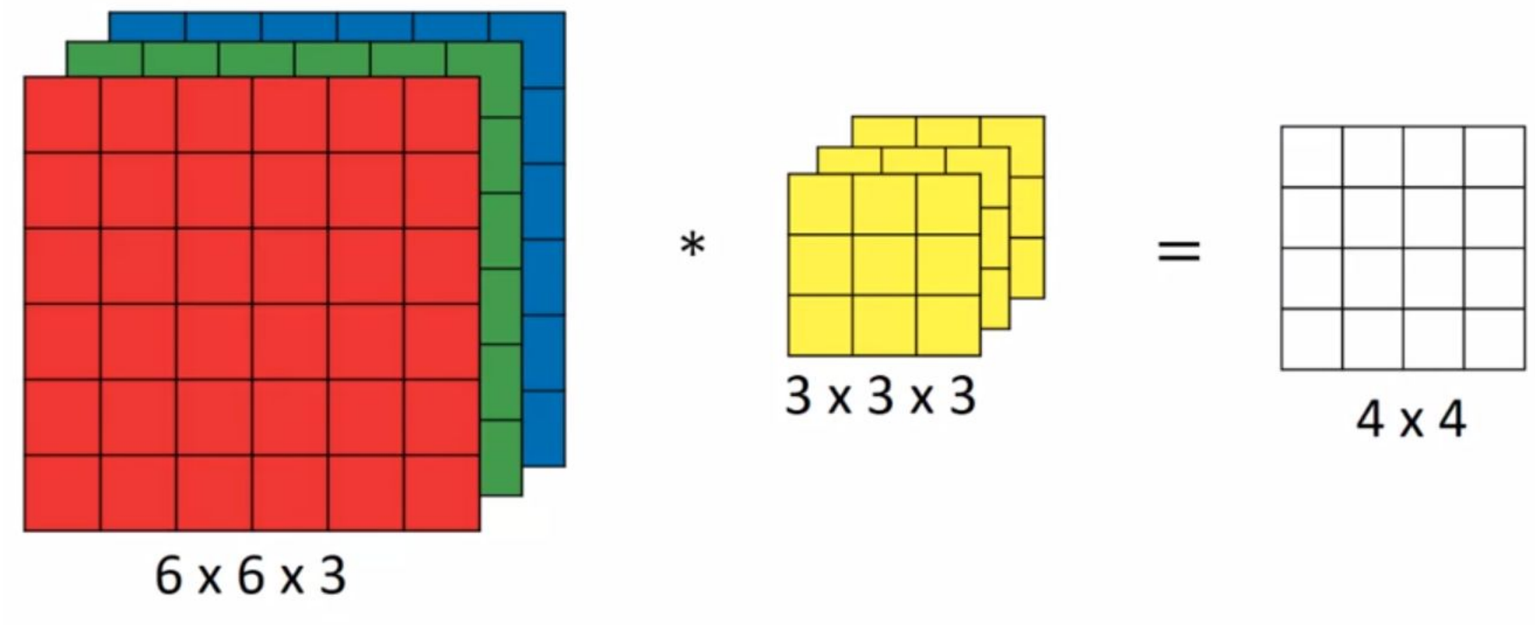
As a side note: In math books/articles you'll find that this operation we call convolution is called **cross-correlation.** Which is actually the correct name for this.

However, **the majority of articles/DL researchers agreed on calling this operation "convolution"**, even though we don't invert the filter/kernel before sliding it in the image

Sources:
- Coursera

5. One layer of a convolutional network

So far we've discussed convolutions over 2D images. Let's see how it works over three dimensional volumes:

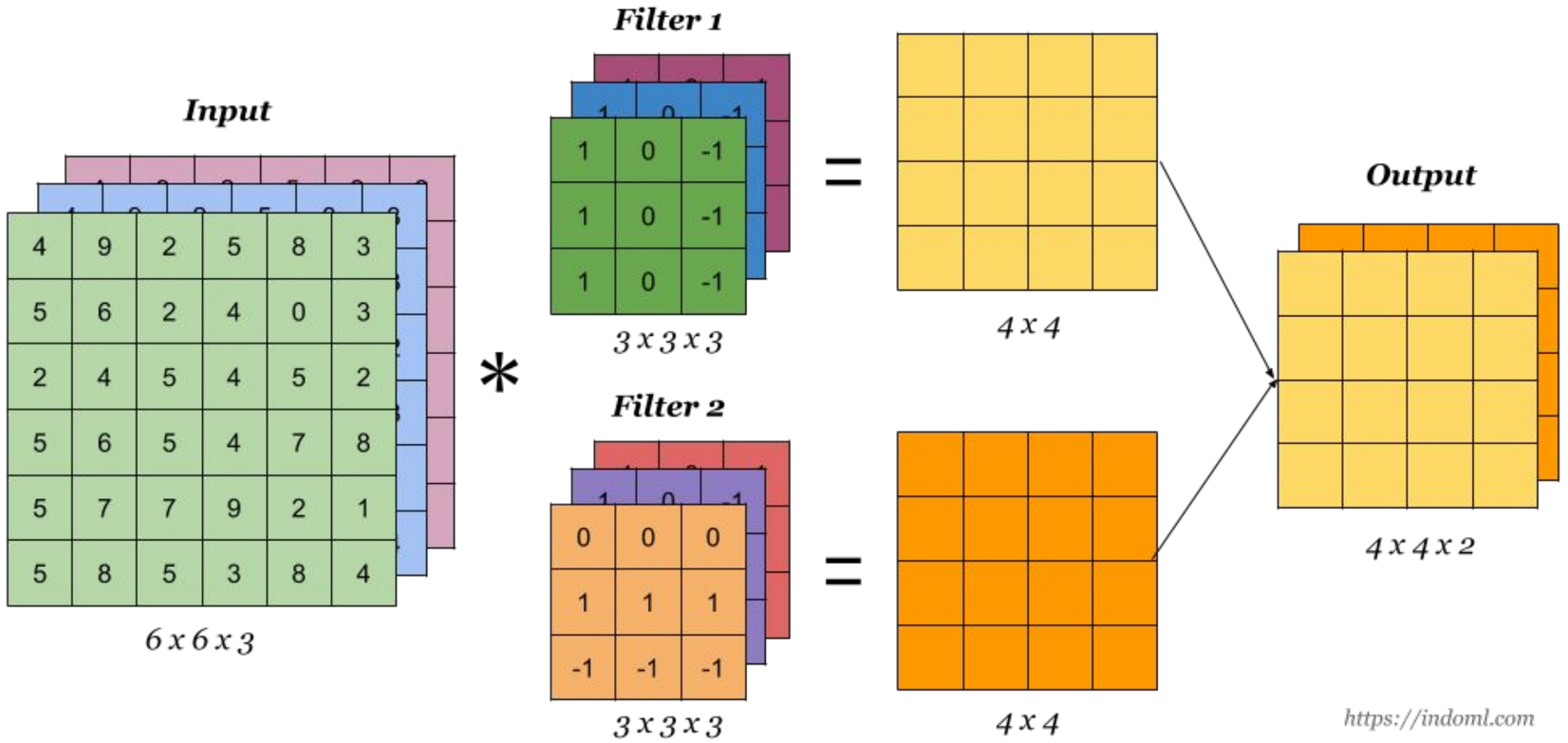


These 3D filters allow to extract edges from red, green or blue images

Sources:
- Coursera

5. One layer of a convolutional network

But we use more than one filter!

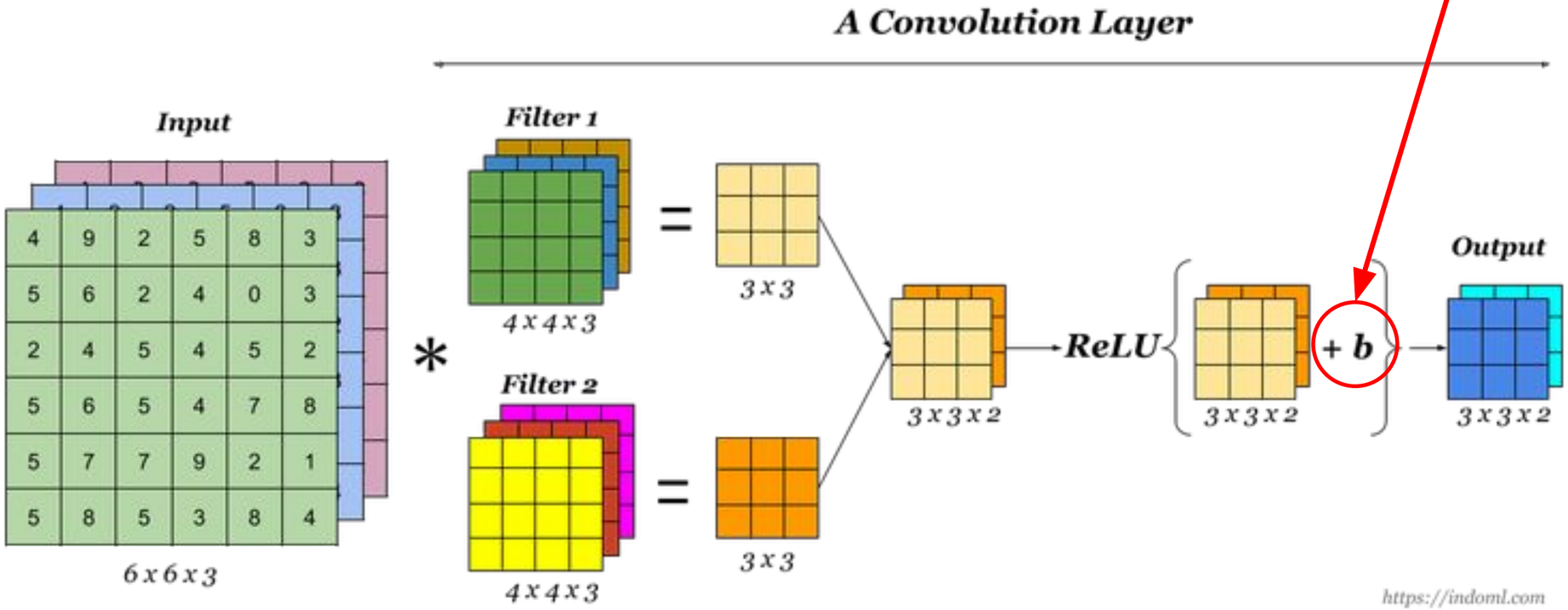


Sources:
- Coursera
- https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

5. One layer of a convolutional network

Now we know how to implement **a convolutional layer!**

We add the bias before the activation function!



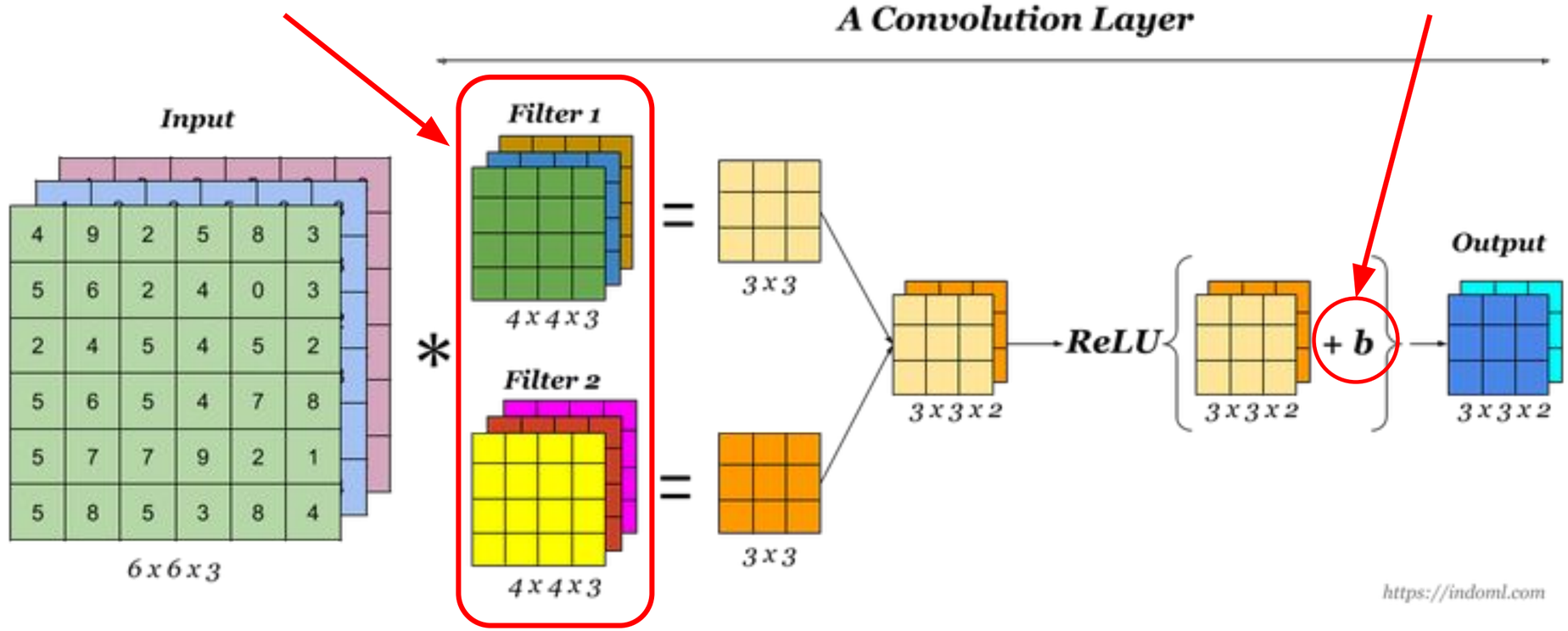
Sources:
- Coursera
- https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

5. One layer of a convolutional network

Convolution is a linear operation

This play the role of W
Remember the linear equation?

We add the bias before the activation function!

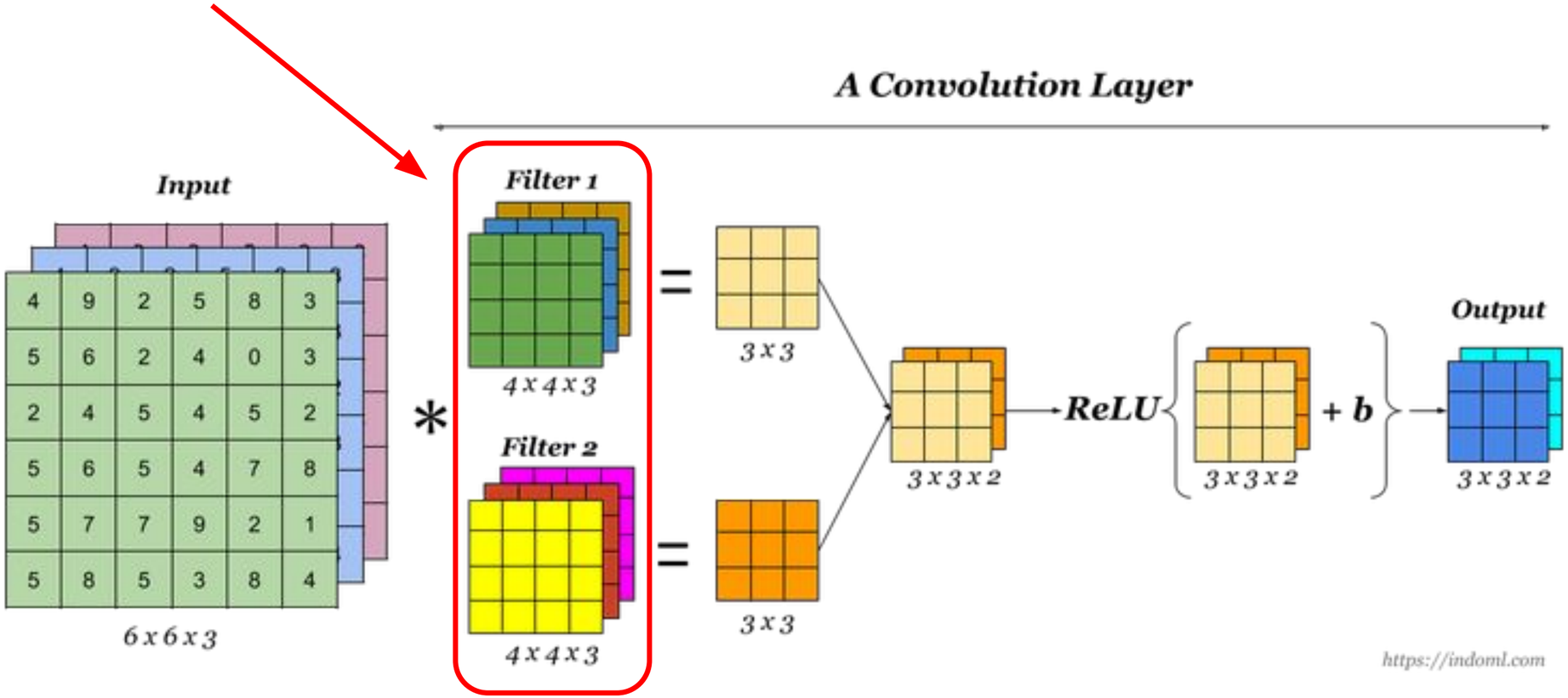


Sources:
- Coursera
- https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

5. One layer of a convolutional network

Number of filters

In this case we have 2 filters, but we can any number we want!



Sources:
- Coursera
- https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

5. One layer of a convolutional network

Number of parameters in one convolutional layer.

Let’s say we have 10 filters that are 3x3x3 in one layer of a neural network, how many parameters does that layer have?

Each filter has 3x3x3 parameters = 27 parameters per filter
27 parameters x 10 filters = 270 parameters
10 parameters as biases

Total: 280 parameters.

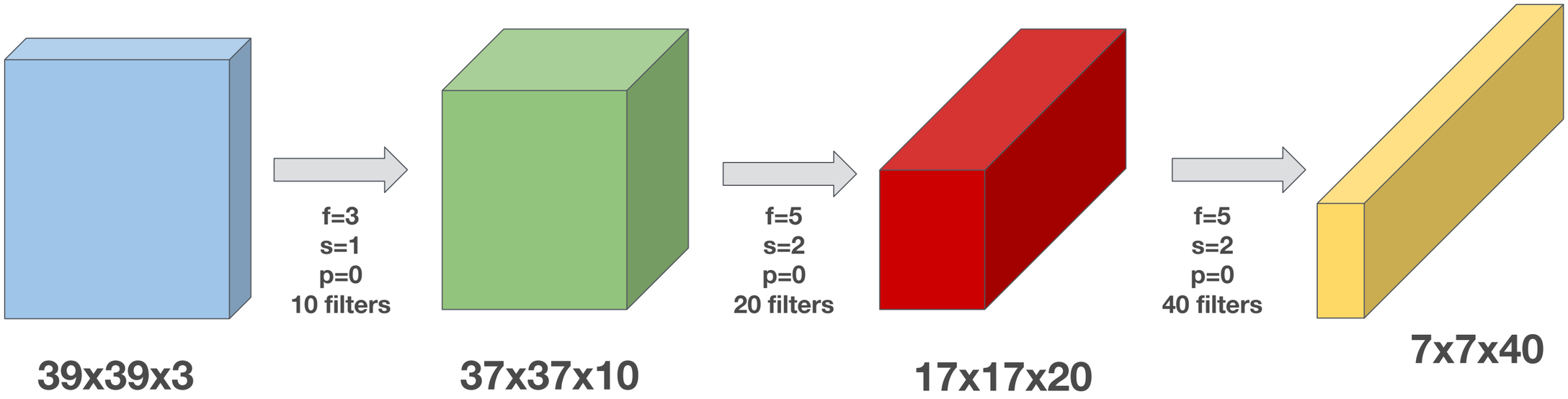
It doesn’t matter if the image is 1024 px or 64 px.

The number of parameters remains the same

Sources:
- Coursera
- https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

6. Simple convolutional network

Let’s implement a deep convolutional layer



We use these equations to calculate the output size

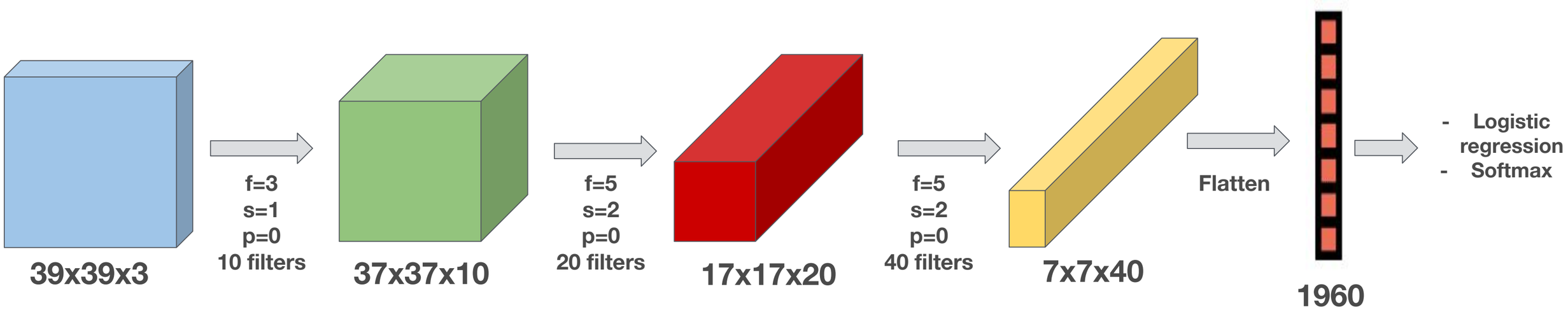
$$\text{output width} = \frac{W - F_w + 2P}{S_w} + 1$$

$$\text{output height} = \frac{H - F_h + 2P}{S_h} + 1$$

Sources:
- Coursera

6. Simple convolutional network

Unrolling the last layer



As you may see the deeper you go, the smaller is the image and the more channels it has

Sources:
- Coursera

6. Simple convolutional network

Typically, in a convolution network there are three types of layers:

- Convolution (Conv)
- Pooling (Pool)
- Fully connected (FC)

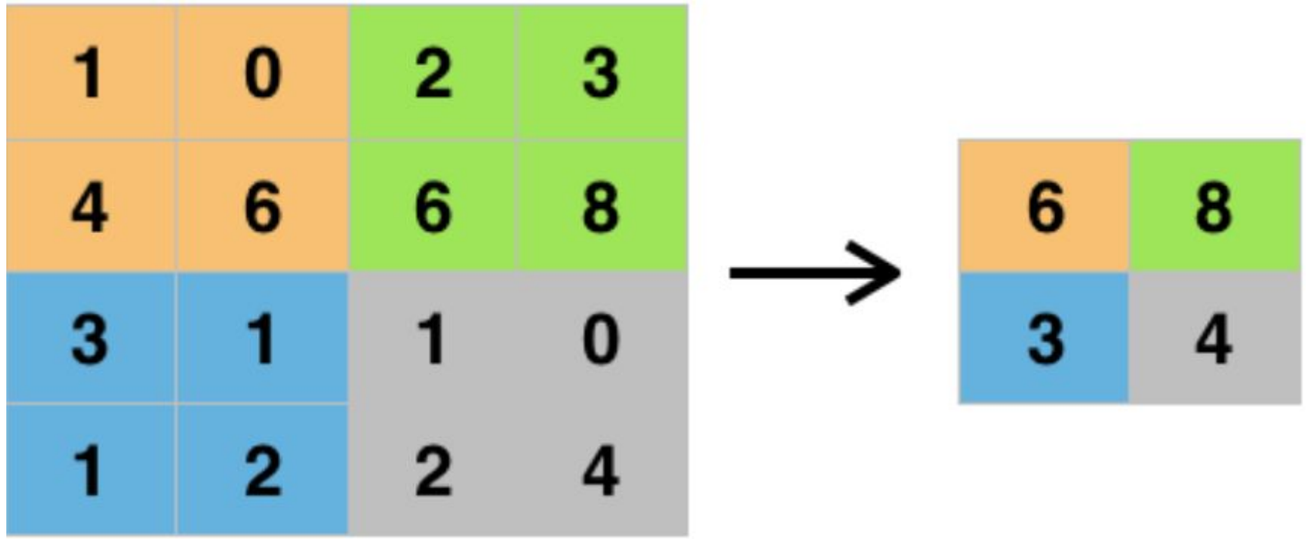
They are all combined to create an even more powerful convolutional neural network

Sources:
- Coursera

7. Pooling layers

Pooling layers are used to **reduce the size of the representation**, to speed up the computation and to make the features detected a bit more robust.

There are different types of pooling. One of the most commonly used is the max pooling:



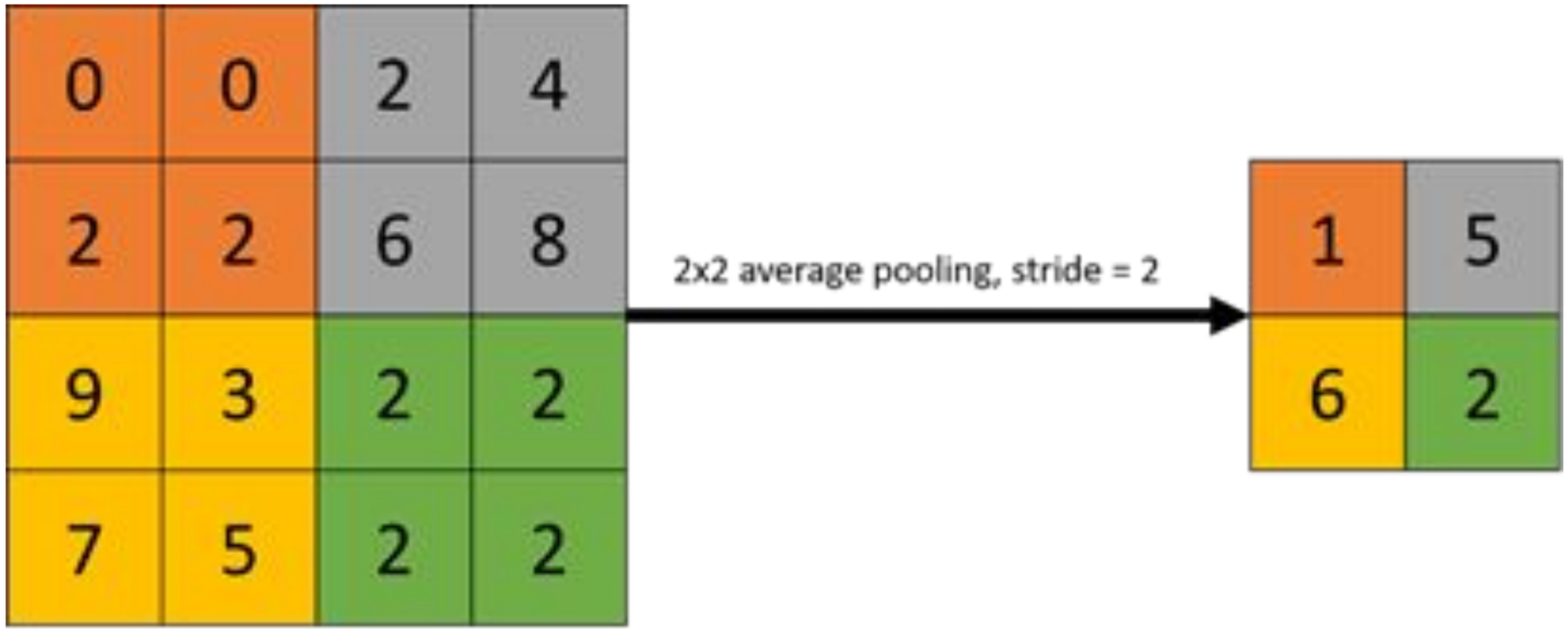
Hyperparameters:
Filter size = 2
Stride = 2

- Pooling **doesn't have parameters to learn**
- Pooling is applied to a single slice and it doesn't change the number of slices

Sources:
- Coursera
- https://deepai.org/machine-learning-glossary-and-terms/max-pooling

7. Pooling layers

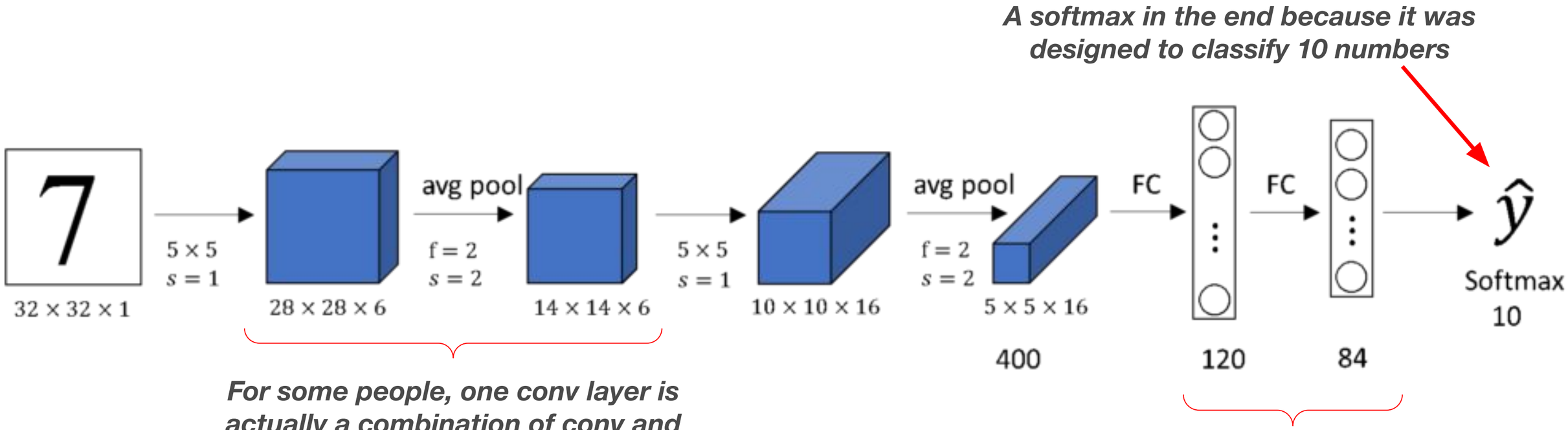
There is a less used type of pooling called **average pooling**:



Sources:
- Coursera
- https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/pooling_avg.html

8. CNN example

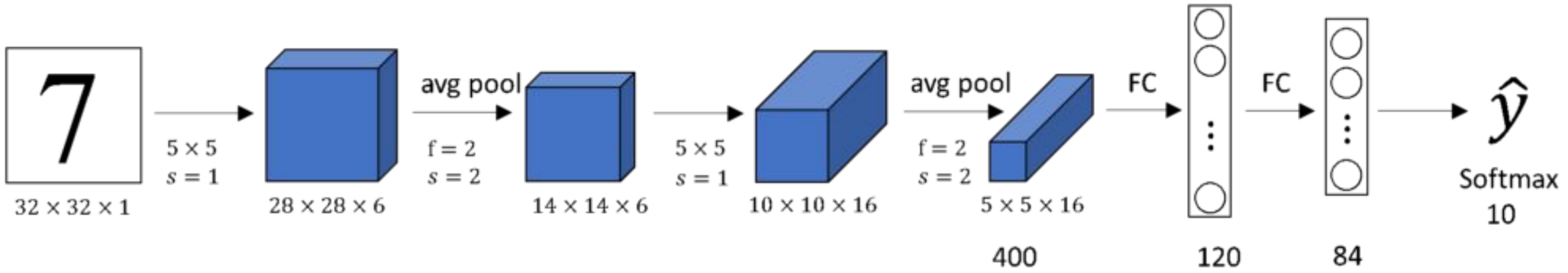
Now we know all the building blocks to create a convolutional neural network. Let's see a classic convolutional neural networks created by Yann LeCun called **LeNet-5**



Sources:
- Coursera
- http://datahacker.rs/deep-learning-lenet-5-architecture/

8. CNN example

Let's count the number of parameters



	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1.024	0
CONV1 (f=5, s=1)	(28,28,6)	4.704	156
POOL1	(14,14,6)	1.176	0
CONV2 (f=5, s=1)	(10,10,16)	1.600	2416
POOL2	(5,5,16)	400	0
FC	(120,1)	120	48.120
FC	(84,1)	84	10.164
FC + Softmax	(10,1)	10	850

How can we compute this value???

You can see that most of the parameters come from the FC layers

Sources:
- Coursera

8. CNN example

Designing a new neural network from scratch is not easy an easy task.

There are **many hyperparameters to choose (stride, filter size, padding, activation functions, number of filters, etc)**

The most common way of creating new convolutional neural networks is **taking inspiration from already built networks**.

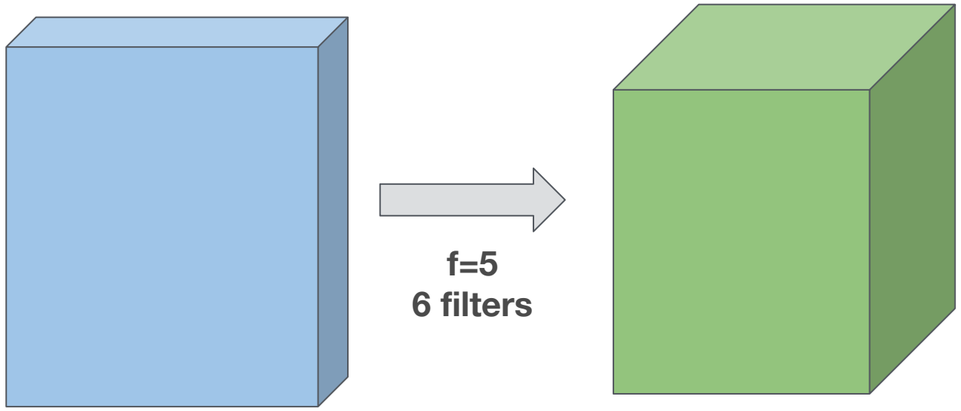
We'll talk more about this later!

Sources:
- Coursera

9. Why convolutions?

The **main advantages** of using convolutions when compared against fully connected layers are:

- Parameters sharing
- Sparsity of connections



32x32x3

3.072 parameters

28x28x6

4.704 parameters

If we used a CONV layer, the number of parameters will be (5 x 5 x 3 x 6) + 6 = 456

If we used an FC layer, the number of parameters will be 3.072 x 4.704 ~ 14 million

Sources:
- Coursera

9. Why convolutions?

Parameters sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

Sparsity of connections: In each layer, each output value depends only on a small number of input.

Translation invariance: A cat that is shifted or mirrored it is still cat in convolution neural networks

AND we still can use gradient descent to train it

Sources:
- Coursera

Let’s move to Google Colab!

Notebooks:

- *6_CNN_model.ipynb*

Sources:
- Coursera