deeplearning.ai

Convolutional Neural Networks

Computer vision

Computer Vision Problems

You have a picture and you want it painted in a different style.

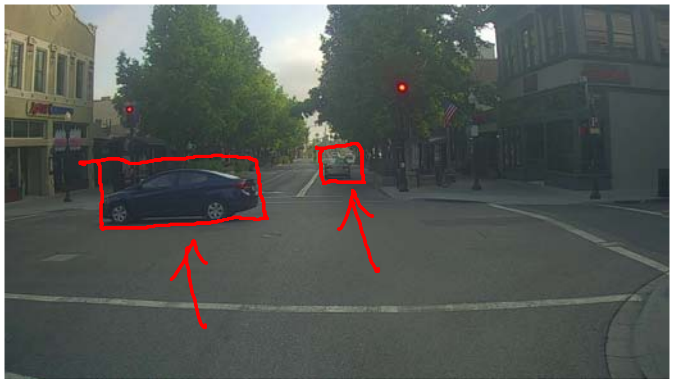
Image Classification

64x64

→ Cat? (0/1)

Neural Style Transfer

Object detection



: we not just need to classify them but also need to know where in picture these objects are so that we can draw box around them to detect them.

Andrew Ng

Deep Learning on large images

one of the prob in comp vision is that img can get really big

$\longrightarrow$ Cat? (0/1)

64x64 x 3

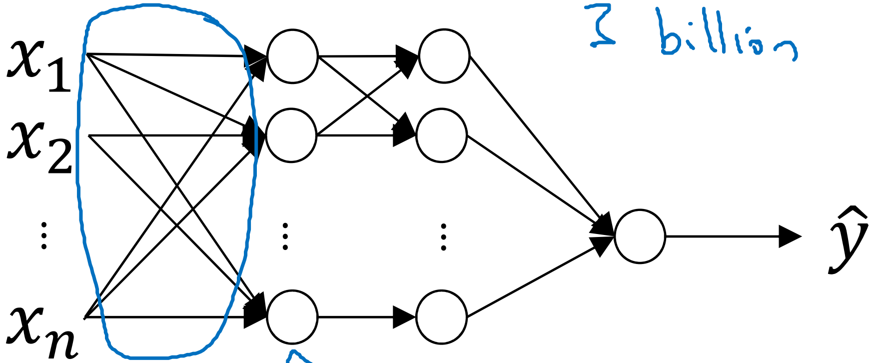
12288

1000x1000x3
= 3 million

$X \in \mathbb{R}^{3M}$

$W^{[1]}$ (1000, 3M)

3 billion



3M

1000

Andrew Ng

With 3 bill data is difficult to prevent a NN from overfittin and also the computational requirements and memory requirements to train a NN with 3 billin is just a bit infeasable.
But for comp vision u dont want to be stuck using little img u want to use large img
To do that you need to better implement the convolution operation, which is one of the fundamental building blocks of conv NN

deeplearning.ai

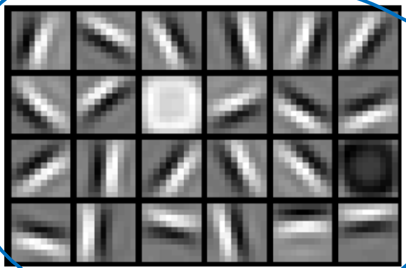
Convolutional Neural Networks

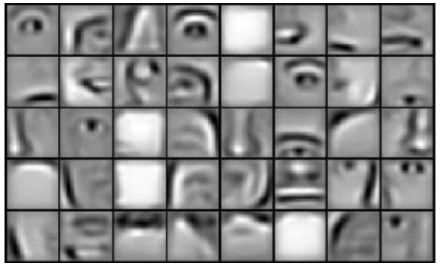
Edge detection example

The convolution operation is one of the fundamental building blocks of a convolutional neural network. Using edge detection as the motivating example in this video, you will see how the convolution operation works.

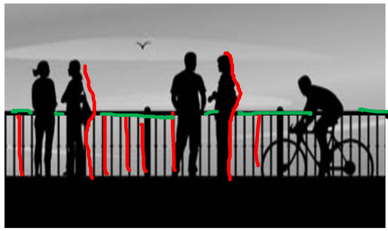
Computer Vision Problem

In previous videos, I have talked about how the early layers of the neural network might detect edges and then the some later layers might detect cause of objects and then even later layers may detect cause of complete objects like people's faces in this case.





In this video we gone see how you can detect edges in an img.
Given a picture like that



vertical edges

horizontal edges

Andrew Ng

Vertical edge detection

$3\times1 + 1\times1 + 2\times1 + 0\times0 + 5\times0 + 7\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

Here is a 6x6 grey scale img

"convolution" convolve with that operation

$*$

1	0	-1
1	0	-1
1	0	-1

3x3

→ filter

kernel

$=$

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4x4

python: conv-forward

tensoflow: tf,nn,convfd

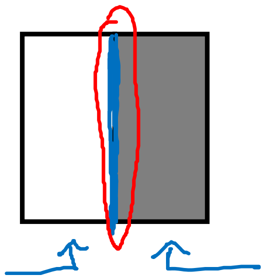
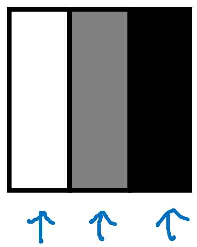
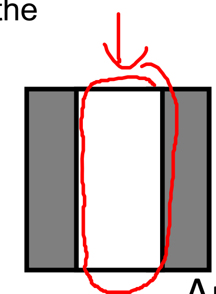
keras; conv2D

Why is it doing edge detection.

Vertical edge detection

$$
\underset{6 \times 6}{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}} * \underset{3 \times 3}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{4 \times 4}{\begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}
$$

this white part correspond to having detected the midle edge

 *  

So we did vertical edge detection which is one of the building bolcks of CNN

U can detect edges using that convolution operation

Andrew Ng

deeplearning.ai

Convolutional Neural Networks

More edge detection

Vertical edge detection examples

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

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

Andrew Ng

Vertical and Horizontal Edge Detection

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

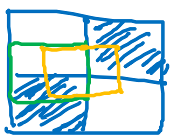
6 x 6

*

1	1	1
0	0	0
-1	-1	-1

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0



Andrew Ng

Learning to detect edges

1	0	-1
1	0	-1
1	0	-1

the advantage of this is that it puts some more weight to the center row

1	0	-1
2	0	-2
1	0	-1

Sobel filter

this has other properties

3	0	-3
10	0	-10
3	0	-3

Scharr filter

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

convolution

$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3x3

U can learn these by putting them as parameters and use backprop

=

45°
70°
73°

it can maybe learn to detect edges that are of different degrees

Andrew Ng

The idea that you can put these parameters to be learned is one of the most powerful idea in computer vision
later we gone see how u use backprop to learn these 9 parameters

In order to build deep neural networks one modification to the basic convolutional operation that you need to really use is padding

Convolutional Neural Networks

deeplearning.ai

Padding

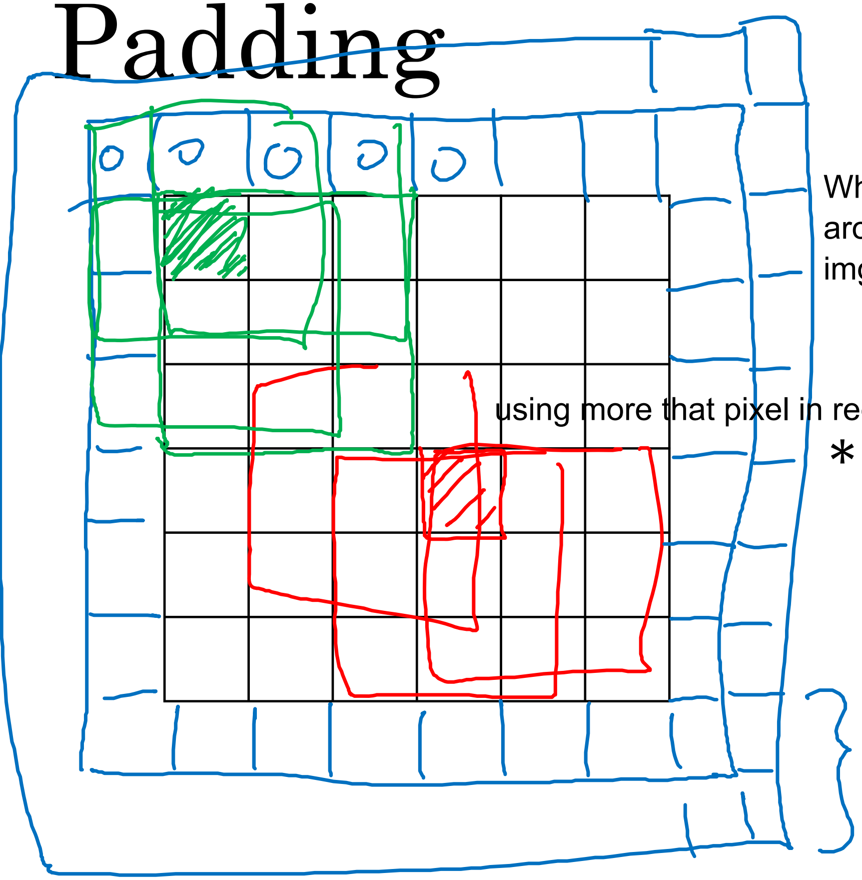
These are the 2 drowbacks:

Output size is smaller if you dont apply padding

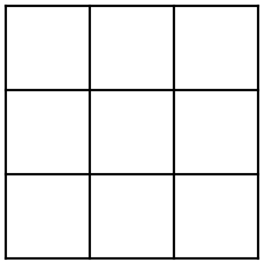
Padding

- shrinky output
- throw away info from edge

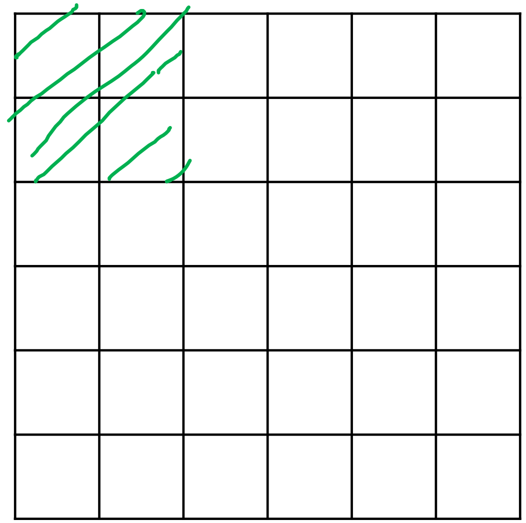
What u do for this is pad img with one boarder around the edges so instead of 6x6 u have 8x8 img and with convolut operation u get a 6x6 output



$*$



$=$



$p=2$

3×3

$f\times f$

6×6

~~4x4~~

2 downsides;
- img shrinks
. pixels in corners are used much less in output

$6\times 6 \rightarrow 8\times 8$

$n\times n$

this will be the dimention of output

$n-f+1 \times n-f+1$

$6-3+1=4$

$n+2p-f+1 \times n+2p-f+1$

$6+2-3+1 \times \_\_ = 6\times 6$

$p = \text{padding} = 1$

Andrew Ng

by convention u pad with 0 and p is the padding amout and in this case is 1

So u surround that img with these layer of 0.

Valid and Same convolutions

"Valid": → no padding

$n \times n \;*\; f \times f \;\longrightarrow\; n-f+1 \times n-f+1$

$6 \times 6 \;*\; 3 \times 3 \;\longrightarrow\; 4 \times 4$

"Same": Pad so that output size is the same as the input size.

$n+2p-f+1 \times n+2p-f+1$

$\cancel{n}+2p-f+1 = \cancel{n} \Rightarrow \boxed{p = \frac{f-1}{2}}$

This is how choose pad in order not change size.

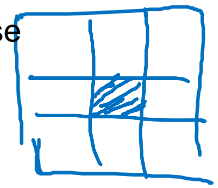
$\boxed{f \text{ is usually odd}}$

1×1

3×3

5×5

7×7



$3 \times 3 \quad p = \frac{3-1}{2} = 1$

$5 \times 5 \quad p = 2$

$f = 5$

Andrew Ng

deeplearning.ai

Convolutional Neural Networks

Strided convolutions

Strided conv is another building block of the convolution as using convolution NN
Say we want to convolve that 7x7 with that 3x3 filter, instead of doing the usual way we gone do it with stride of 2, so we gone move box not just as before by 1 but by 2 this time.

Strided convolution

the filter must lay entirely within the img or img plus padding
thats a convention

2 3	3 4	7 3	4 4	6 3	2 4	9 4
6 1	6 0	9 1	8 0	7 1	4 0	3 2
3 -3	4 4	8 -3	3 4	8 -3	9 4	7 4
7 1	8 0	3 1	6 0	6 1	3 0	4 2
4 -3	2 4	1 -3	8 4	3 -3	4 4	6 4
3 1	2 0	4 1	1 0	9 1	8 0	3 2
0 -1	1 0	3 -1	9 0	2 -1	1 0	4 3

$\underline{7\times7}$

*

3	4	4
1	0	2
-1	0	3

$\underline{3\times3}$

=

91	100	83
69	91	127
44	72	74

$\underline{3\times3}$

stride = 2

$\lfloor z \rfloor = \text{floor}(z)$

$n \times n \quad * \quad f \times f$

padding p $\quad$ stride s

$s = 2$

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

$$\frac{7+0-3}{2} + 1 = \frac{4}{2} + 1 = 3$$

this is why we end up
with this 3x3 matrix

Andrew Ng

what if the fraction is not an integer??
in that case we gone round to floor.

Summary of convolutions

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

padding p stride s

Output size:

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

Technical note on cross-correlation vs. convolution

Just dont confuse the notations of the convolution with the cross-correlation.

Convolution in math textbook:

2^{7}	3^{2}	7^{5}	4	6	2
6^{9}	6^{0}	9^{4}	8	7	4
3^{-1}	4^{1}	8^{3}	3	8	9
7	8	3	6	6	3
4	2	1	8	3	4
3	2	4	1	9	8

*

3	4	5
1	0	2
-1	9	7

=

7	2	5
9	0	4
-1	1	3

In math text books u flip first the filter horizontaly and vertically

$$(A * B) * C = A * (B * C)$$

we do not do it in machine learning

Andrew Ng

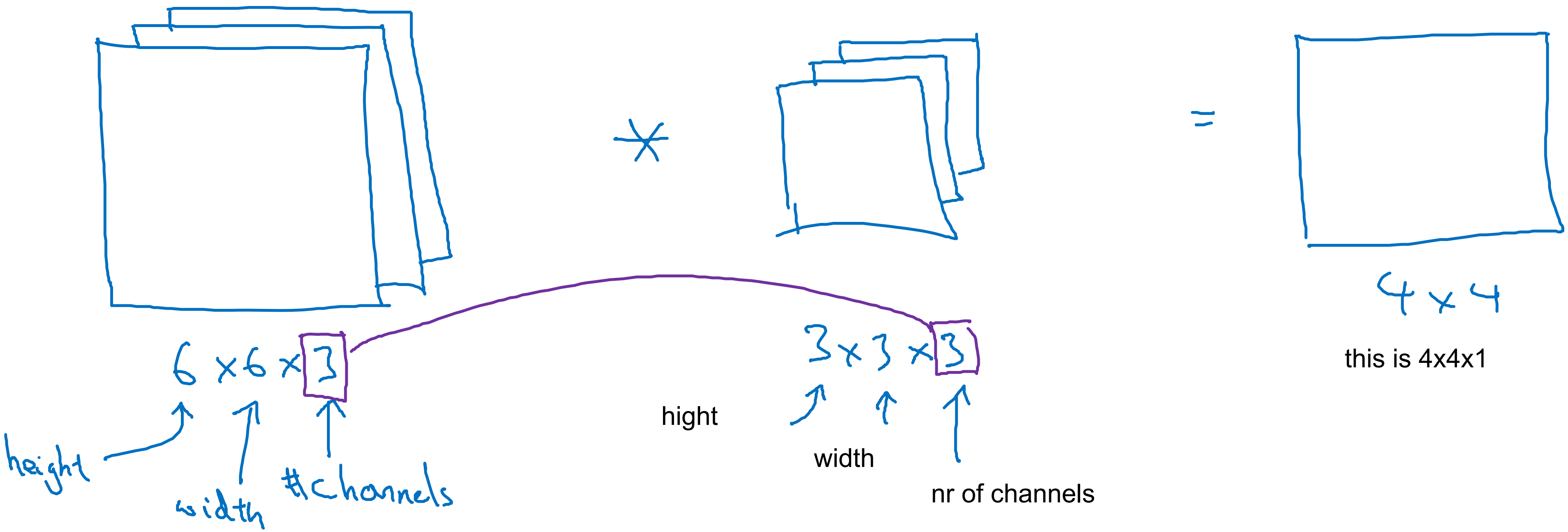
deeplearning.ai

Convolutional Neural Networks

Convolutions over volumes

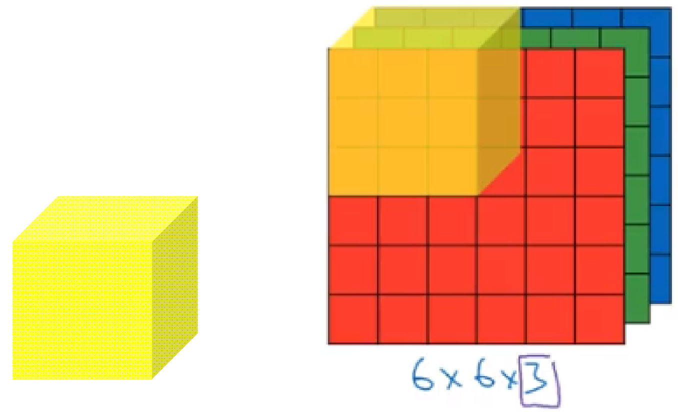
Now we will see how to implement convolutions over 3D img.

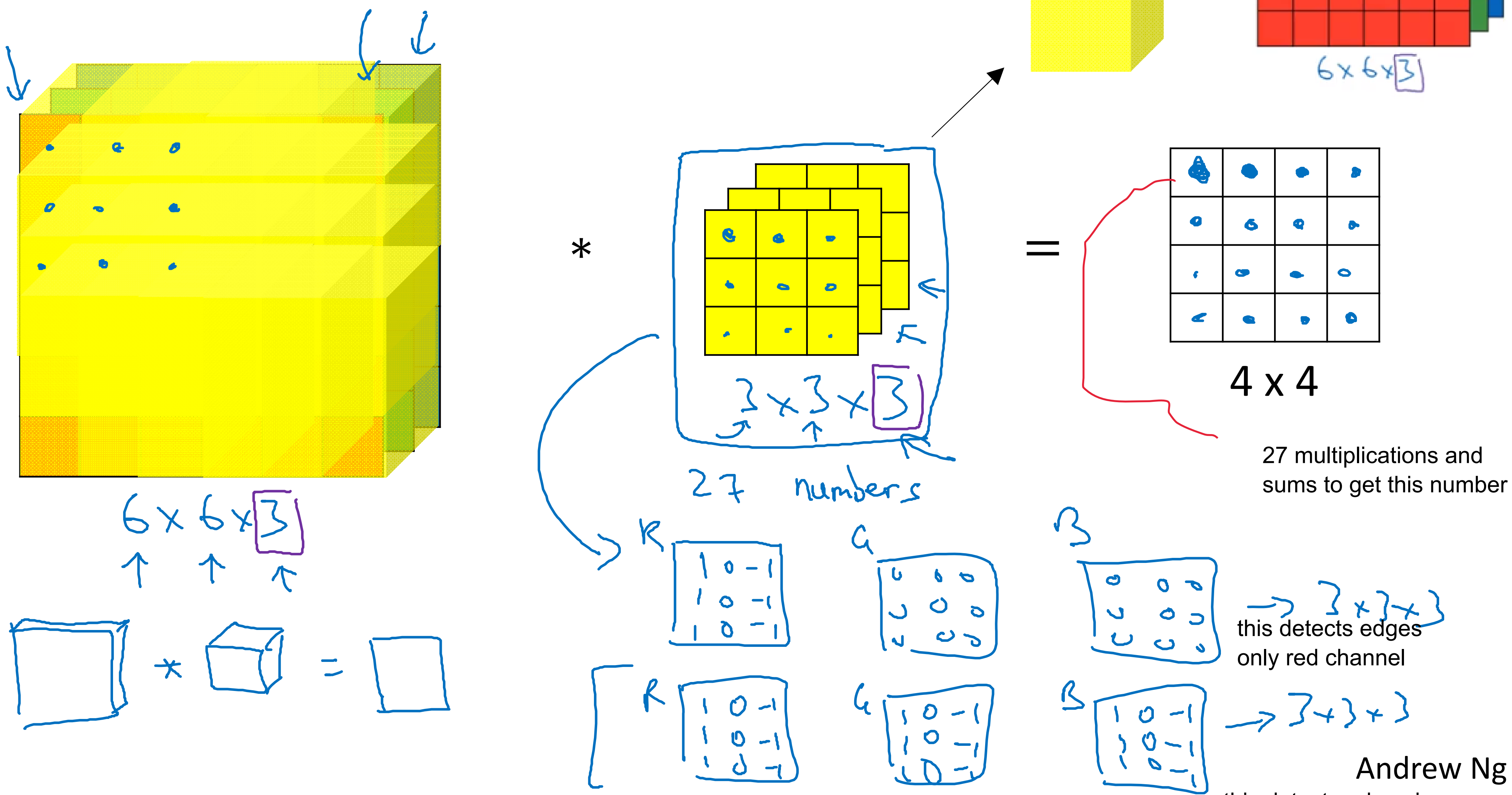
Convolutions on RGB images



To detect edges in this rgb img u convolve with a 3x3x3 filter

Convolutions on RGB image





27 multiplications and sums to get this number

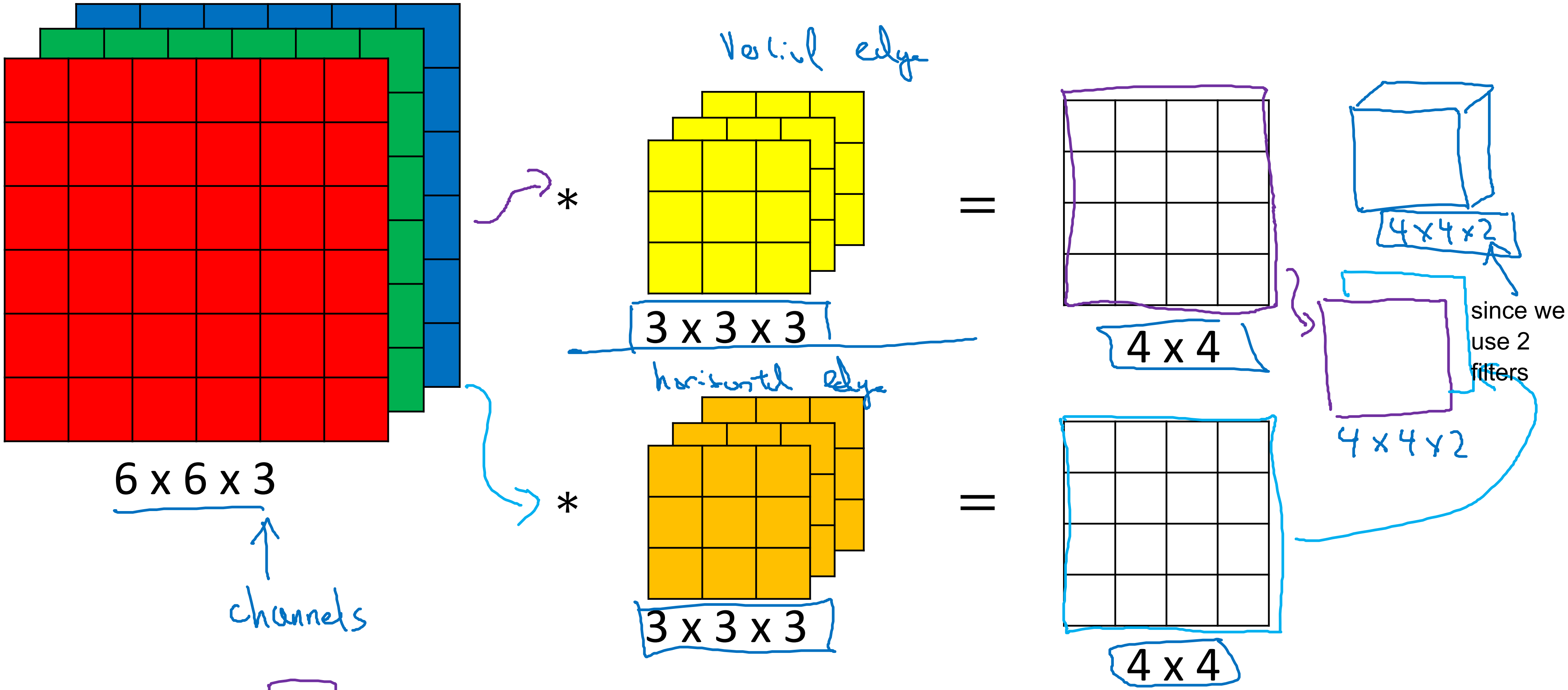
this detects edges only red channel

this detects edges in r g b channels

I is normal sometime to have a filter that looks only at green channel or red channel or blu channel.

What if we want to detect vertical and horizontal edges, so not just one at a time

Multiple filters



Summary: $n \times n \times \boxed{n_c}$ $*$ $f \times f \times \boxed{n_c}$ $\longrightarrow$ $n-f+1 \times n-f+1 \times n_c'$

$6 \times 6 \times 3$ $\quad$ $3 \times 3 \times 3$ $\quad$ $4 \times 4 \times 2$ ← # filters

Andrew Ng

We are now ready to see how to build a one layer of a convolution NN. Thats go through an example

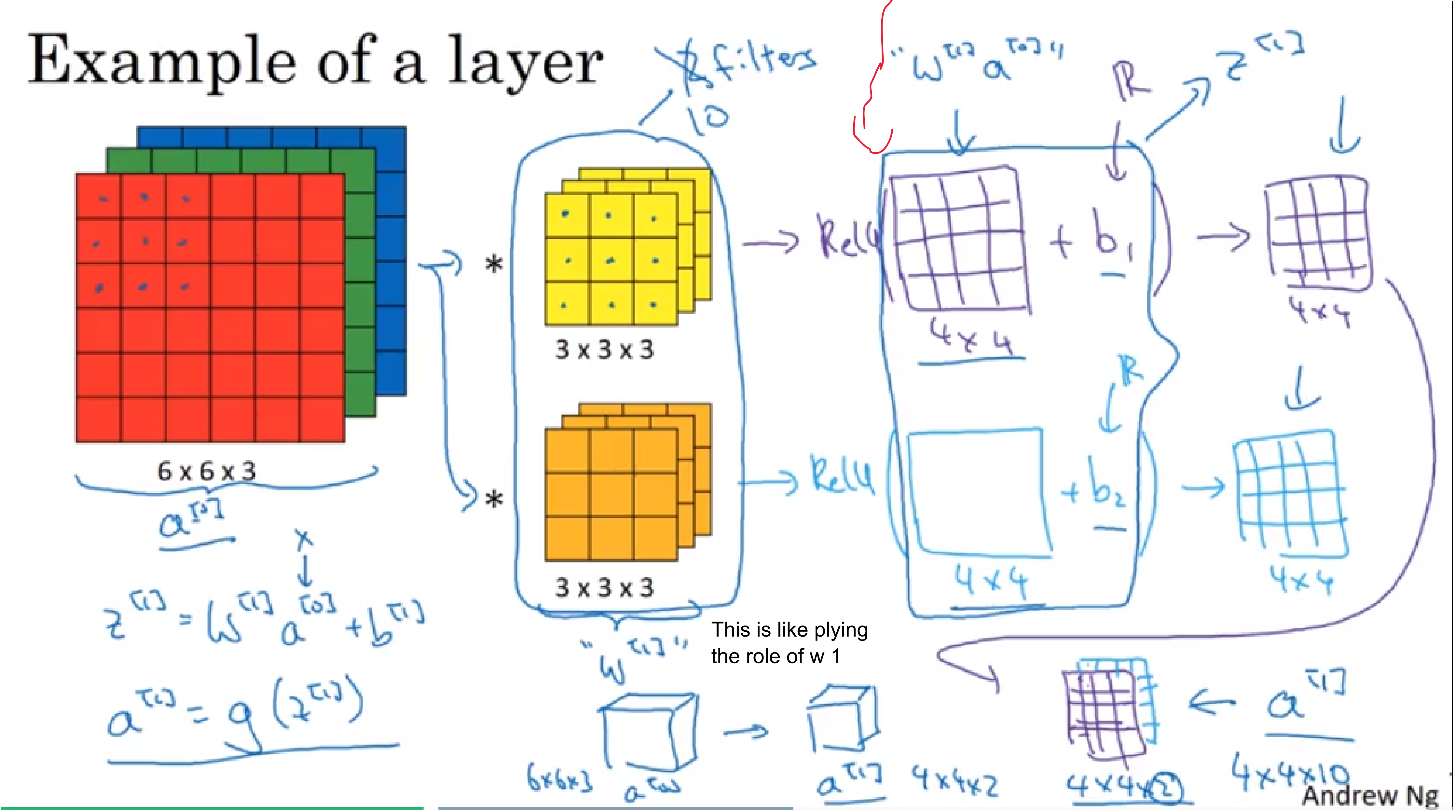
deeplearning.ai

Convolutional Neural Networks

One layer of a convolutional network

To turn this into a CNN layer we add bias and apply ReLu and we get that output

Example of a layer

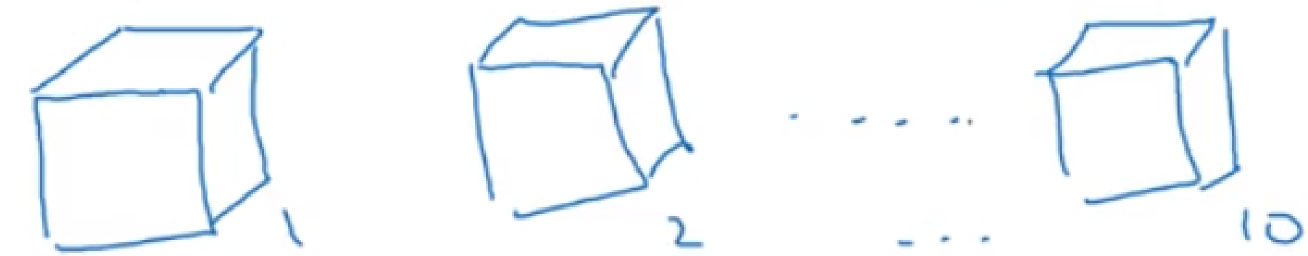


This is like plying the role of w 1

THIS IS ONE LAYER OF THE CONVOLUTION NN

Number of parameters in one layer

If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?



$3 \times 3 \times 3$

27 parameters.

+ bias

→ 28 parameters.

280 parameters.

Andrew Ng

No matter what the size of the img is the size of the parameters reamin fixed 280
and u can use these 10 filters to detect features, vertical edges horizontal ones maybe ofther features

Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$\boxed{n_c^{[l]}}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times \boxed{n_c^{[l-1]}}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$ ← #filters in layer l.

bias: $n_c^{[l]}$ – $(1,1,1,n_c^{[l]})$

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times \boxed{n_c^{[l-1]}}$ ←

Output: $n_H^{[l]} \times n_W^{[l]} \times \boxed{n_c^{[l]}}$ ←

$$n_{HW}^{[l]} = \left\lfloor \frac{n_{HW}^{[l-1]}+2p^{[l]}-f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

this means take floor

$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$n_c \times n_H \times n_W$

Andrew Ng

In the last video, you saw the building blocks of a single layer, of a single convolution layer in the ConvNet. Now let's go through a concrete example of a deep convolutional neural network.

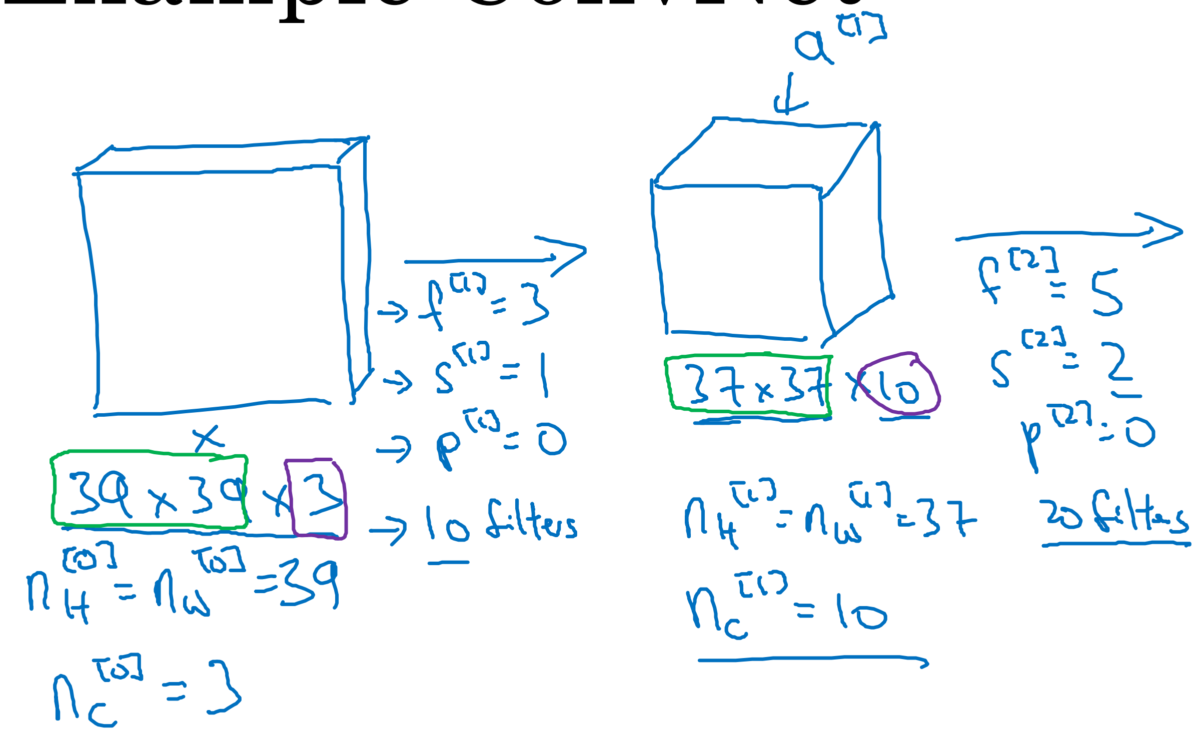
deeplearning.ai

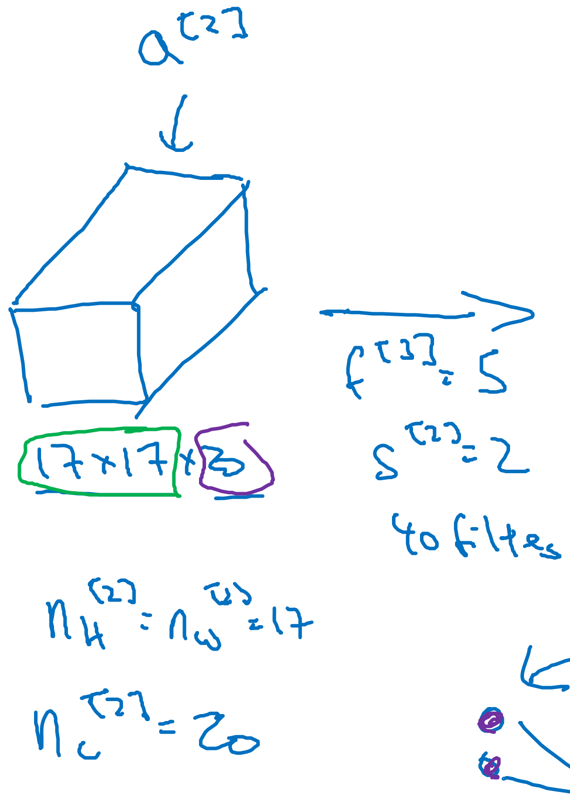
Convolutional Neural Networks

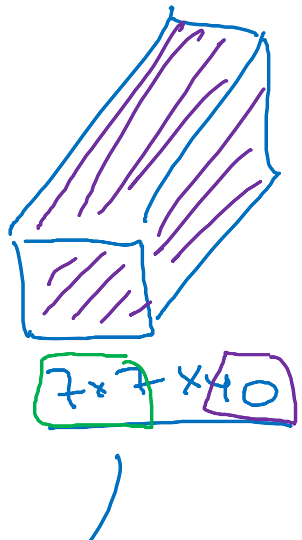
A simple convolution network example

Example ConvNet

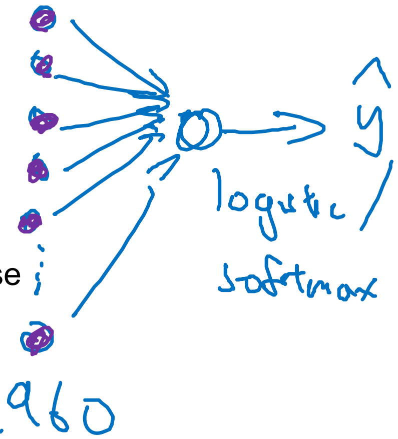
So what we have done is take that img and transform into this set fo features of 7x7x40







we take these features, unrow them, flatten.



$$\frac{n+2p-f}{s}+1$$

$$\frac{39+0-3}{1}+1=37$$

A lot of the work in building CNN is to select hyperparameter like these, whats the filter size whats the stride, the pading, how many filters to use This and next week we will give some suggestions

We feed this to a logistic regression or a softmax unit, depending on whether u trying to recognize cat, no cat or any other object

Andrew Ng

Types of layer in a convolutional network:

- Convolution (CONV)
- Pooling (POOL)
- Fully connected (FC)

We have talked only abut the conv layer
we will see later the pooling layer and the fully connected layer

It is possible to build without the pool and fc
most architecture NN will have a few pool layers and FC layers

Andrew Ng

Other than convolutional layers, ConvNets often also use pooling layers to reduce the size of the representation, to speed the computation, as well as make some of the features that detects a bit more robust.

deeplearning.ai

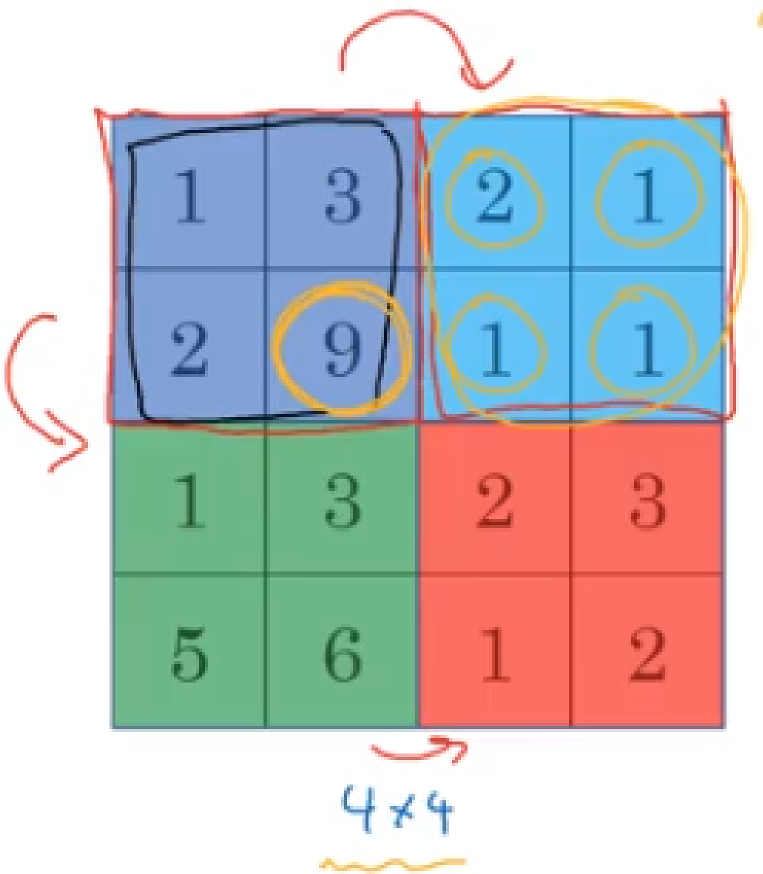
Convolutional Neural Networks

Pooling layers

Pooling layer: Max pooling

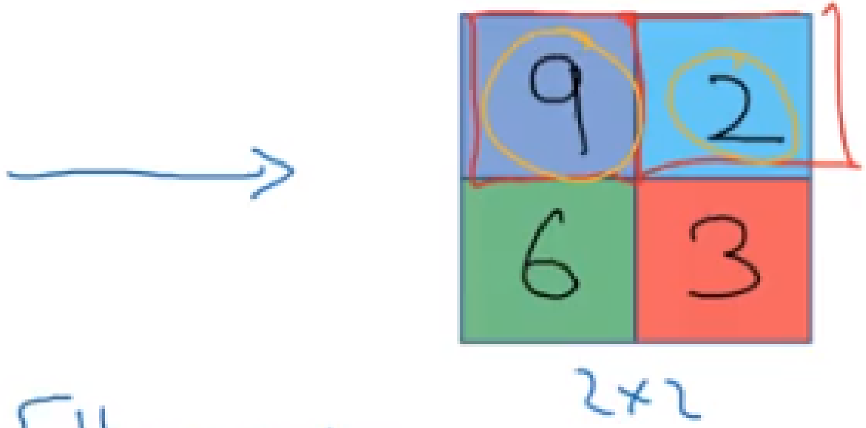
Suppose we want to apply a type of pooling called max pooling

Intuition is not that great but a lot of people use it as it works well, he does not know very well wether that is the real intuition.



Take input an break into four regions

this will be the output



in the output, each will be the max from the four regions in the input

Hyperparameters:

$f = 2$

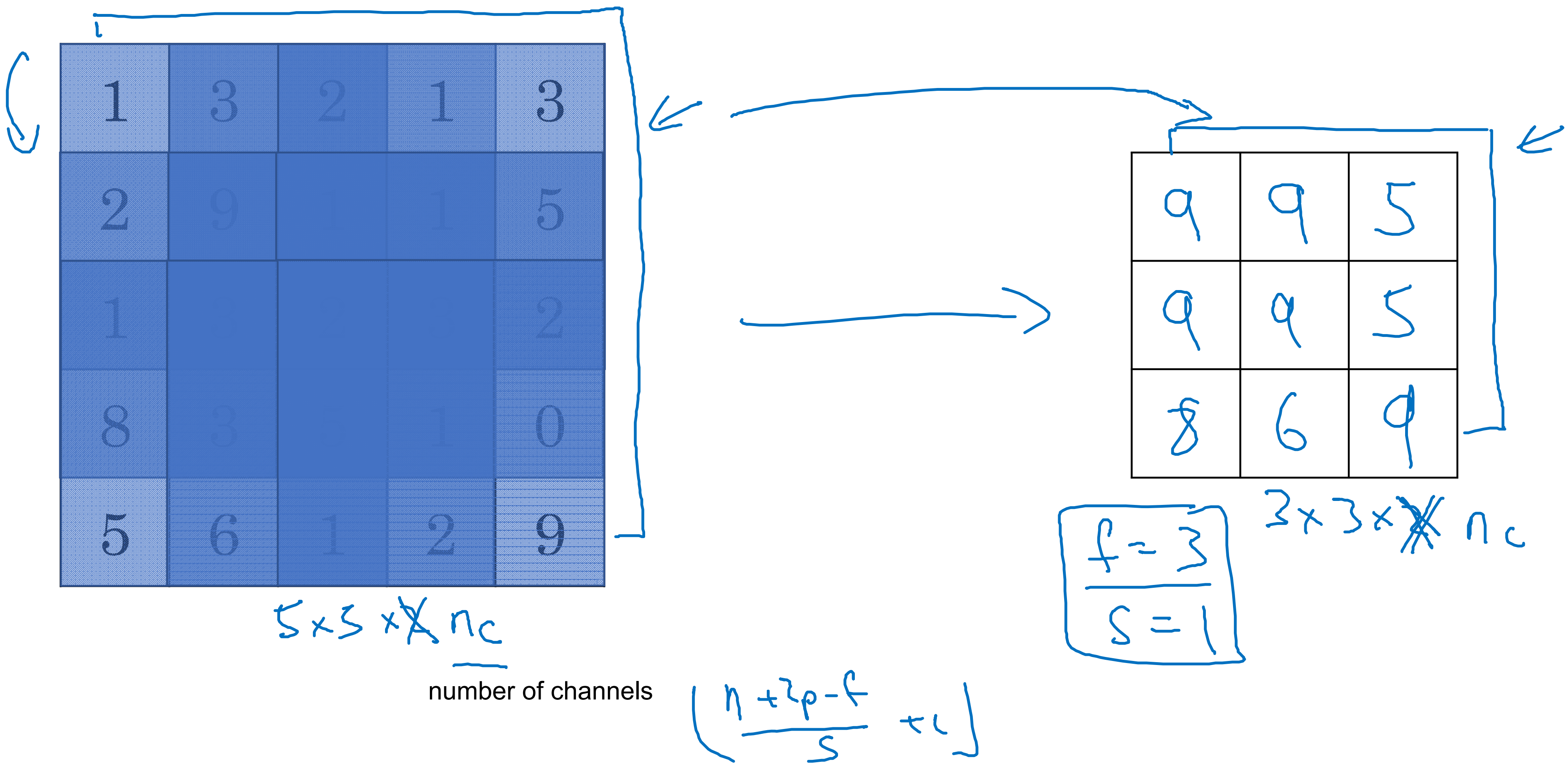
$s = 2$

Interesting fact about max pooling is that it has a set of parameter bu it has nothing to learn.

No parameters

So here's the intuition behind what max pooling is doing. If you think of this four by four region as some set of features, the activations in some layer of the neural network, then a large number, it means that it's maybe detected a particular feature. So, the upper left-hand quadrant has this particular feature. It maybe a vertical edge or maybe a higher or whisker if you trying to detect a [inaudible]. Clearly, that feature exists in the upper left-hand quadrant. Whereas this feature, maybe it isn't cat eye detector. Whereas this feature, it doesn't really exist in the upper right-hand quadrant. So what the max operation does is a lots of features detected anywhere, and one of these quadrants , it then remains preserved in the output of max pooling.

Pooling layer: Max pooling



Andrew Ng

Pooling layer: Average pooling

take avg instead of max
max pooling is used much more

1	3	2	1
2	9	1	1
1	4	2	3
5	6	1	2

$\longrightarrow$

3.75	1.25
4	2

$f = 2$

$s = 2$

$\underline{7 \times 7} \times 1000 \longrightarrow 1 \times 1 \times 1000$

Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

$f=2, s=2$

$f=3, s=2$

→ ~~p: padding.~~

No parameters to learn!

input

$$n_H \times n_W \times n_C$$

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$$

output

Andrew Ng

deeplearning.ai

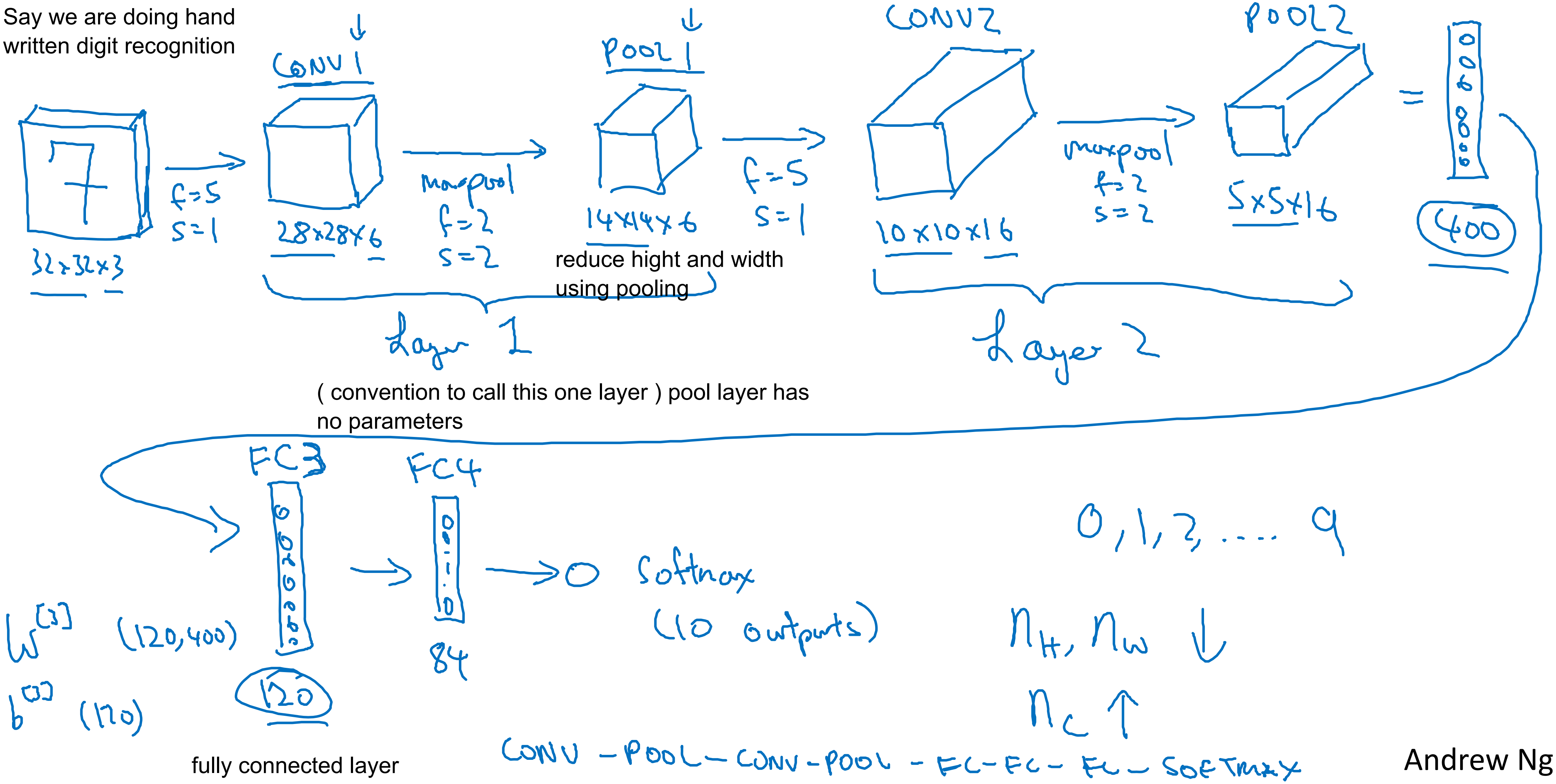
Convolutional Neural Networks

Convolutional neural network example

What we gone do here is inspired by LeNet -5 , very similar to that

Neural network example (LeNet-5)

Say we are doing hand written digit recognition



reduce hight and width using pooling

(convention to call this one layer) pool layer has no parameters

fully connected layer

Andrew Ng

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	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

Andrew Ng

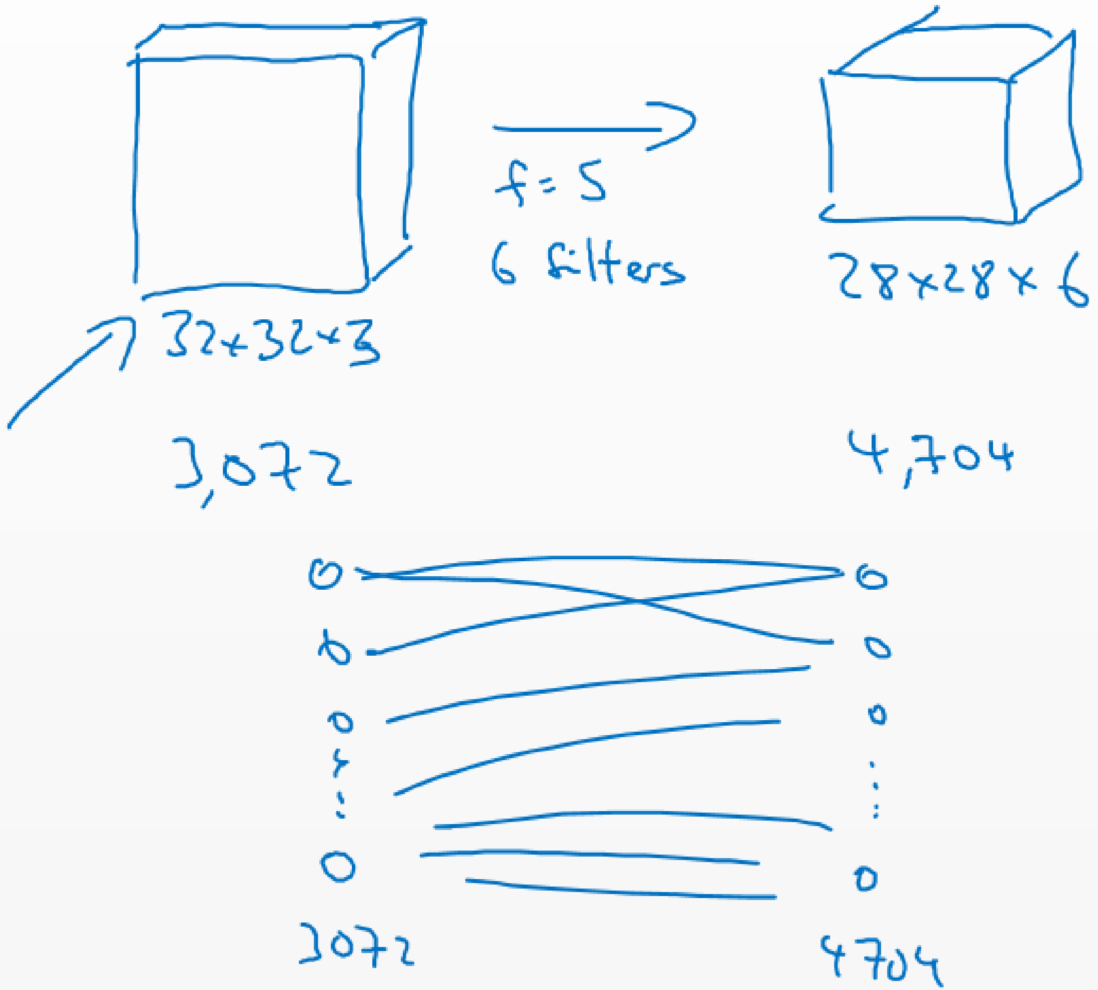
deeplearning.ai

Convolutional Neural Networks

Why convolutions?

Why convolutions

I think there are two main advantages of convolutional layers over just using fully connected layers.
And the advantages are:
- parameter sharing and
- sparsity of connections.



$5 \times 5 - 25$

26

$6 \times 26 = 156$ parameters

$3{,}072 \times 4{,}704 \approx \underline{14M}$

Andrew Ng

Why convolutions

translation invariance

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{3\times 3} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

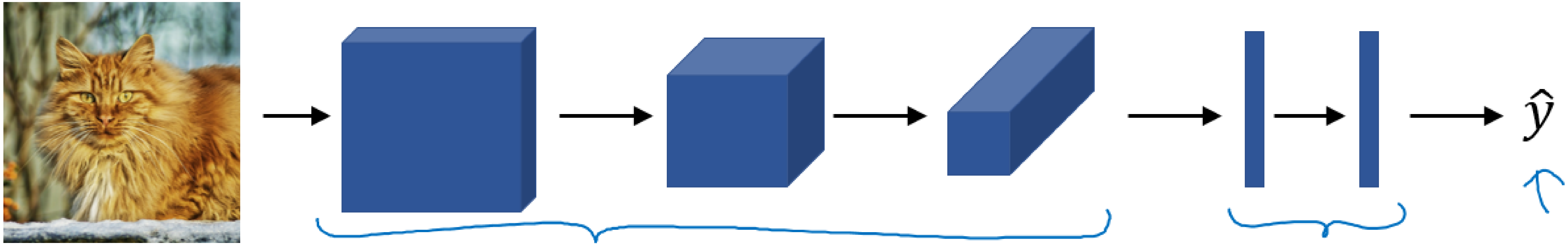
Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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

Putting it together

Training set $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$.

$\boxed{w, b}$



Cost $J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Use gradient descent to optimize parameters to reduce J