Convolutional Neural Networks DeepLearning AI

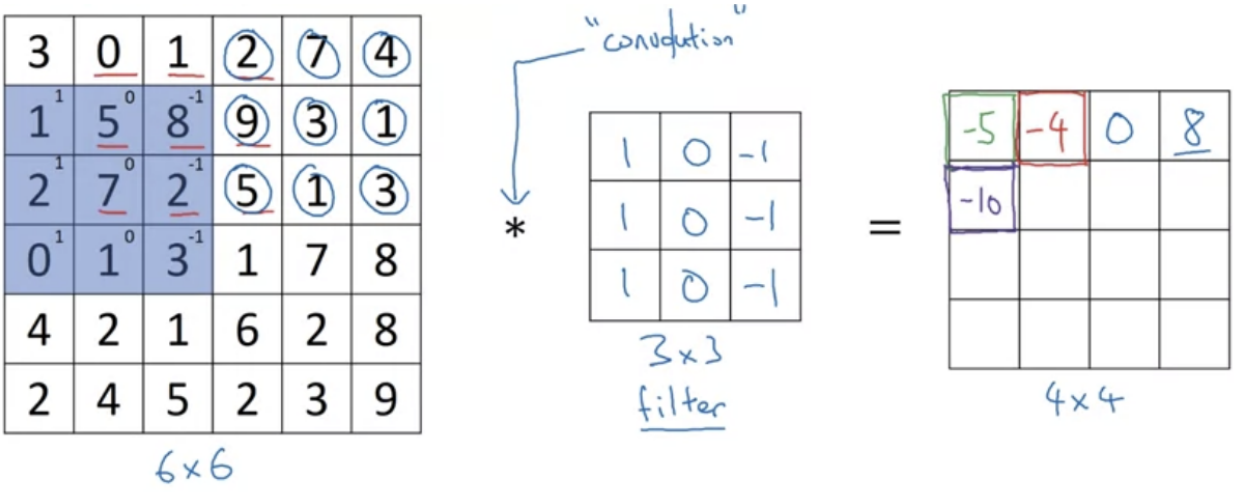
Convolutional Neural Networks

Alec Dewulf

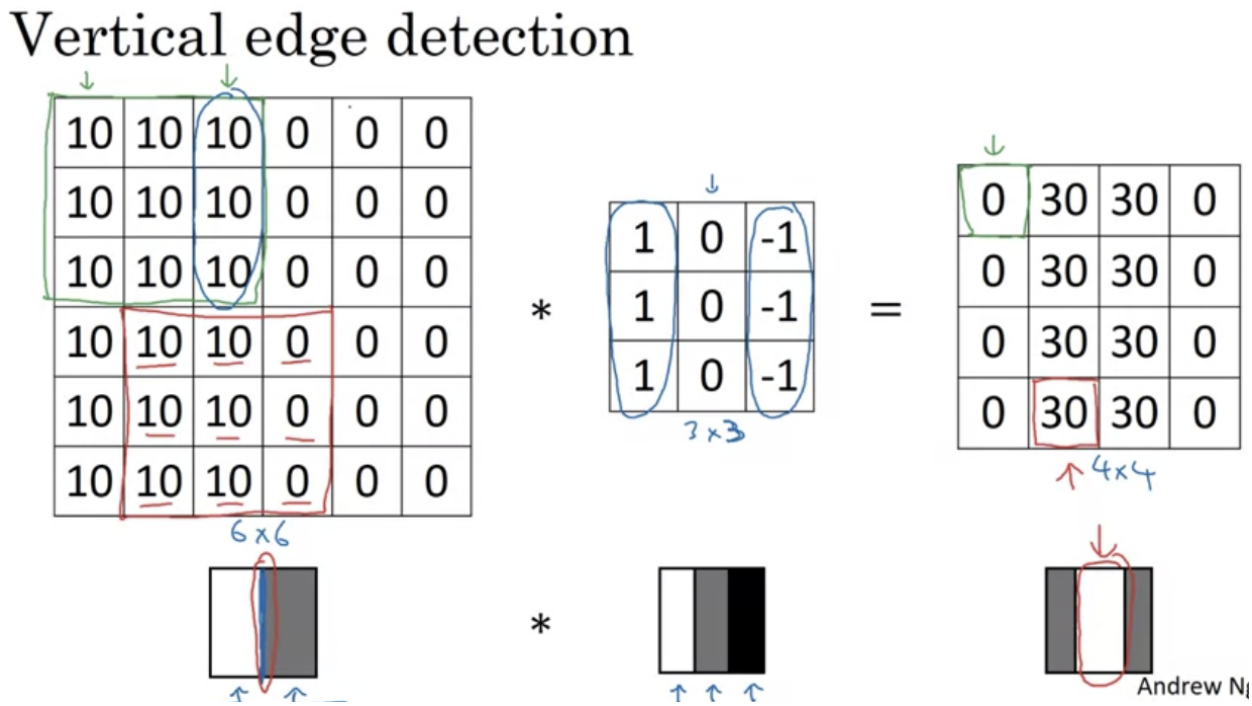
Edge Detection Example

A **filter** or **kernel** is a matrix that you use for the convolution operation.

The convolution operation works by superimposing the filter over each section in the image matrix (starting with the upper right) and then summing the element-wise products. This will give you a single number in the image.

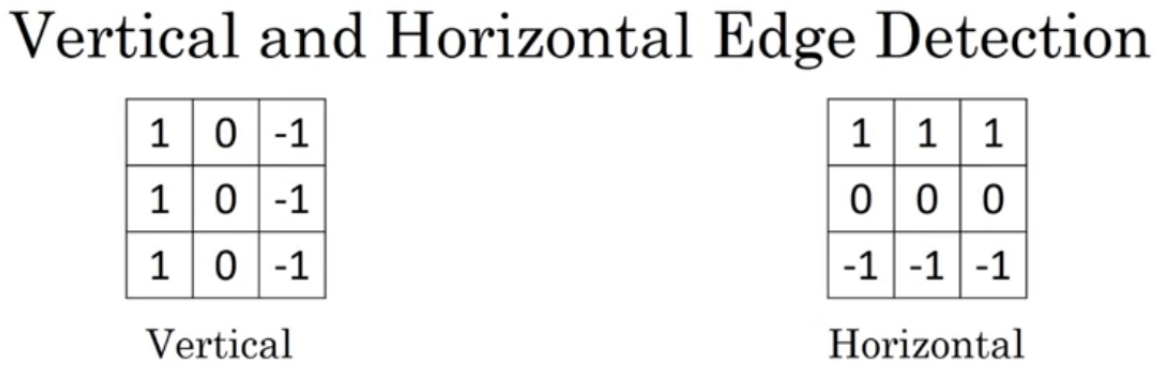


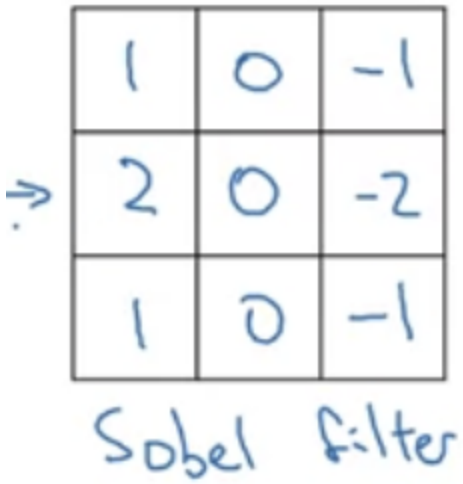
This system is for vertical edge detection.



Intuitively, this is saying that a vertical line has dark pixels on the left and light pixels on the right.

More Edge Detection





The **Sobel filter** puts more weight on the middle row and the **Schorr filter** still more weight.

In computer vision, we treat these 9 numbers as parameters that are learned by the algorithm. Neural networks can learn to detect edges more robustly this way.

Padding

If you have an nxn image and convolve that with an fxf filter, the output image will be (n-f+1)x(n-f+1).

The pixels in the corners of your image will only be used once, whereas more centered pixels will be used much more. Consequently, the information contained in this part of the image will be used less which may be problematic. As well, convoluting the image shrinks it. **Padding** is a solution to both of these problems.

Before applying the convolution, pad the image with 0s around the border. If p is the number of padded rings then our output size will be (n+2p-f+1)x(n+2p-f+1).

A **valid convolution** is one with no padding. The **same convolution** is one padded so that the output size is the same as the input size.

f the feature dimension is usually odd. This allows for symmetric padding since p = (f-1)/2 (this follows from the dimensions of the output matrix and what is required to make it same), and it has a central position which can be useful for describing positioning.

Strided Convolutions

Instead of shifting the "box" within which we take the element-wise products by 1 we shift it by some other *stride*.

If you have an nxn image that you convolve with an fxf filter with padding p and stride s, the output matrix will have dimension floor((n+2p-f)/s +1) x floor((n+2p-f)/s +1).

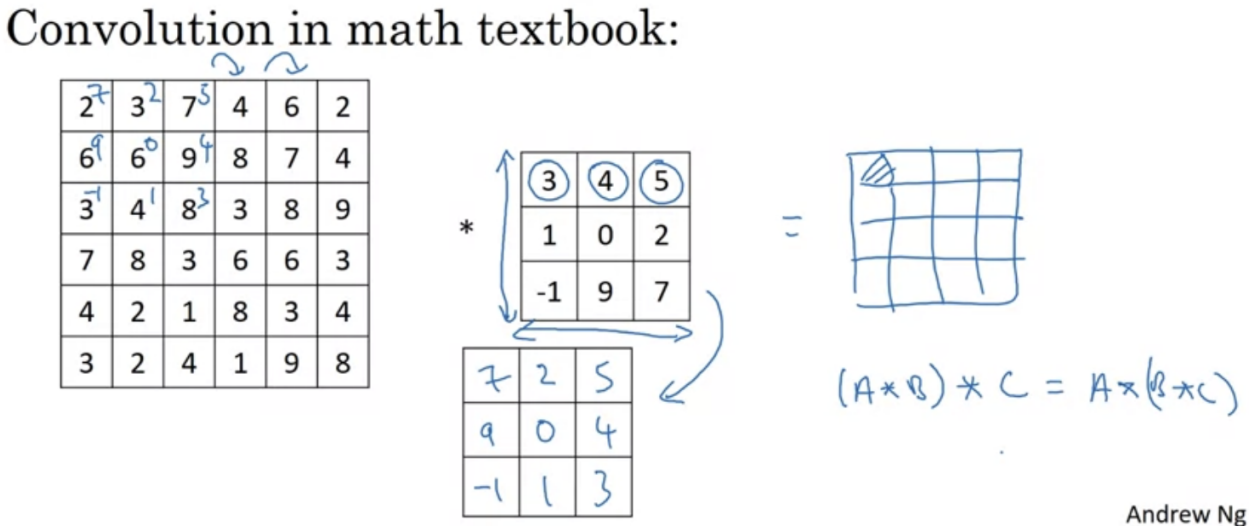
$n \times n$ image $\quad f \times f$ filter

padding p $\quad$ stride s

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

In some math textbooks the convolution is defined differently. In this definition, the filter is mirrored before it is multiplied by different parts of the matrix.

Technically, the convolution that ignores this mirroring step is called a *cross-correlation* in math. In deep learning it is okay to call it a convolution.



The mirroring operation gives the convolution operator the nice property of associativity shown above.

Originally, it is:

3	4	5
1	0	2
-1	9	7

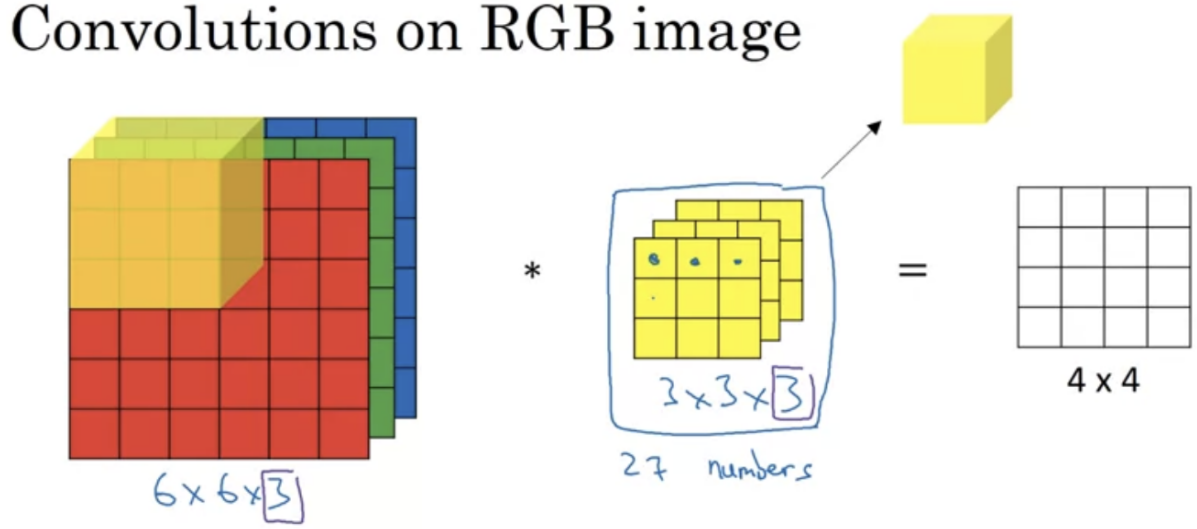
The correct filter after flipping vertically and horizontally would be:

7	9	-1
2	0	1
5	4	3

This is the correct flipping.

Convolutions over Volume

The dimension of the matrix we are convolving must match the dimension of the filter. We may therefore use a 3x3x3 image to convolve a 6x6x6 matrix. The number of channels of the image and the filter must match.

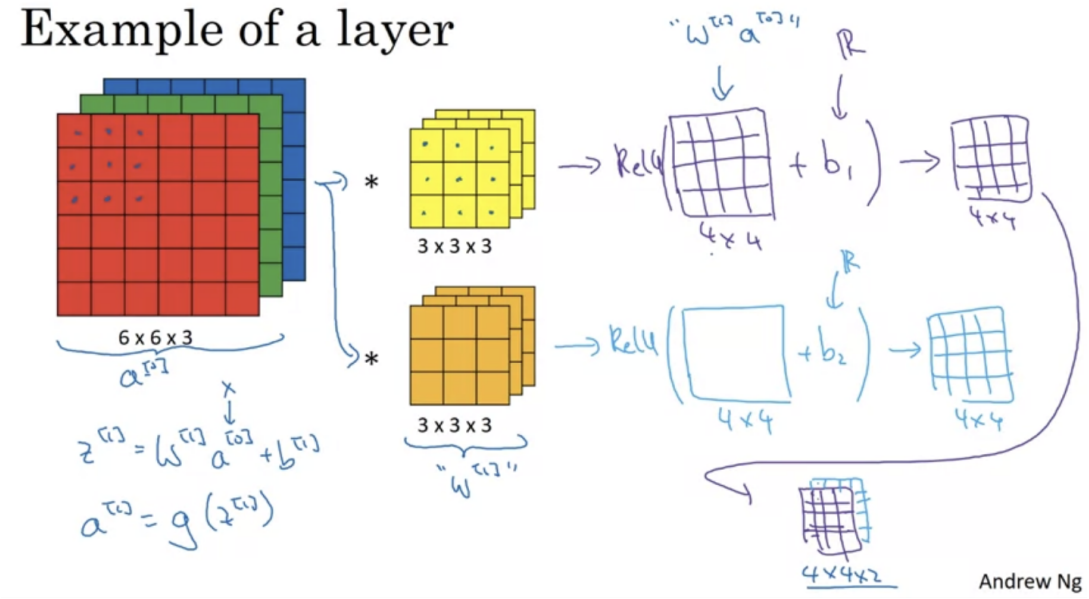


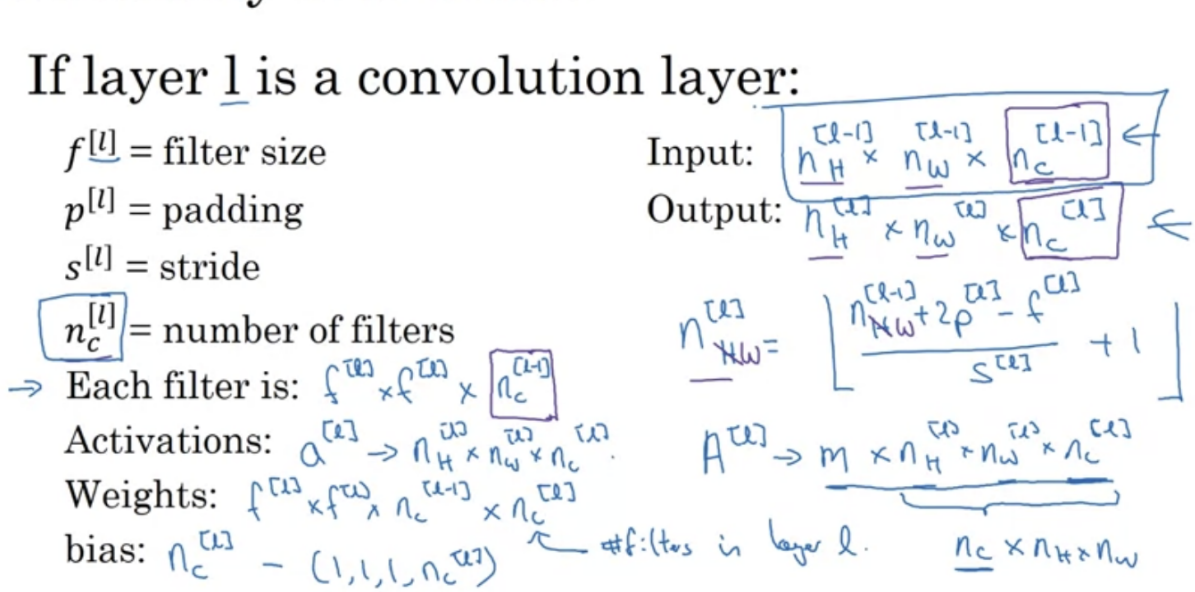
We slide the cube in 2D in the same way as before to compute the convolution.

If you want to use multiple filters to for example detect both horizontal and vertical edges, you can stack them into a 3 tensor.

One Layer of a Convolutional Neural Network

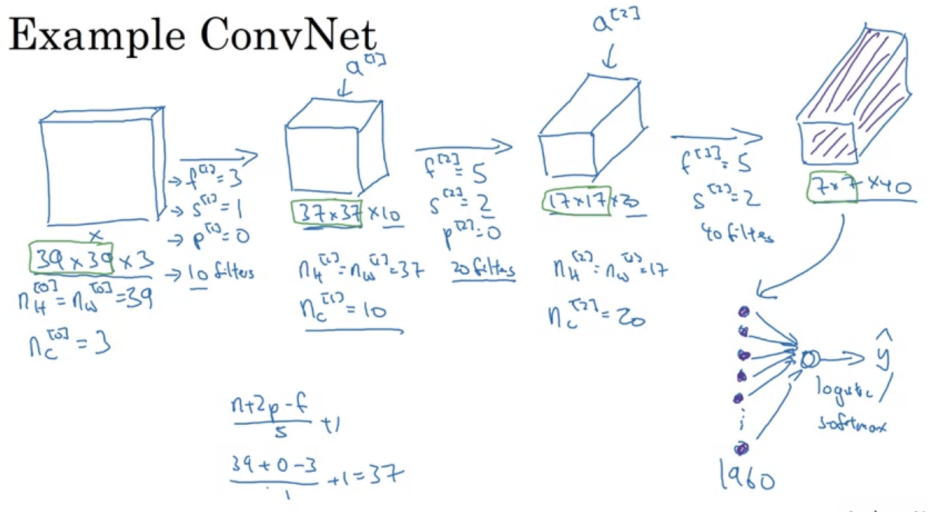
In each layer of the CNN, there will be a bias (some learned real number) and a non-linearity applied. These are applied after the convolution to each of the images.





Simple Convolutional Neural Network Example

We apply a series of convolutional layers and then unroll the resulting volume into a vector which we feed into logistic regression or softnet. Logistic regression for binary classification and softnet for recognizing different objects.



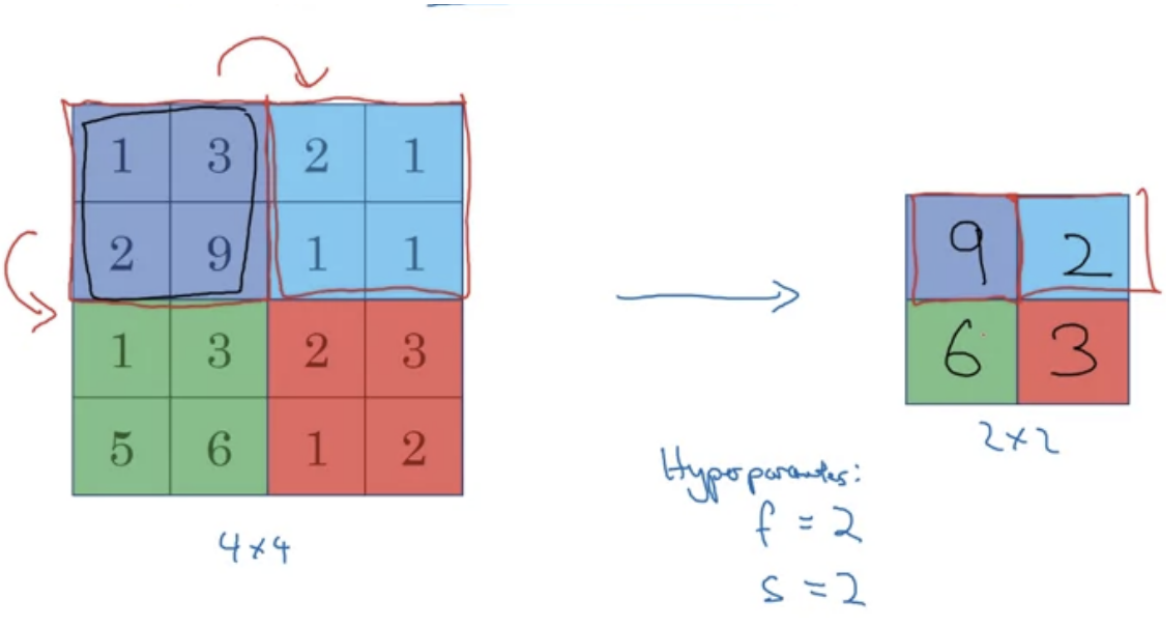
The dimensions will generally decrease, while the channels increase.

There are traditionally three types of layers:

- Convolution (conv)
- Pooling (pool)
- Fully connected (FC)

Pooling Layers

In *max pooling*, you divide an input matrix into equal sections and assign each of the sections the max value in each partition of the original matix.

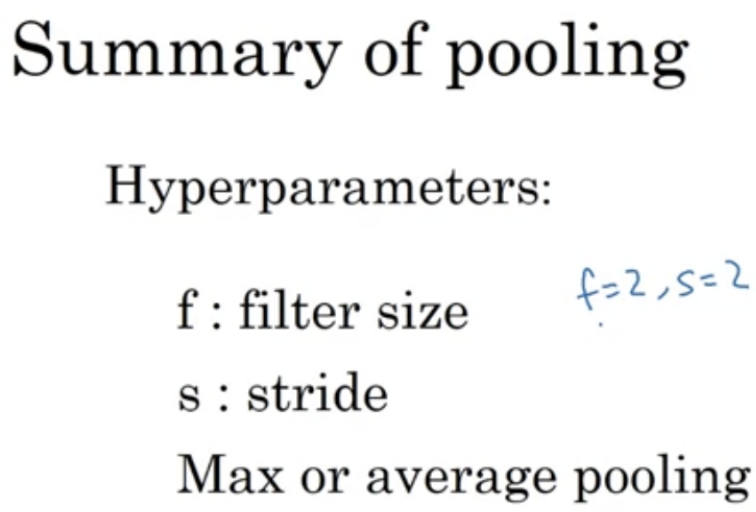


The intuition behind this is that if the feature is detected there will be a high number which should be preserved.

There is nothing to be learned. The hyperparameters are hard-coded previous to the application of max-pooling.

Max-pooling is done independently on each of the channels in an image and the resultant matrices are stacked to form a volume.

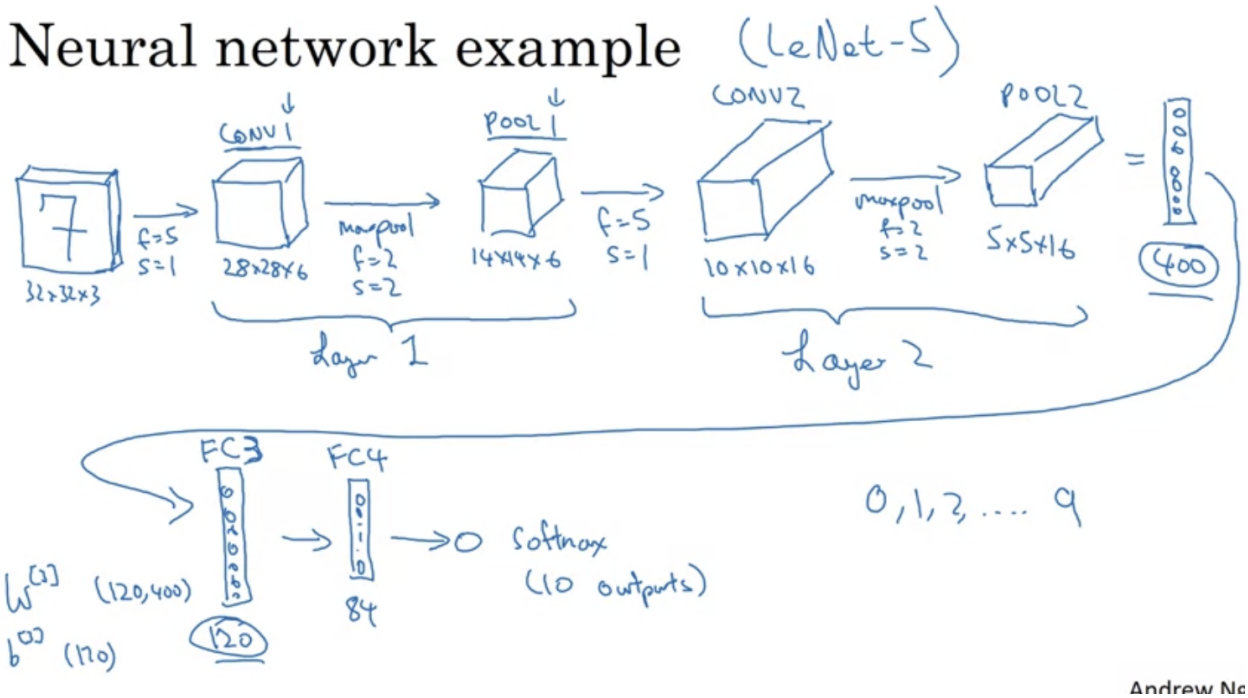
Average pooling is much less common than max pooling. It takes the average of the values in a section. It may be used to collapse your representation further into a neural network.



f=2, s=2 shrinks the height and width of the representation by a factor of two. Padding is rarely used in max pooling.

CNN Example

Usually when people report the number of layers in a net, they report the number of layers with weights. Pooling layers don't have weights because all the hyperparameters are set manually.



NOTE: Look into softmax

Generally n_h and n_w decrease as you get deeper and deeper into the neural network. n_c generally increases.

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

There are 5 typos in the above table:

1. 208 should be (5*5*3 + 1) * 8 = 608

2. 416 should be (5*5*8 + 1) * 16 = 3216

3. In the FC3, 48001 should be 400*120 + 120 = 48120, since the bias should have 120 parameters, not 1

4. Similarly, in the FC4, 10081 should be 120*84 + 84 (not 1) = 10164

(Here, the bias is for the fully connected layer. In fully connected layers, there will be one bias for each neuron, so the bias become In FC3 there were 120 neurons so 120 biases.)

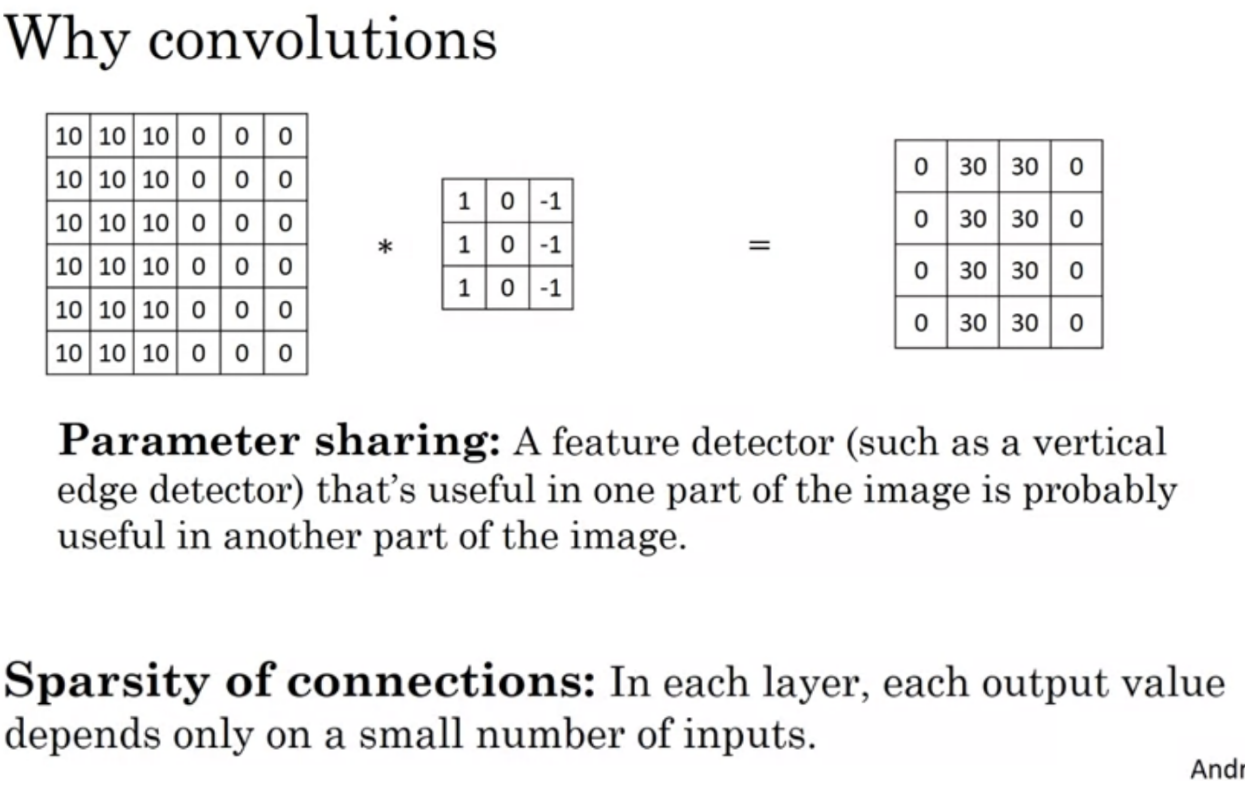
5. Finally, in the softmax, 841 should be 84*10 + 10 = 850

Why Convolutions?

The two main advantages of convolutional layers are:

- Parameter sharing
- Sparsity of connections

You can have multiple filters in each layer.



Many of the values in the input matrix are connected to a single value in the output matrix. All of the values encompassed by the filter map to a single output value.

CNNs do a good job of capturing *translational invariance.* This is the idea that the object in an image is unchanged and should still be able to be detected if it is translated.