



deeplearning.ai

Convolutional Neural Networks

Computer vision

Computer Vision Problems

Image Classification



64x64



Cat? (0/1)



Neural Style Transfer



Object detection



Andrew Ng

Deep Learning on large images



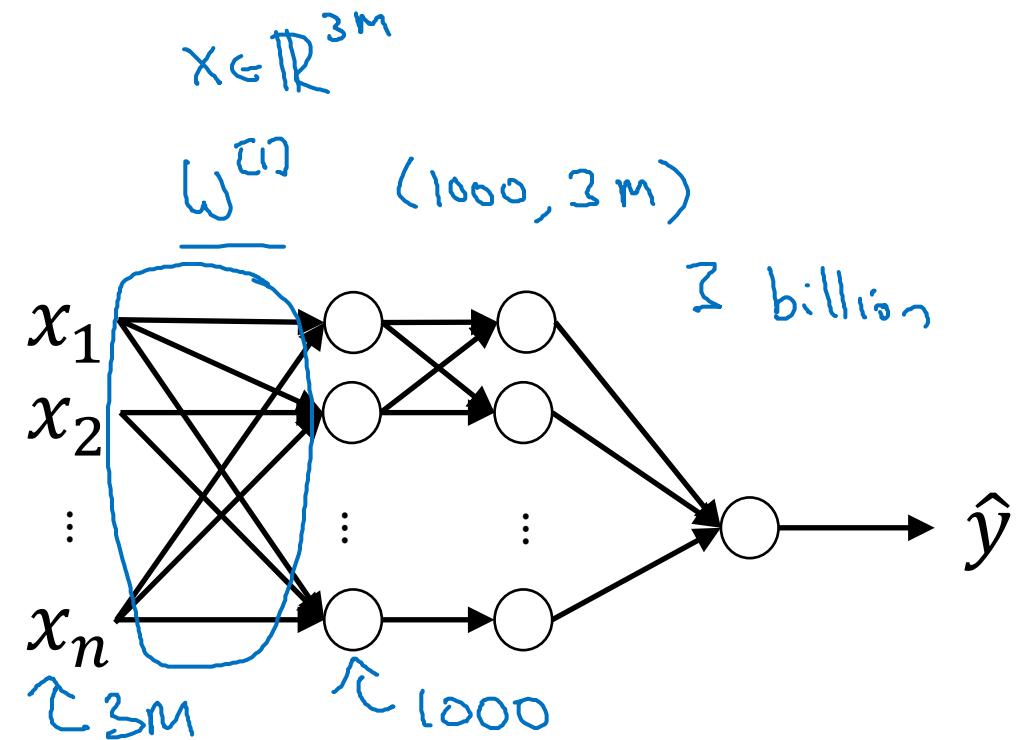
$64 \times 64 \times 3$

→ Cat? (0/1)

12288



$1000 \times 1000 \times 3$
= 3 million



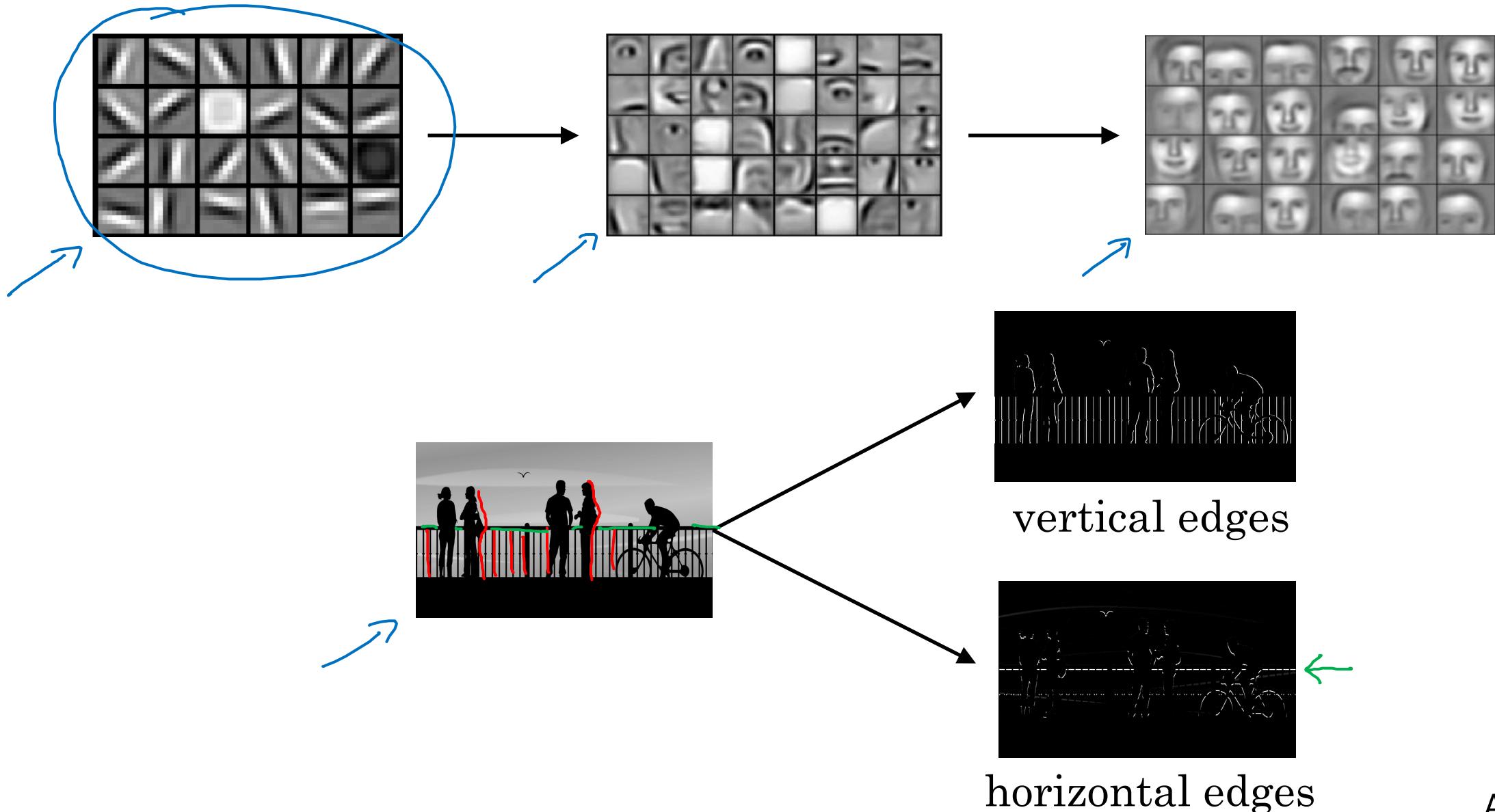


deeplearning.ai

Convolutional Neural Networks

Edge detection
example

Computer Vision Problem

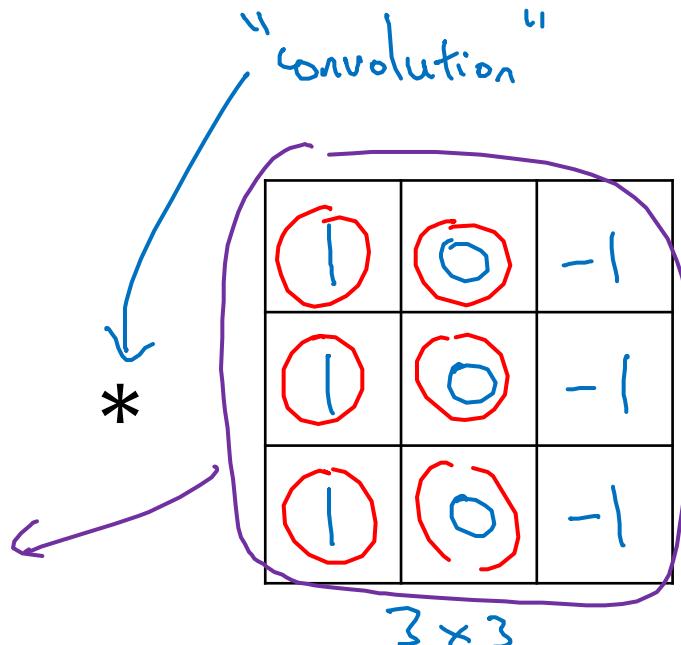


Vertical edge detection

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

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

6×6



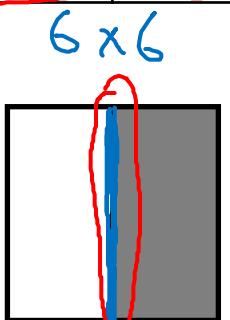
=

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

4×4

Vertical edge detection

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



6×6

*

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

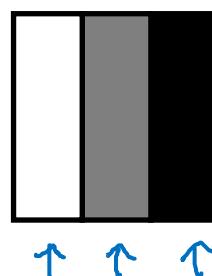
3×3

=

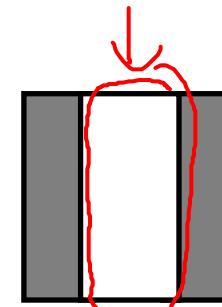
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4×4

*



$\uparrow \uparrow \uparrow$



Andrew Ng



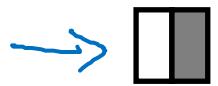
deeplearning.ai

Convolutional Neural Networks

More edge
detection

Vertical edge detection examples

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



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



*

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



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



*

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



=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



Vertical and Horizontal Edge Detection

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

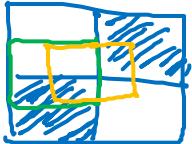
Vertical

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

6×6



*

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

Learning to detect edges

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

→

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

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

↑

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

Sobel filter

convolution

×

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3×3

=

45°
 70°
 73°

↑

Scharr filter



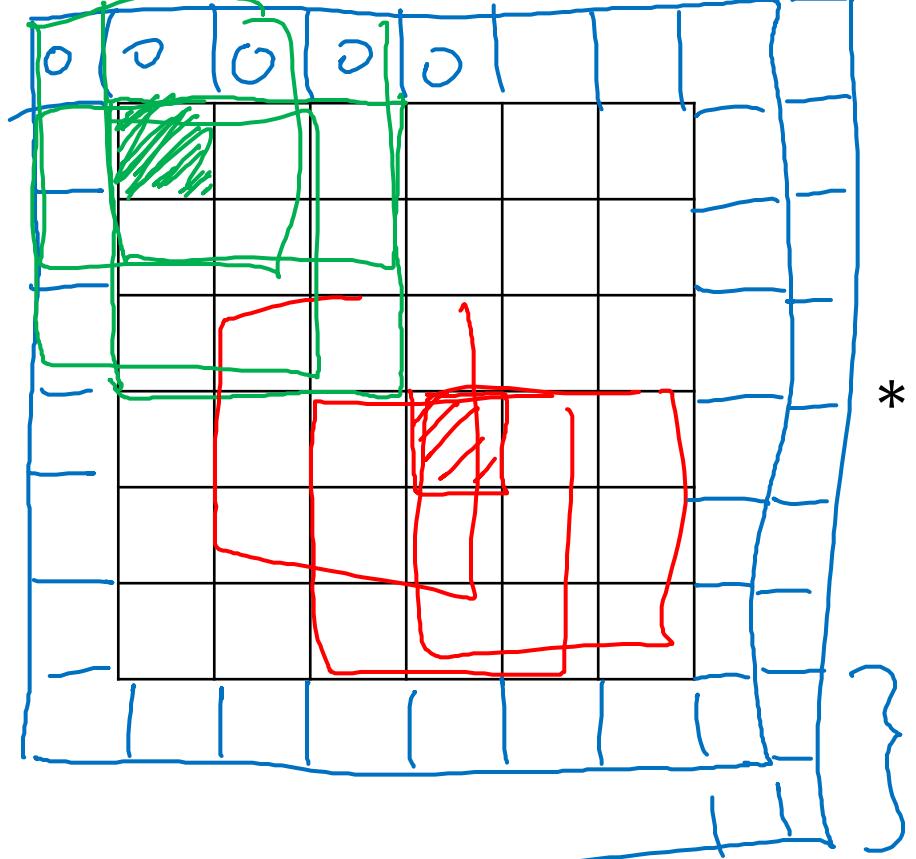
deeplearning.ai

Convolutional Neural Networks

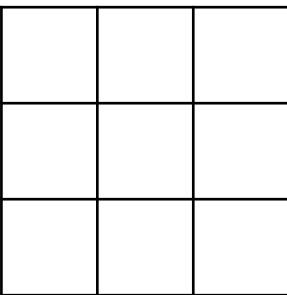
Padding

Padding

- shrinky output
- throw away info from edge



*



3×3
 $f \times f$

 $p=2$

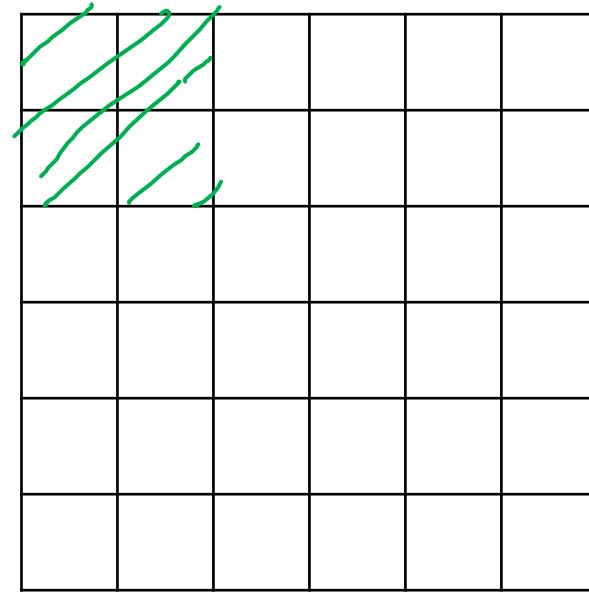
$$\frac{6 \times 6}{n \times n} \rightarrow 8 \times 8$$

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

$$6-3+1 = 4$$

$$P = \text{padding} = 1$$

=

 $\underline{6 \times 6}$

$$\xrightarrow{\quad} \underline{\underline{4 \times 4}}$$

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

$$6+2-3+1 \times \underline{\underline{\quad}} = 6 \times 6$$

Valid and Same convolutions

→ n → padding

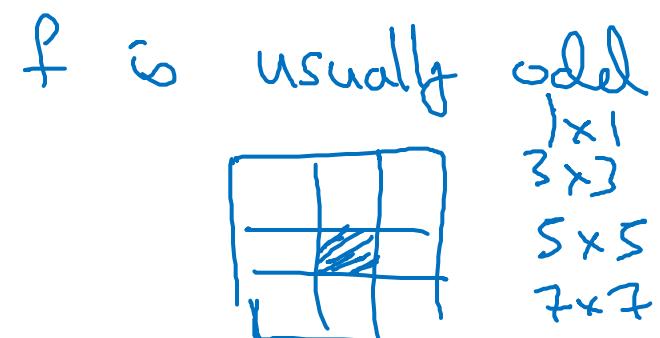
“Valid”: $n \times n \quad * \quad f \times f \quad \rightarrow \frac{n-f+1}{f} \times \frac{n-f+1}{f}$

$$\begin{array}{ccc} n \times n & * & f \times f \\ 6 \times 6 & * & 3 \times 3 \end{array} \rightarrow 4 \times 4$$

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

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

$$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad \left| \begin{array}{c} 5 \times 5 \\ f=5 \end{array} \right. \quad p=2$$



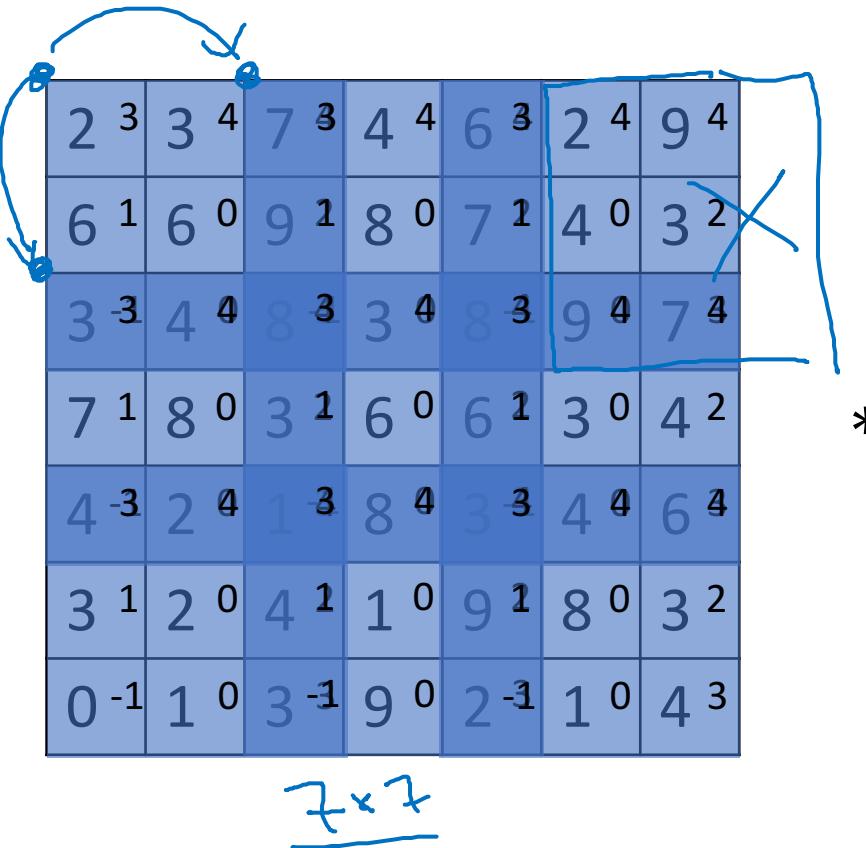


deeplearning.ai

Convolutional Neural Networks

Strided convolutions

Strided convolution



$$\begin{matrix} 91 & 100 & 83 \\ 69 & 91 & 127 \\ 44 & 72 & 74 \end{matrix} = \begin{matrix} 3 \times 3 \\ \lfloor \frac{z}{2} \rfloor = \text{floor}(z) \end{matrix}$$

$n \times n$ * $f \times f$
padding p stride s
 $s=2$

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

$$\left\lfloor \frac{7+0-3}{2} + 1 \right\rfloor = \frac{4}{2} + 1 = 3$$

Summary of convolutions

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

padding p stride s

Output size:

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


Technical note on cross-correlation vs. convolution

Convolution in math textbook:

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

$$\begin{matrix} & * & \begin{matrix} 3 & 4 & 5 \\ 1 & 0 & 2 \\ -1 & 9 & 7 \end{matrix} \\ \begin{matrix} 7 & 2 & 5 \\ 9 & 0 & 4 \\ -1 & 1 & 3 \end{matrix} & \end{matrix}$$

$$= \begin{matrix} & \begin{matrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \end{matrix}$$

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

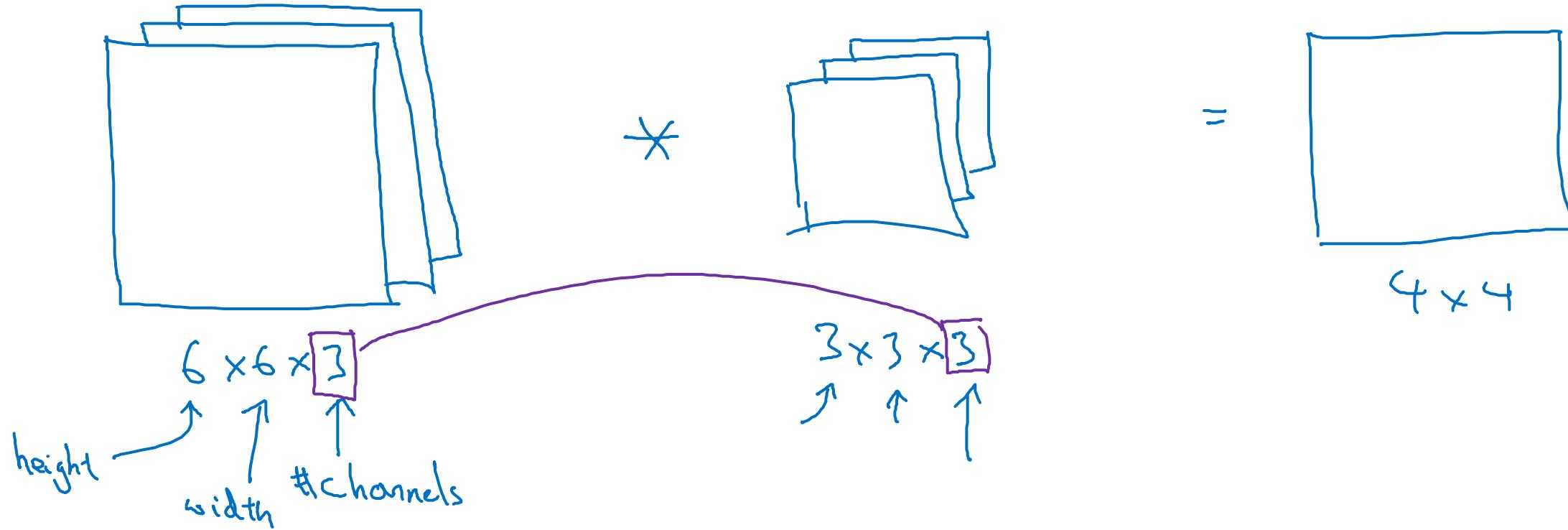


deeplearning.ai

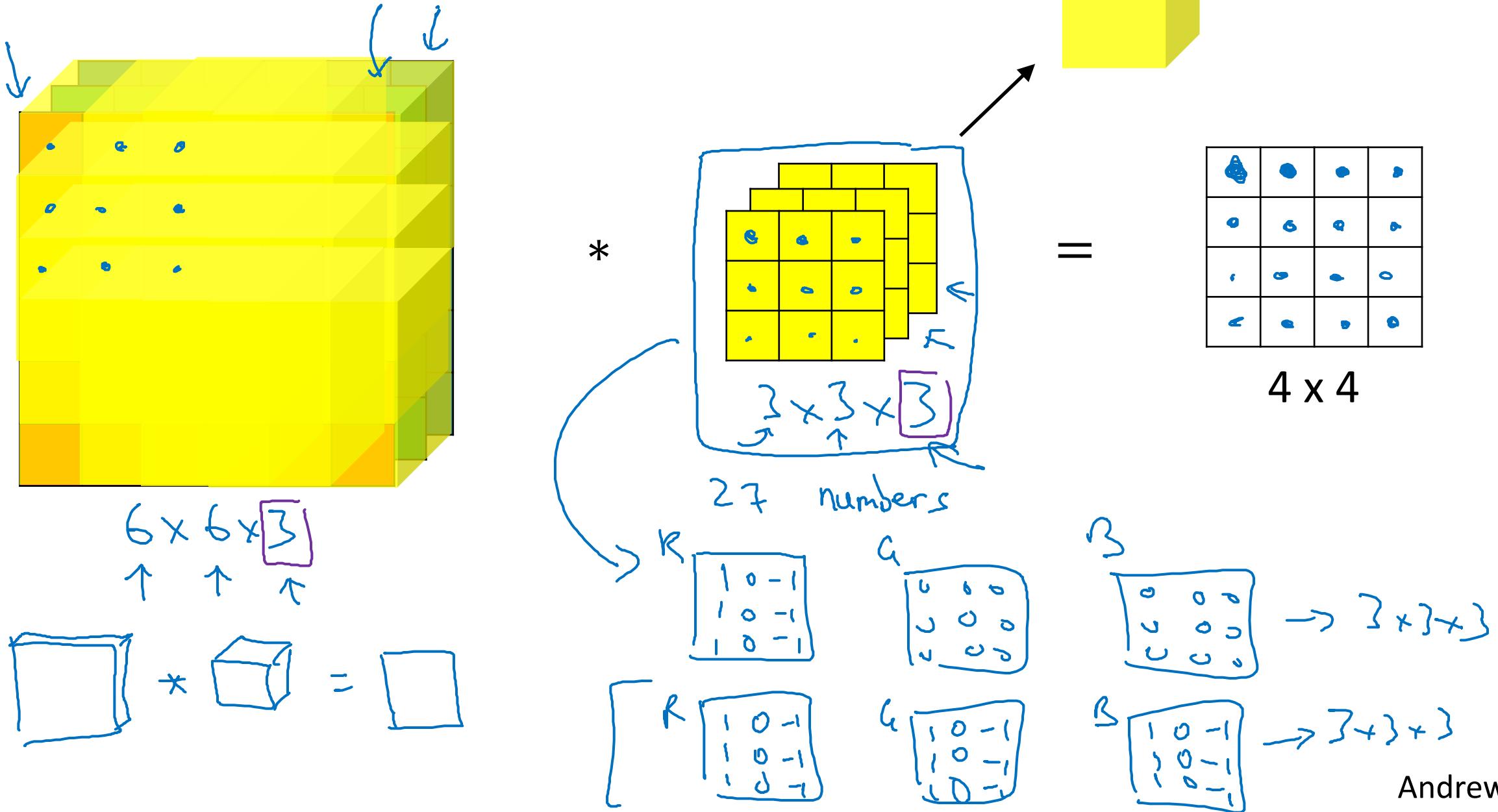
Convolutional Neural Networks

Convolutions over volumes

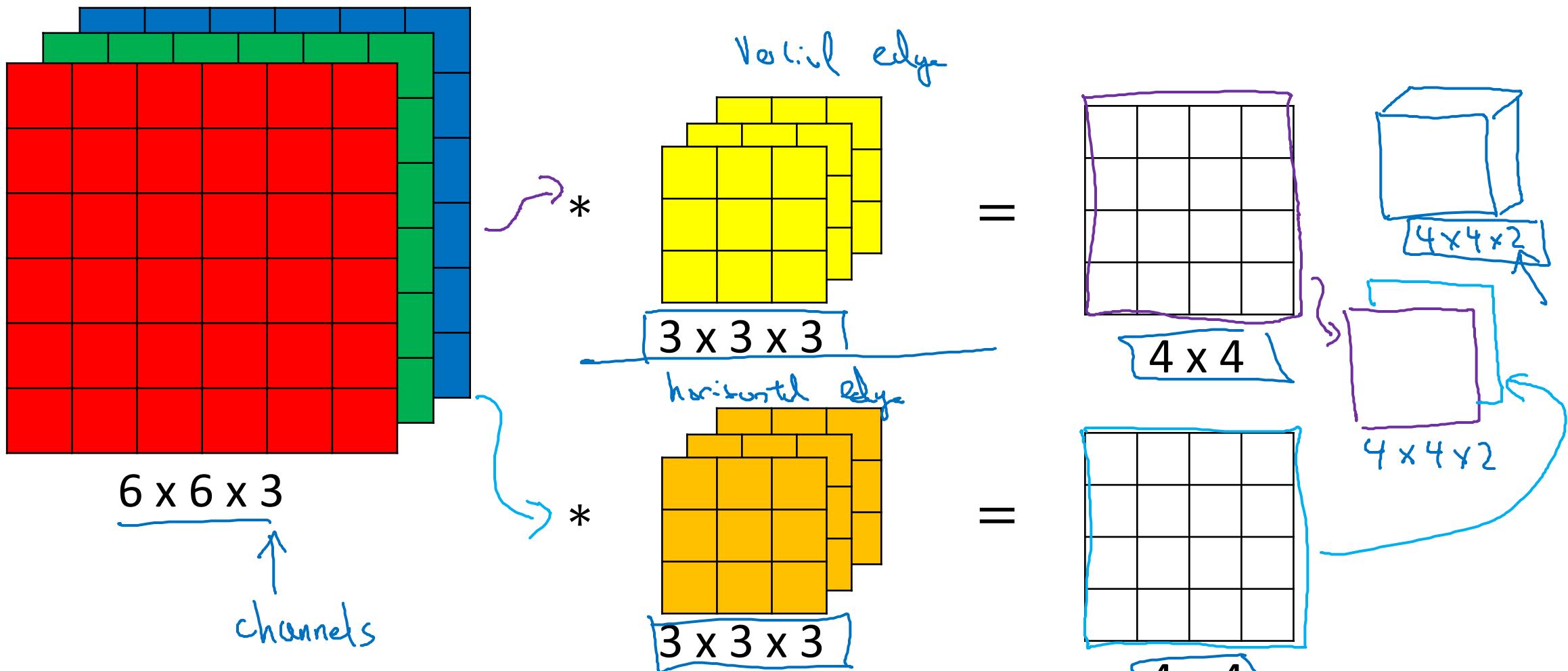
Convolutions on RGB images



Convolutions on RGB image



Multiple filters



Summary: $n \times n \times n_c$ $\times f \times f \times n_c$ $\rightarrow \frac{n-f+1}{4} \times \frac{n-f+1}{4} \times \frac{n_c}{2} \# \text{filters}$

$6 \times 6 \times 3$ $3 \times 3 \times 3$

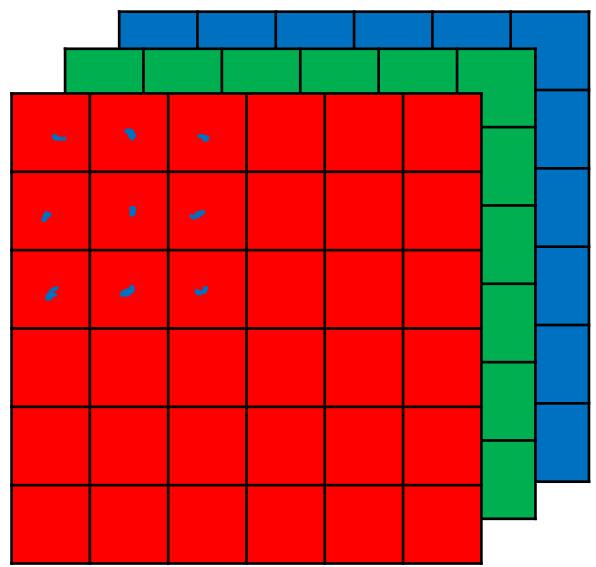


deeplearning.ai

Convolutional Neural Networks

One layer of a
convolutional
network

Example of a layer

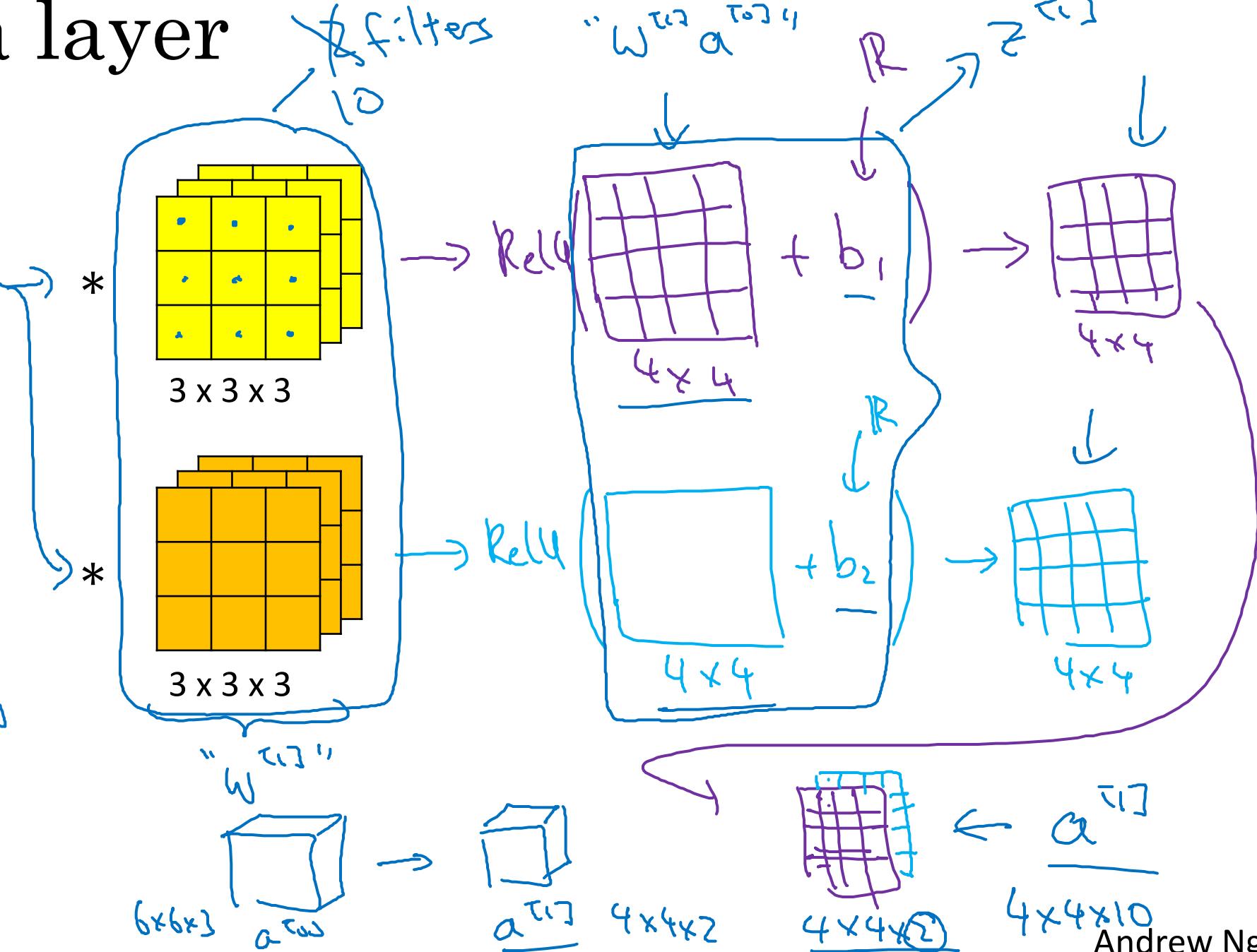


$6 \times 6 \times 3$

$a^{(1)}$

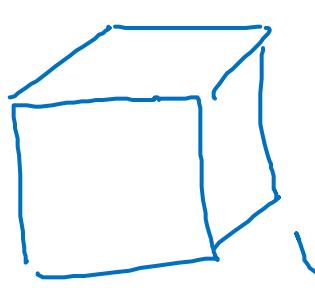
$$z^{(1)} = W^{(1)} a^{(1)} + b^{(1)}$$

$$a^{(2)} = g(z^{(1)})$$



Number of parameters in one layer

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

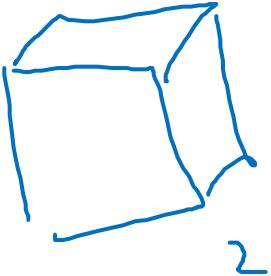


$3 \times 3 \times 3$

27 parameters.

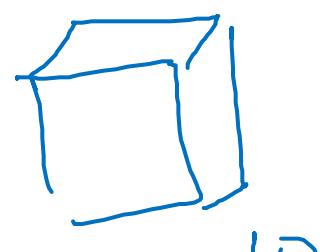
+ bias

→ 28 parameters.



2

...
...



10

280 parameters.

Summary of notation

If layer l is a convolution layer:

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

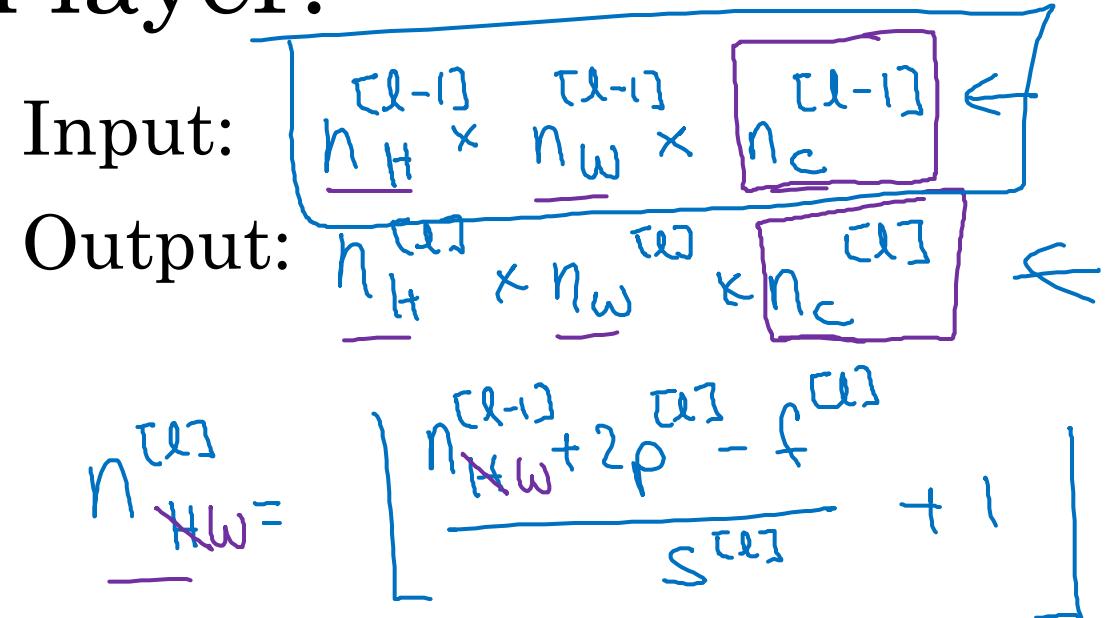
$n_c^{[l]}$ = number of filters

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

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

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$ ↪ #f: #ftrs in layer l.



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

$$n_c \times n_H \times n_W$$

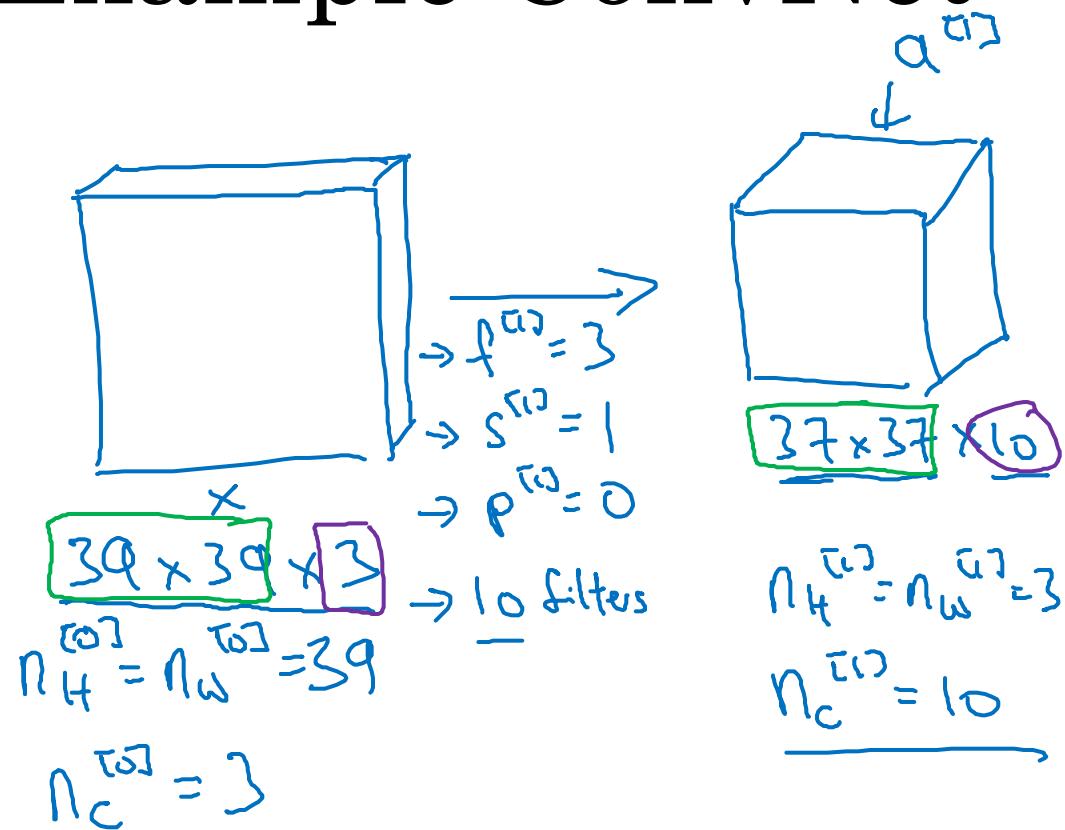


deeplearning.ai

Convolutional Neural Networks

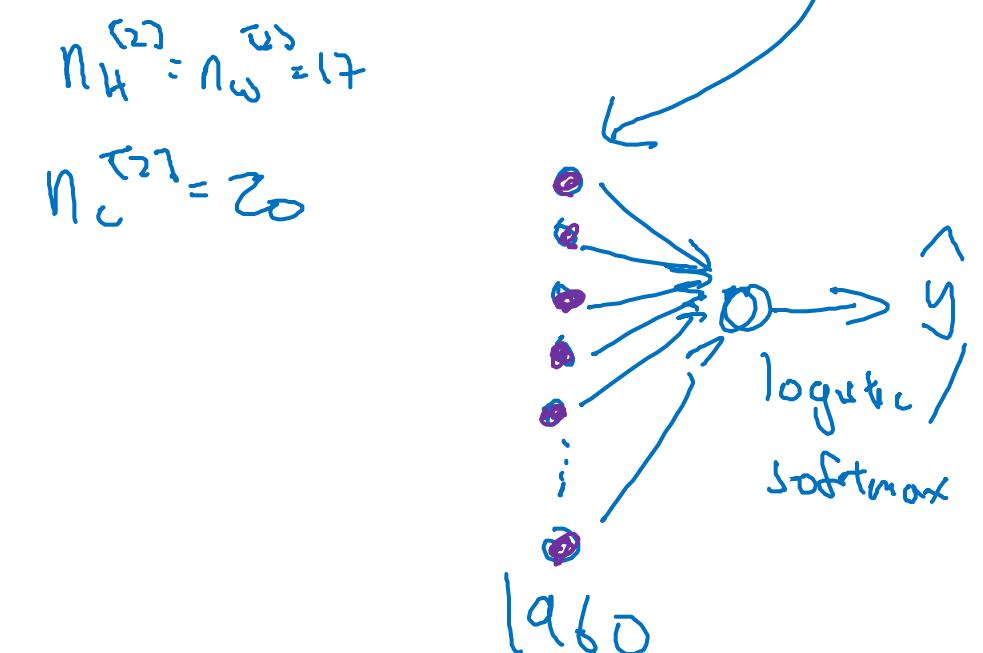
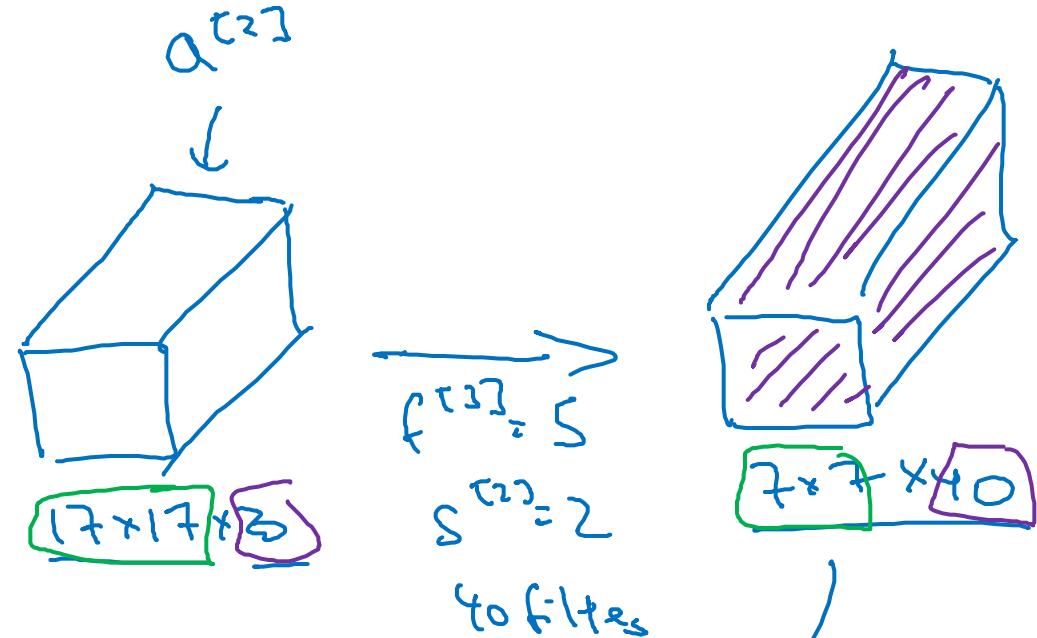
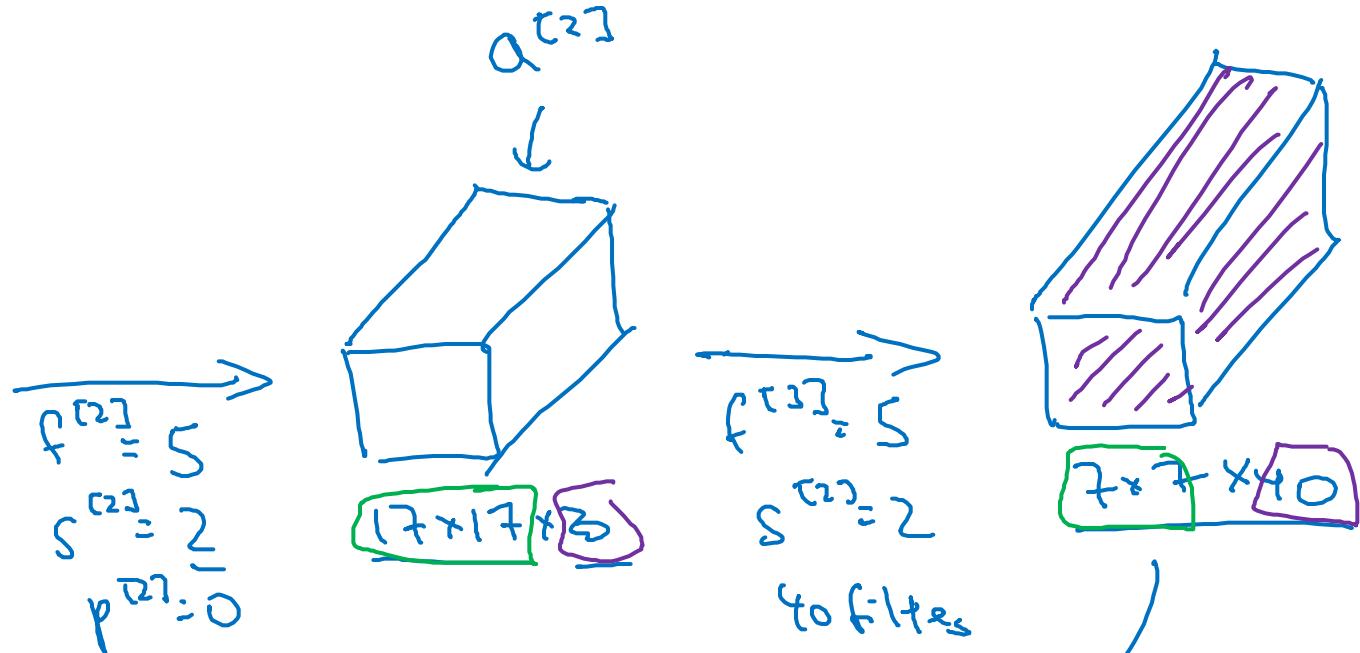
A simple convolution network example

Example ConvNet



$$\frac{n_H^{[1]} + 2p^{[1]} - f^{[1]}}{s^{[1]}} + 1$$

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



Types of layer in a convolutional network:

- Convolution (Conv) ←
- Pooling (pool) ←
- Fully connected (Fc) ←



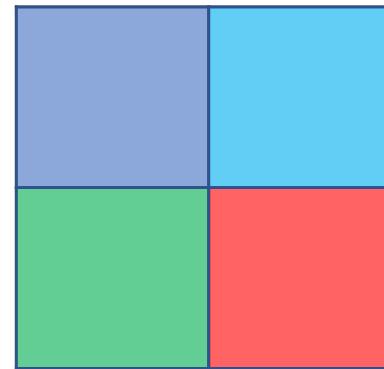
deeplearning.ai

Convolutional Neural Networks

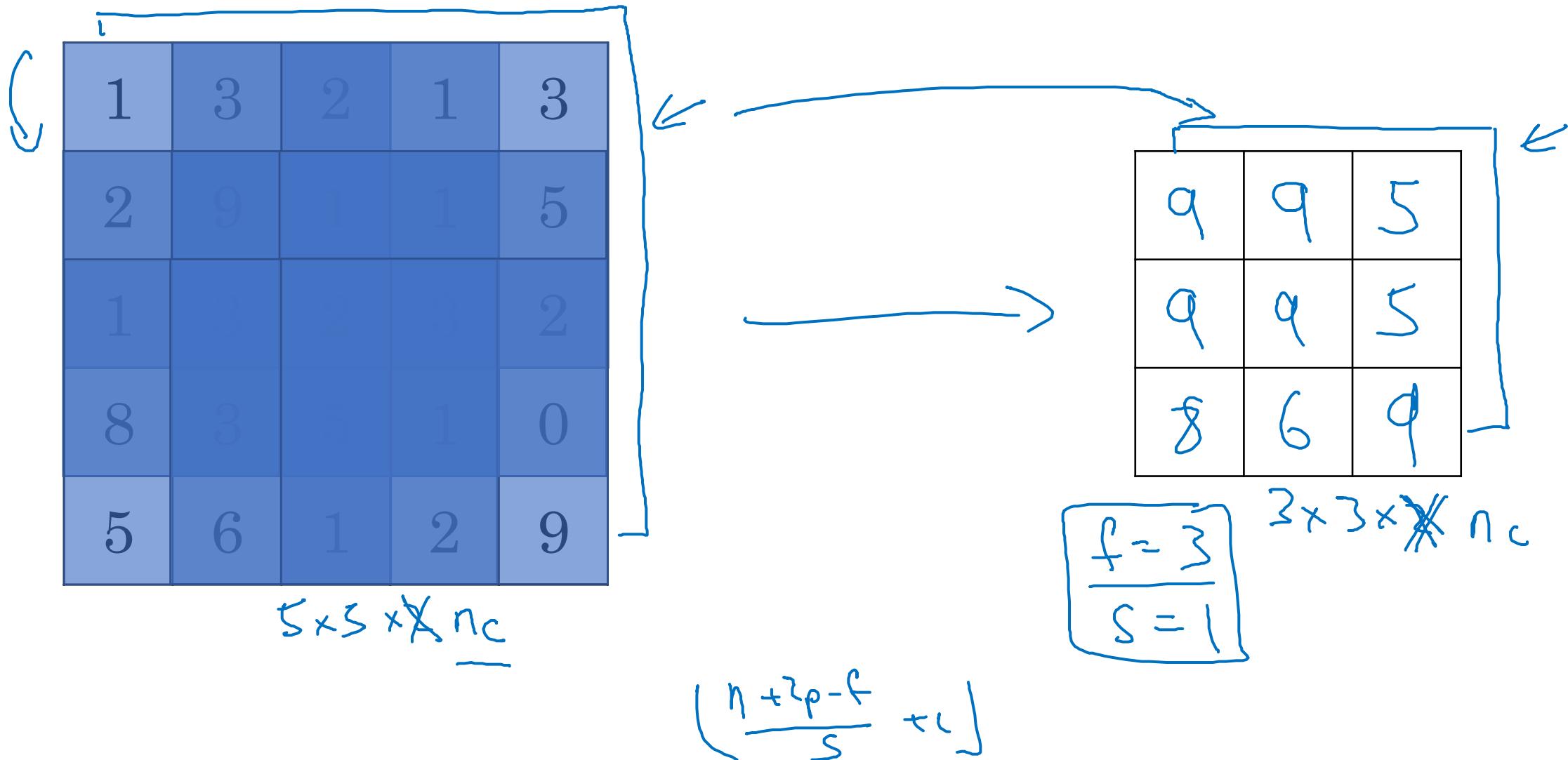
Pooling layers

Pooling layer: Max pooling

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



Pooling layer: Max pooling



Pooling layer: Average pooling

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



3.75	1.25
4	2

$$f=2$$

$$s=2$$

$$\underbrace{7 \times 7 \times 1000}_{\rightarrow} \rightarrow 1 \times 1 \times 1000$$

Summary of pooling

Hyperparameters:

f : filter size

$$f=2, s=2$$

s : stride

$$f=3, s=2$$

Max or average pooling

$\rightarrow p$: padding.

No parameters to learn!

$$n_H \times n_w \times n_c$$



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

$$\times n_c$$



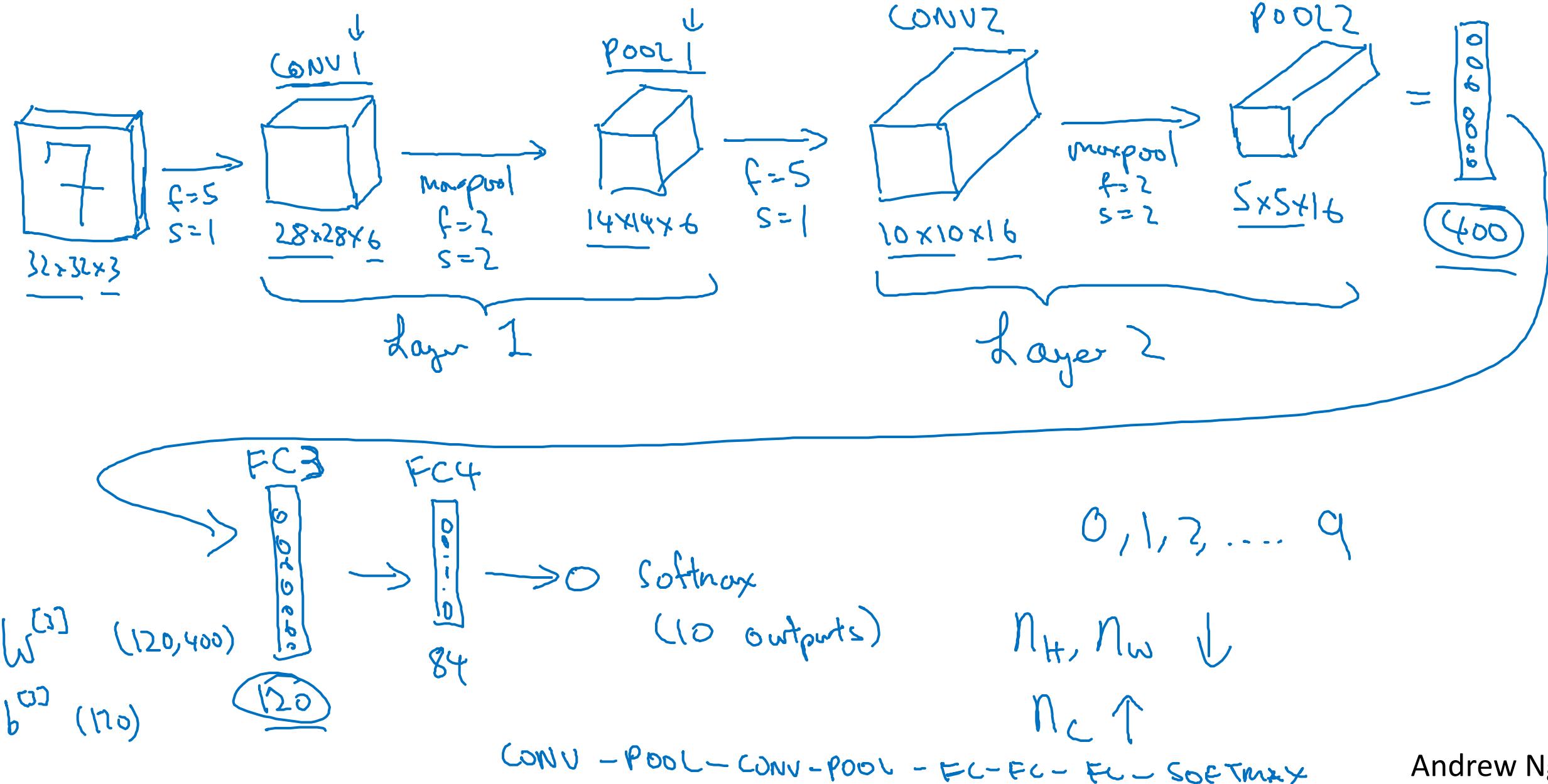
deeplearning.ai

Convolutional Neural Networks

Convolutional neural network example

Neural network example

(LeNet-5)



Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[1]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	608 ←
POOL1	(14,14,8)	1,568	0 ←
CONV2 (f=5, s=1)	(10,10,16)	1,600	3216 ←
POOL2	(5,5,16)	400	0 ←
FC3	(120,1)	120	48120 }
FC4	(84,1)	84	10164 }
Softmax	(10,1)	10	850

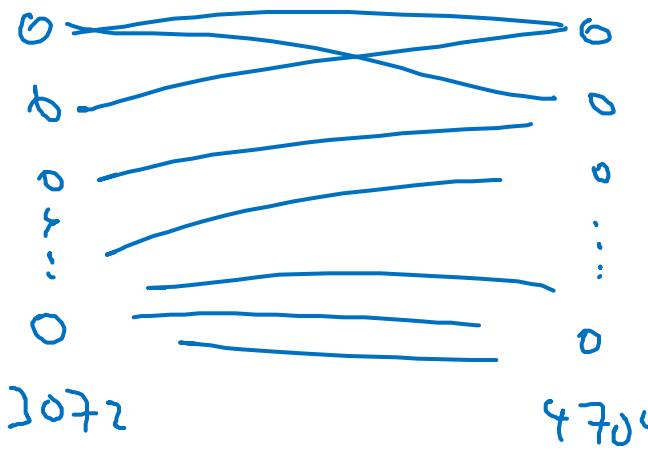
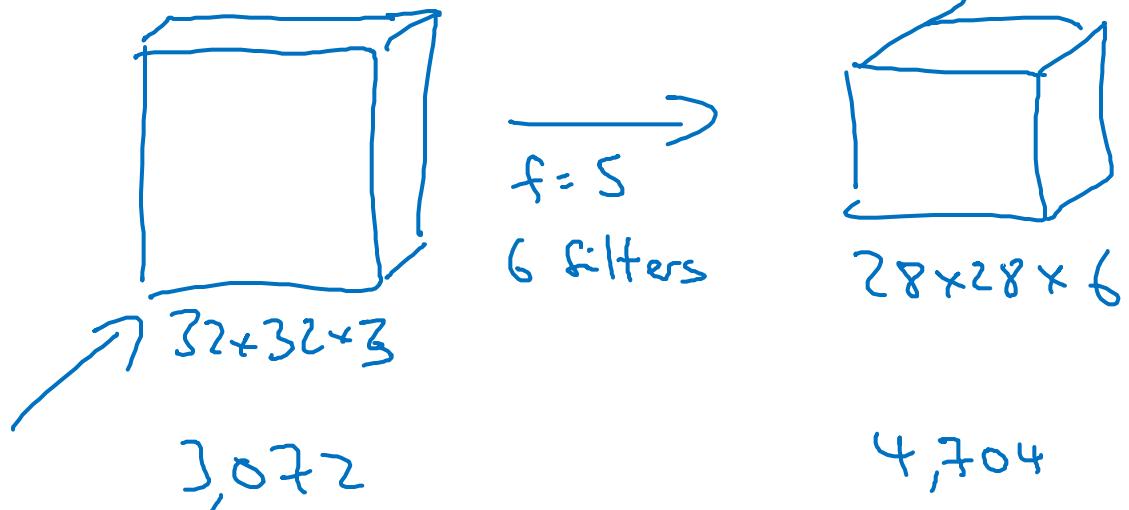


deeplearning.ai

Convolutional Neural Networks

Why convolutions?

Why convolutions

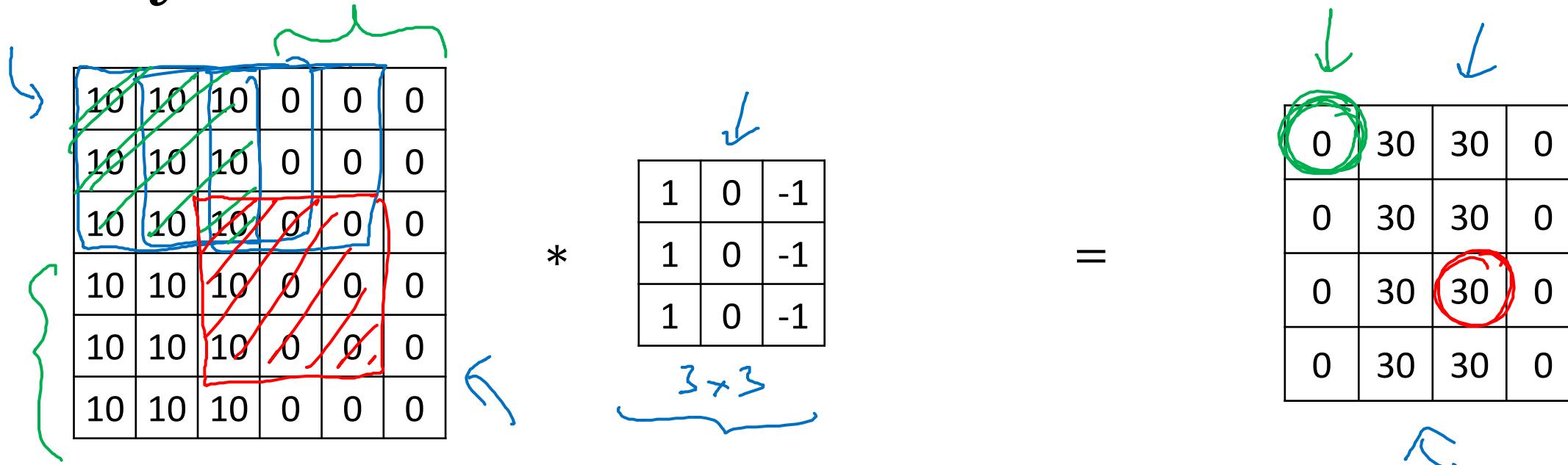


$$5 \times 5 - 25$$

$$26 \\ 6 \times 26 = 156 \text{ Parameters}$$

$$3,072 \times 4,704 \approx 14M$$

Why convolutions

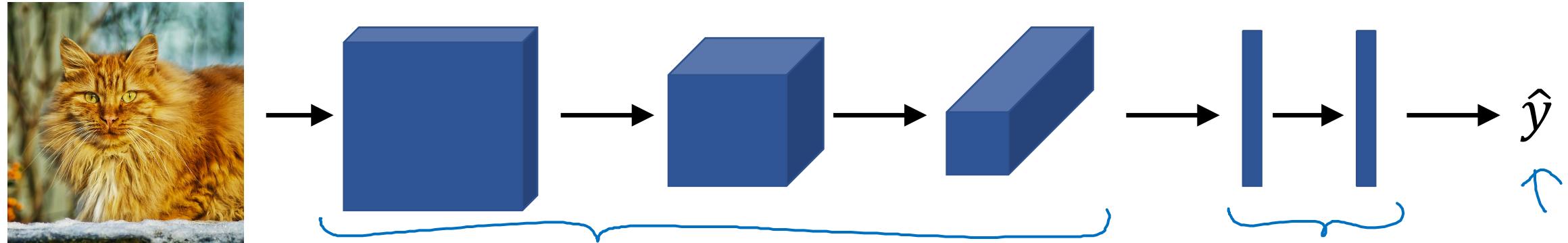


Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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

Putting it together

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



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

Use gradient descent to optimize parameters to reduce J