



deeplearning.ai

# Convolutional Neural Networks

---

## Computer vision

# Computer Vision Problems

Image Classification



64x64

→ Cat? (0/1)

Object detection



Neural Style Transfer



Andrew Ng

# Deep Learning on large images



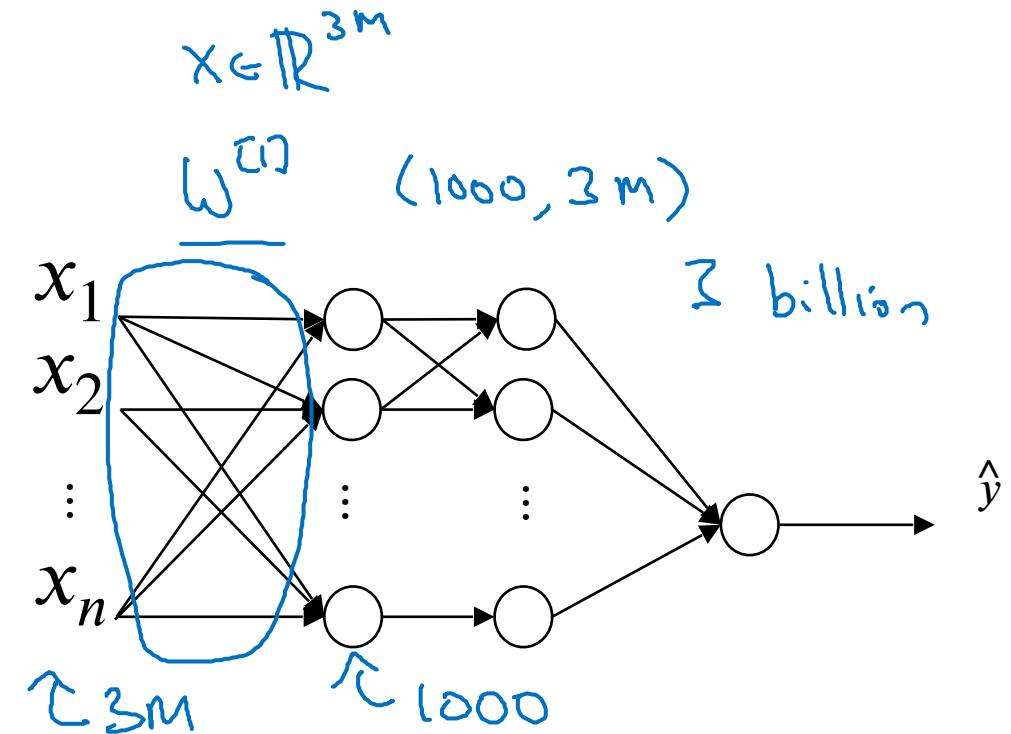
$64 \times 64 \times 3$

→ Cat? (0/1)

12288



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





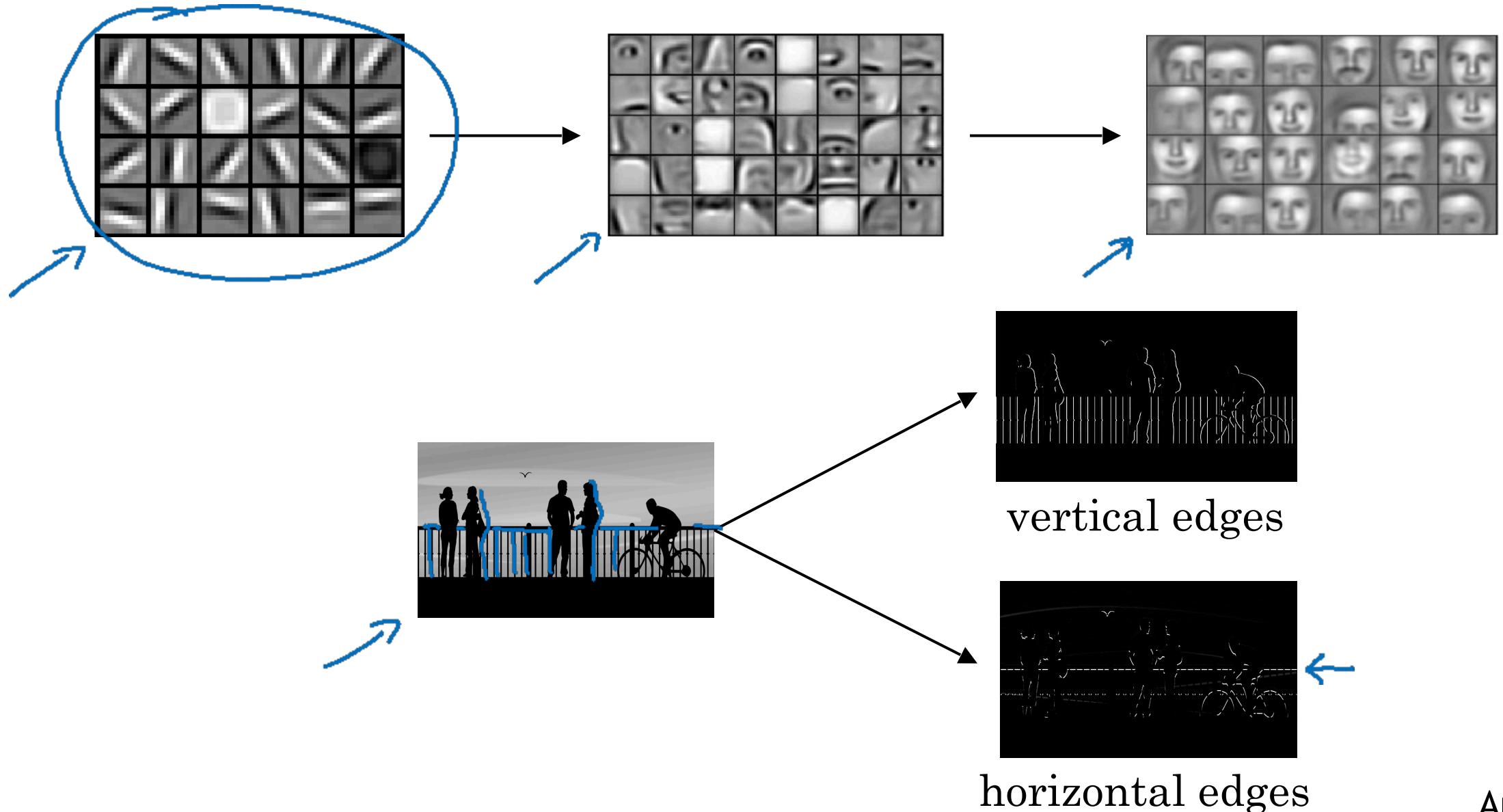
deeplearning.ai

# Convolutional Neural Networks

---

## Edge detection example

# Computer Vision Problem

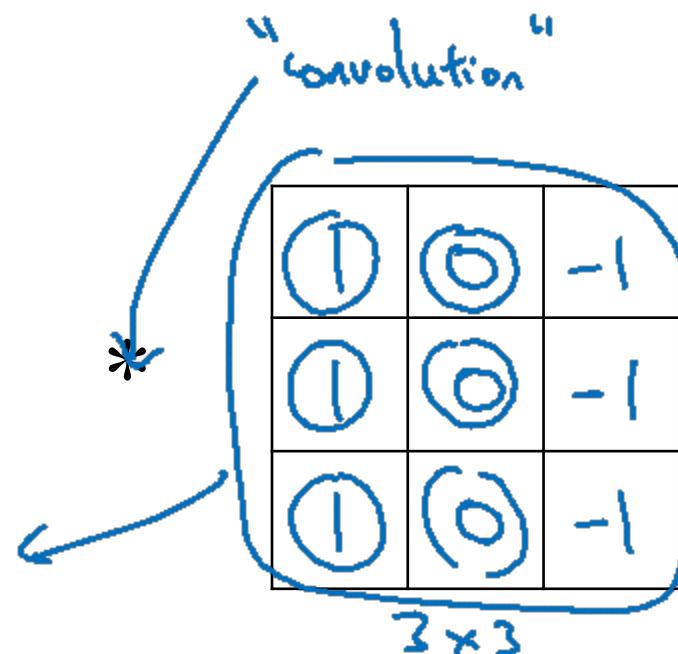


# Vertical edge detection

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

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

$6 \times 6$



=

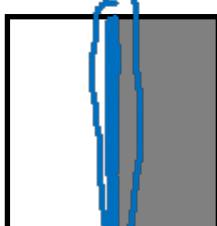
-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

$4 \times 4$

# 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

$6 \times 6$



$\uparrow \uparrow \uparrow$

\*

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

$3 \times 3$

=

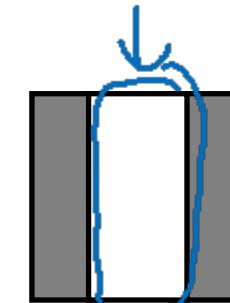
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

$\uparrow 4 \times 4$

\*



$\uparrow \uparrow \uparrow$



Andrew Ng



deeplearning.ai

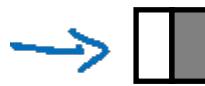
# Convolutional Neural Networks

---

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



\*

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



=

0	-3	-3	0
0	-3	-3	0
0	-3	-3	0
0	0	0	0
0	-1	3	0
0	0	0	0



# Vertical and Horizontal Edge Detection

A 3x3 input matrix with values 1, 0, -1 in each row. A blue arrow points from the first row to the second row, indicating the shift of the kernel. The second row is circled in blue.

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

Vertical

A 3x3 input matrix with values 1, 1, 1 in the first row. A blue arrow points from the first row to the second row, indicating the shift of the kernel. The first row is circled in blue.

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

Horizontal

A 6x6 input matrix with values 10 in the top-left 3x3 block and 10 in the bottom-right 3x3 block. A blue arrow points from the top-left block to the bottom-right block, indicating the shift of the kernel. The top-left and bottom-right blocks are circled in blue.

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

$6 \times 6$



\*

A 3x3 kernel with values 1, 1, 1 in the first row, 0, 0, 0 in the second row, and -1, -1, -1 in the third row.

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

=

The resulting 6x6 output matrix. The top-left 3x3 block has values 30, 10, -1 and the bottom-right 3x3 block has values -3, 0, 0. The other elements are 0. A blue arrow points from the top-left block to the bottom-right block, indicating the shift of the kernel. The top-left and bottom-right blocks are circled in blue.

0	0	0	0
30	10	-1	-3
0	0	0	0
30	10	-1	-3
0	0	0	0
0	0	0	0

# Learning to detect edges

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

→

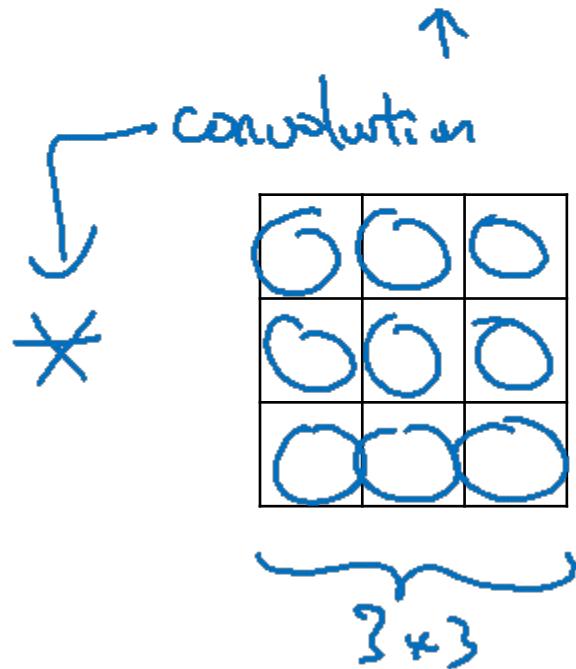
1	0	-1
2	0	-2
1	0	-1

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

Sobel 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



↑




deeplearning.ai

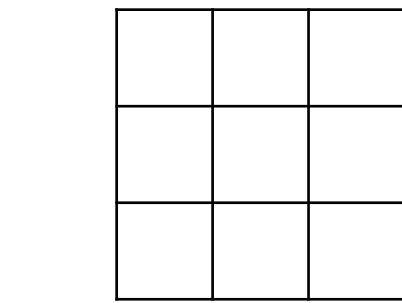
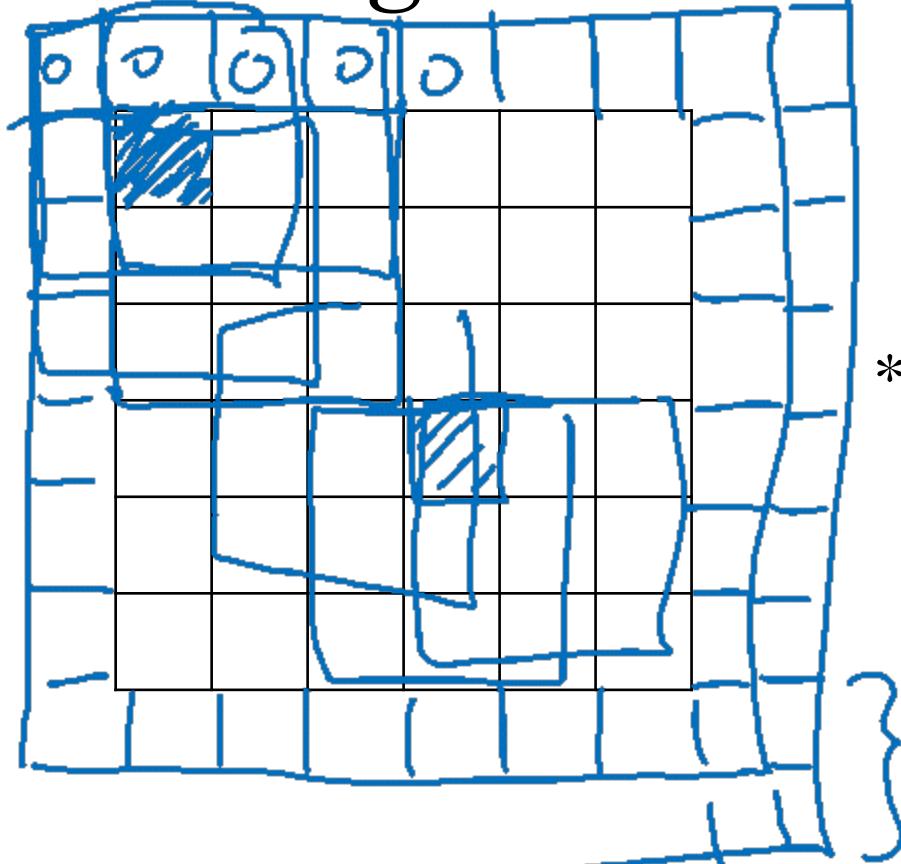
# Convolutional Neural Networks

---

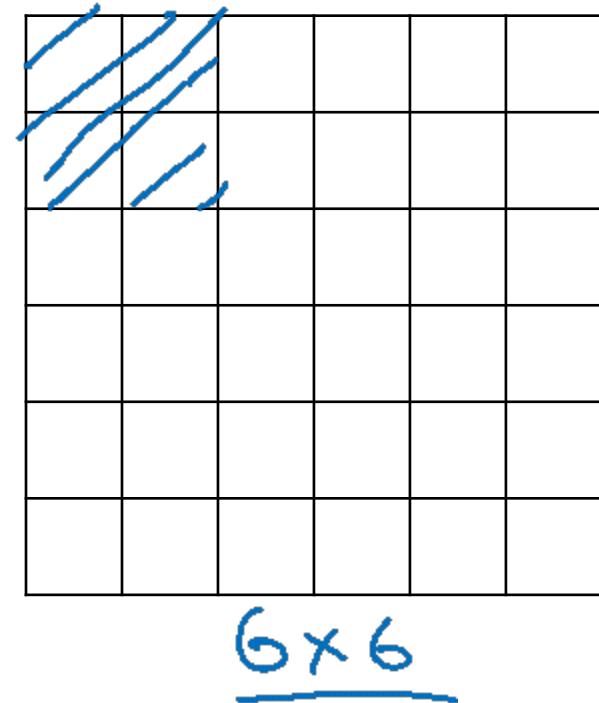
## Padding

# Padding

- shrinks output
- throws away info from edge



=



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

$$n-f+1 \times n-f+1$$
$$6-3+1 = 4$$

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

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

# Valid and Same convolutions

→  $n = \text{padding}$

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

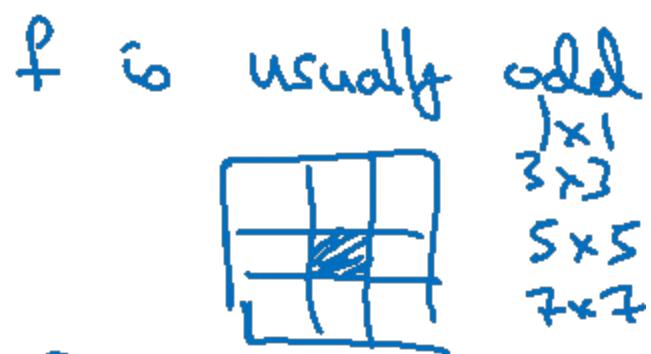
$$\begin{array}{ccc} 6 \times 6 & * & 3 \times 3 \\ 6 \times 6 & * & 3 \times 3 \end{array} \rightarrow 4 \times 4$$

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

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

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

$$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad | \quad \begin{matrix} S \times S \\ f=5 \end{matrix} \quad p=2$$





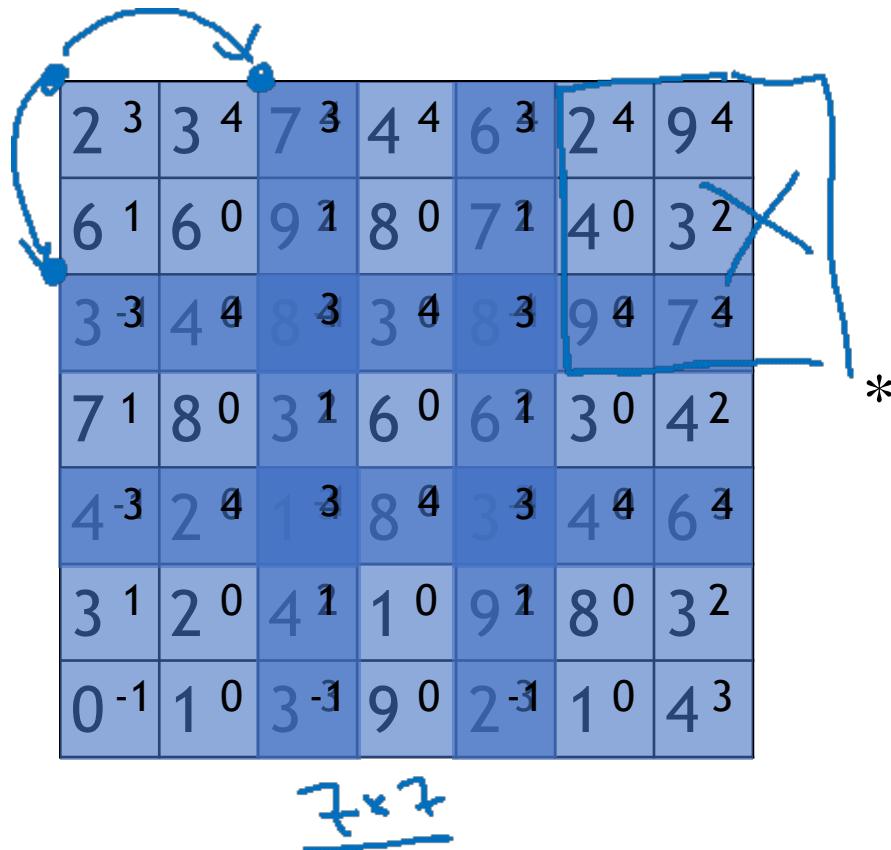
deeplearning.ai

# Convolutional Neural Networks

---

## Strided convolutions

# Strided convolution



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

$$\begin{array}{|c|c|c|} \hline 3 & 4 & 4 \\ \hline 1 & 0 & 2 \\ \hline -1 & 0 & 3 \\ \hline \end{array}$$

$3 \times 3$

stride = 2

=

$$\begin{array}{|c|c|c|} \hline 91 & 100 & 83 \\ \hline 69 & 91 & 127 \\ \hline 44 & 72 & 74 \\ \hline \end{array}$$

$3 \times 3$

$\lfloor \frac{n}{s} \rfloor = \text{floor}(\frac{n}{s})$

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

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

# Summary of convolutions

image

filter

padding  $p$

stride  $s$

Output size:

$\times$



# Technical note on cross-correlation vs. convolution

Convolution in math textbook:

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

$$A \star B = A \star (B \star C)$$

Diagram illustrating the convolution operation  $A \star B$ :

The input matrix  $A$  is:

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

The filter matrix  $B$  is:

3	4	5
1	0	2
-1	9	7

The result of the convolution  $A \star B$  is:

7	2	5
9	0	4
-1	1	3

$$(A \star B) \star C = A \star (B \star C)$$

Diagram illustrating the convolution operation  $(A \star B) \star C$ :

The input matrix  $A$  is:

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

The filter matrix  $B$  is:

3	4	5
1	0	2
-1	9	7

The result of the convolution  $(A \star B) \star C$  is:

7	2	5
9	0	4
-1	1	3



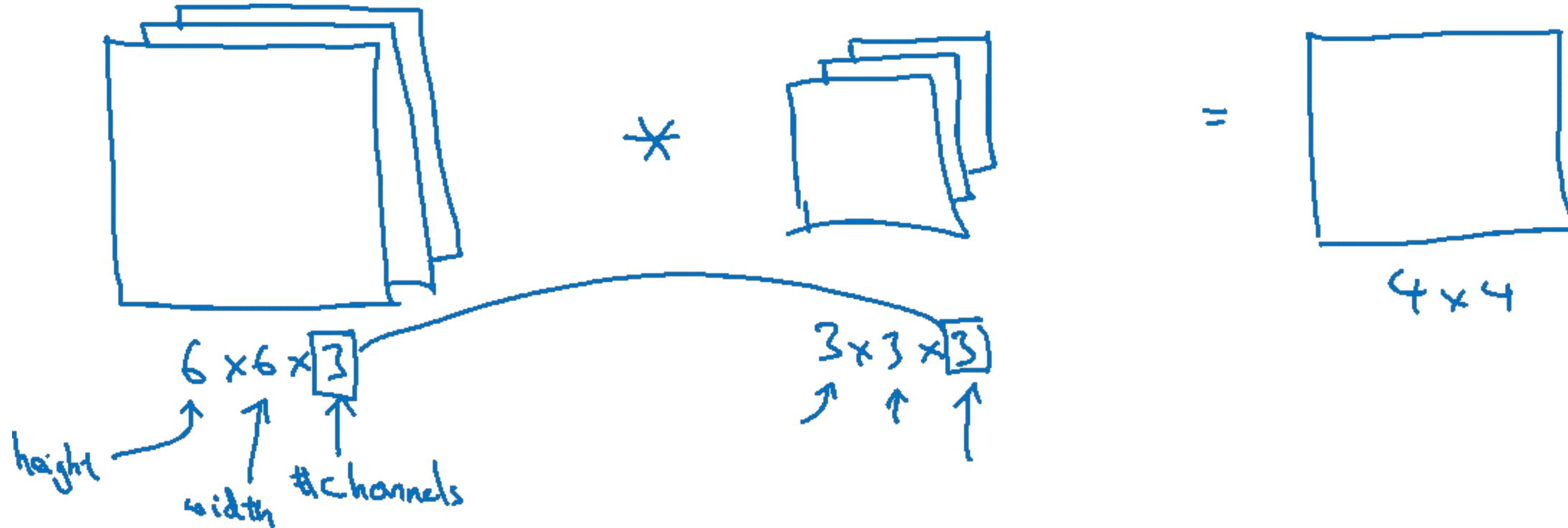
deeplearning.ai

# Convolutional Neural Networks

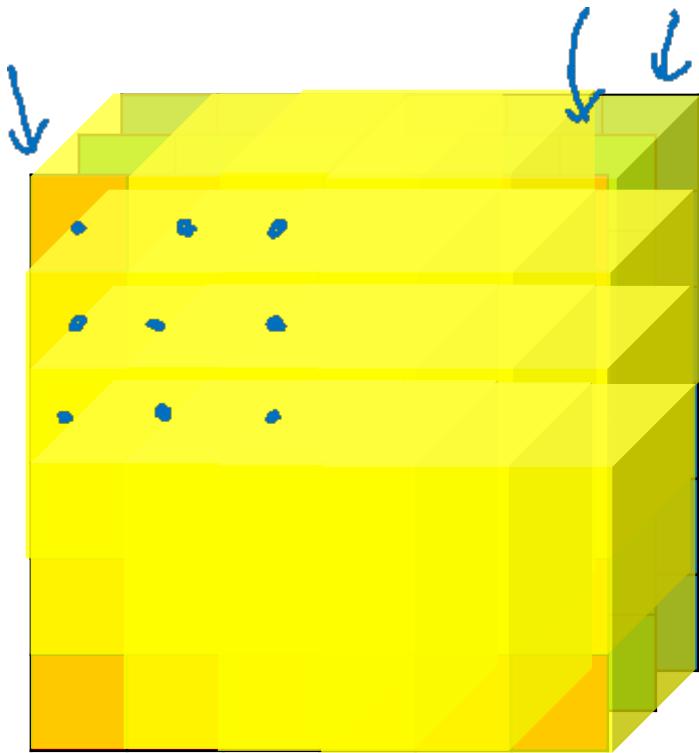
---

## Convolutions over volumes

# Convolutions on RGB images

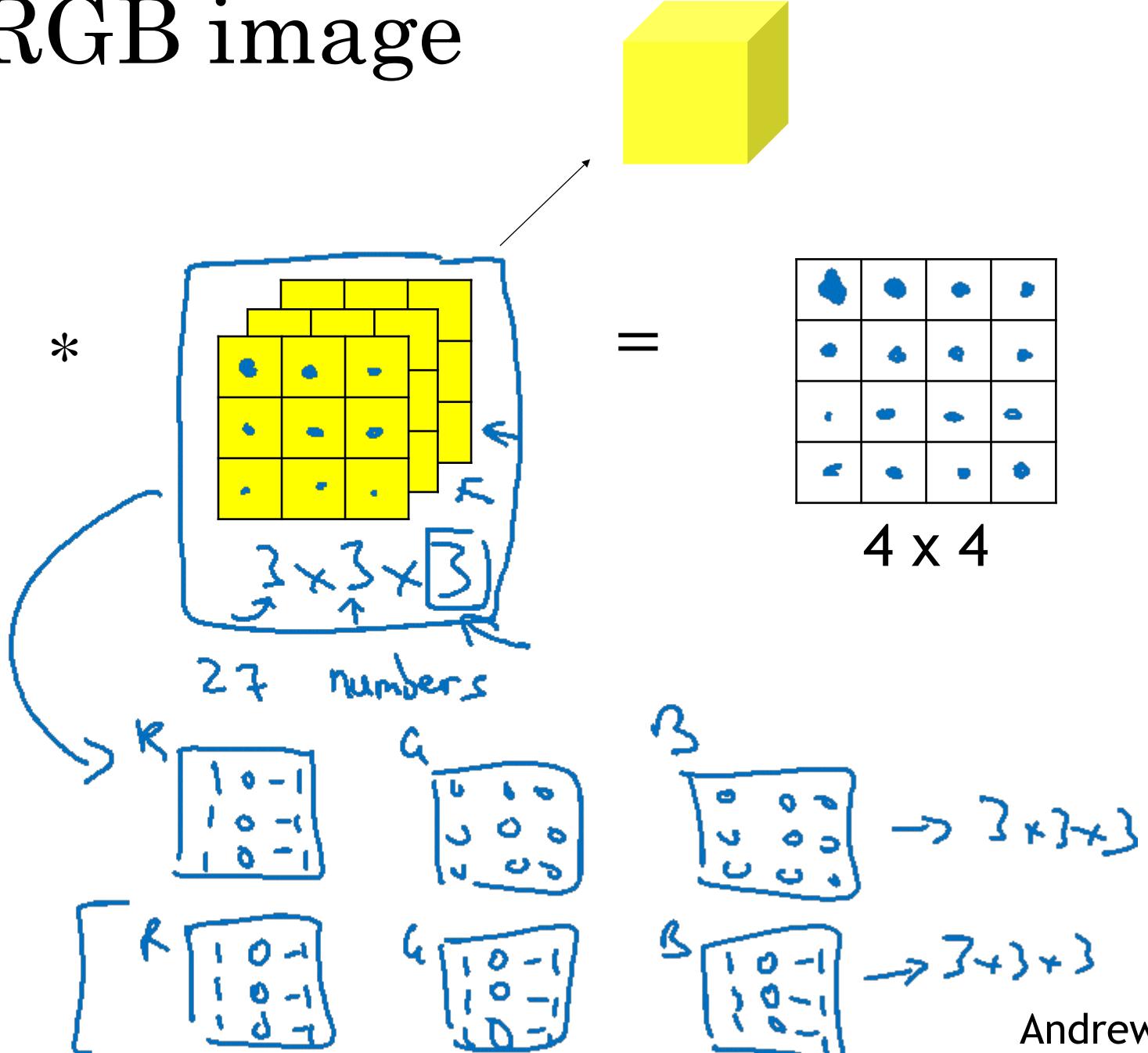


# Convolutions on RGB image

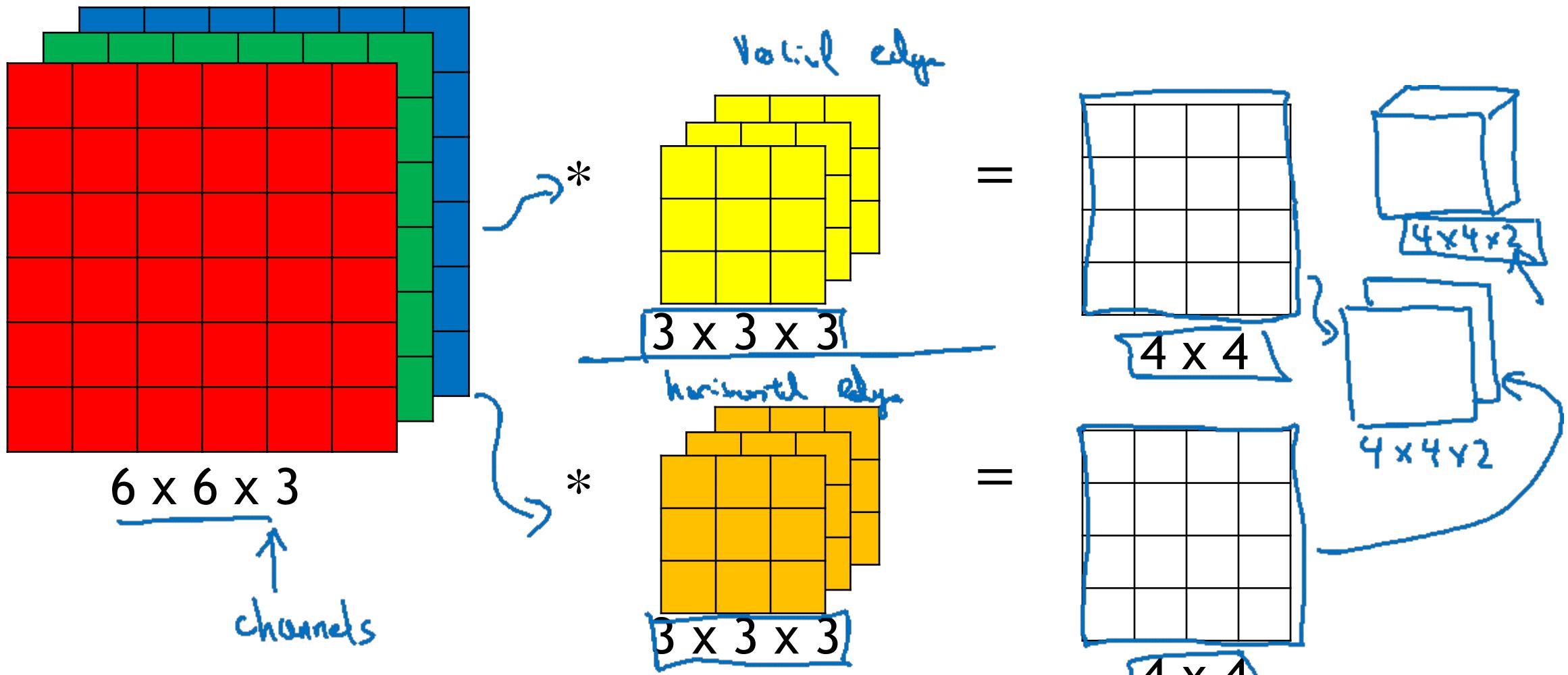


$$6 \times 6 \times 3$$

$$\boxed{\text{ }} * \boxed{\text{ }} = \boxed{\text{ }}$$



# Multiple filters



$$\begin{matrix} \text{Summary: } & n \times n \times n_c & \star f \times f \times n_c & \rightarrow \frac{n-f+1}{4} \times \frac{n-f+1}{4} \times n'_c \\ & 6 \times 6 \times 3 & 3 \times 3 \times 3 & \quad \# \text{filters} \end{matrix}$$

Andrew Ng



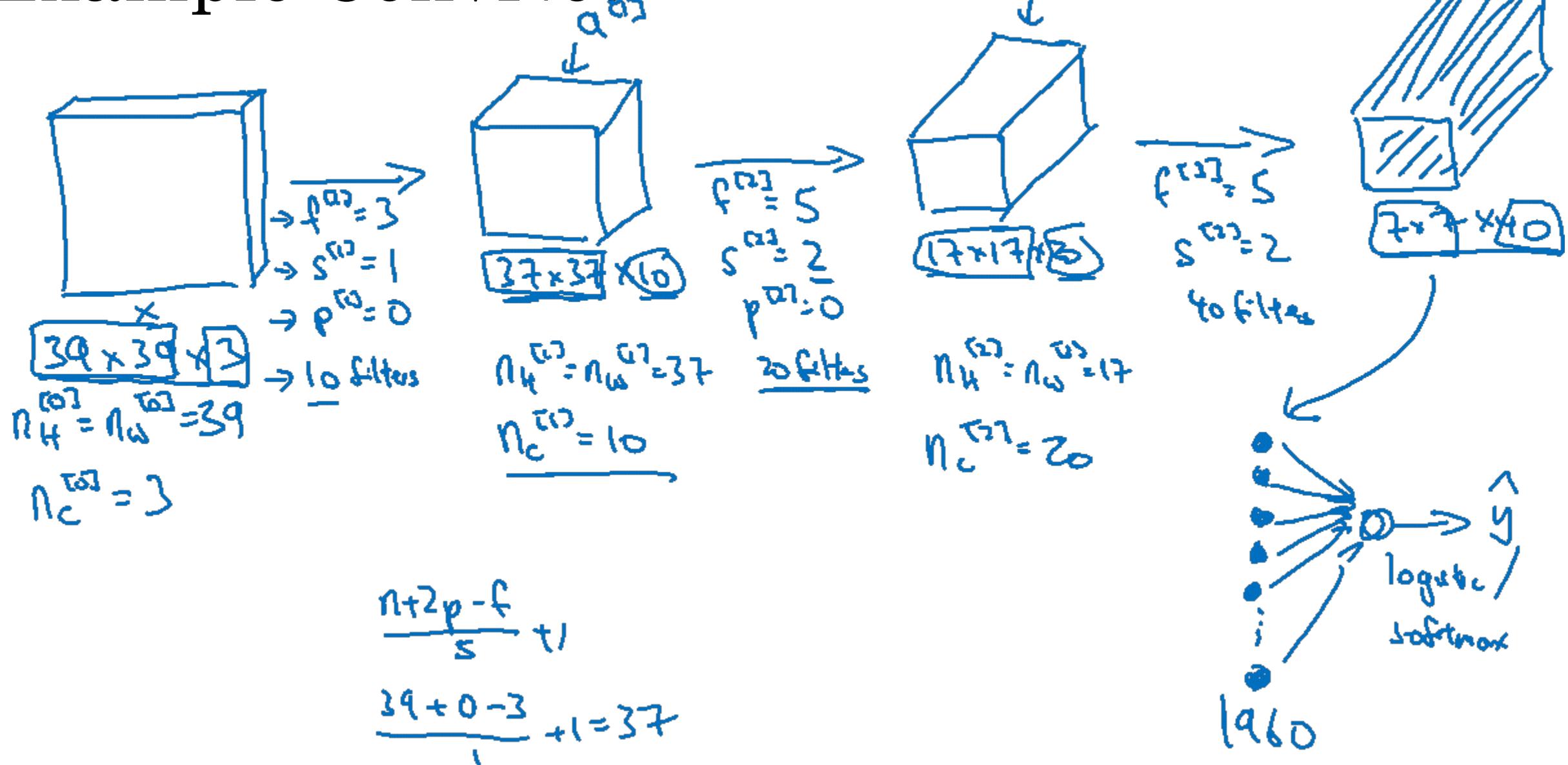
deeplearning.ai

# Convolutional Neural Networks

---

A simple  
convolution  
network example

# Example ConvNet



# Types of layer in a convolutional network:

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



deeplearning.ai

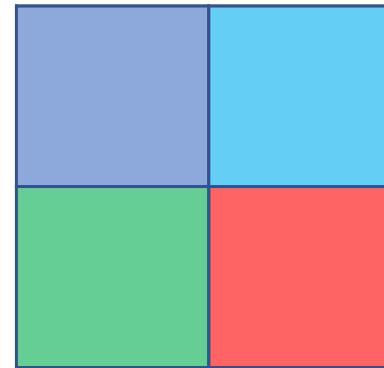
# Convolutional Neural Networks

---

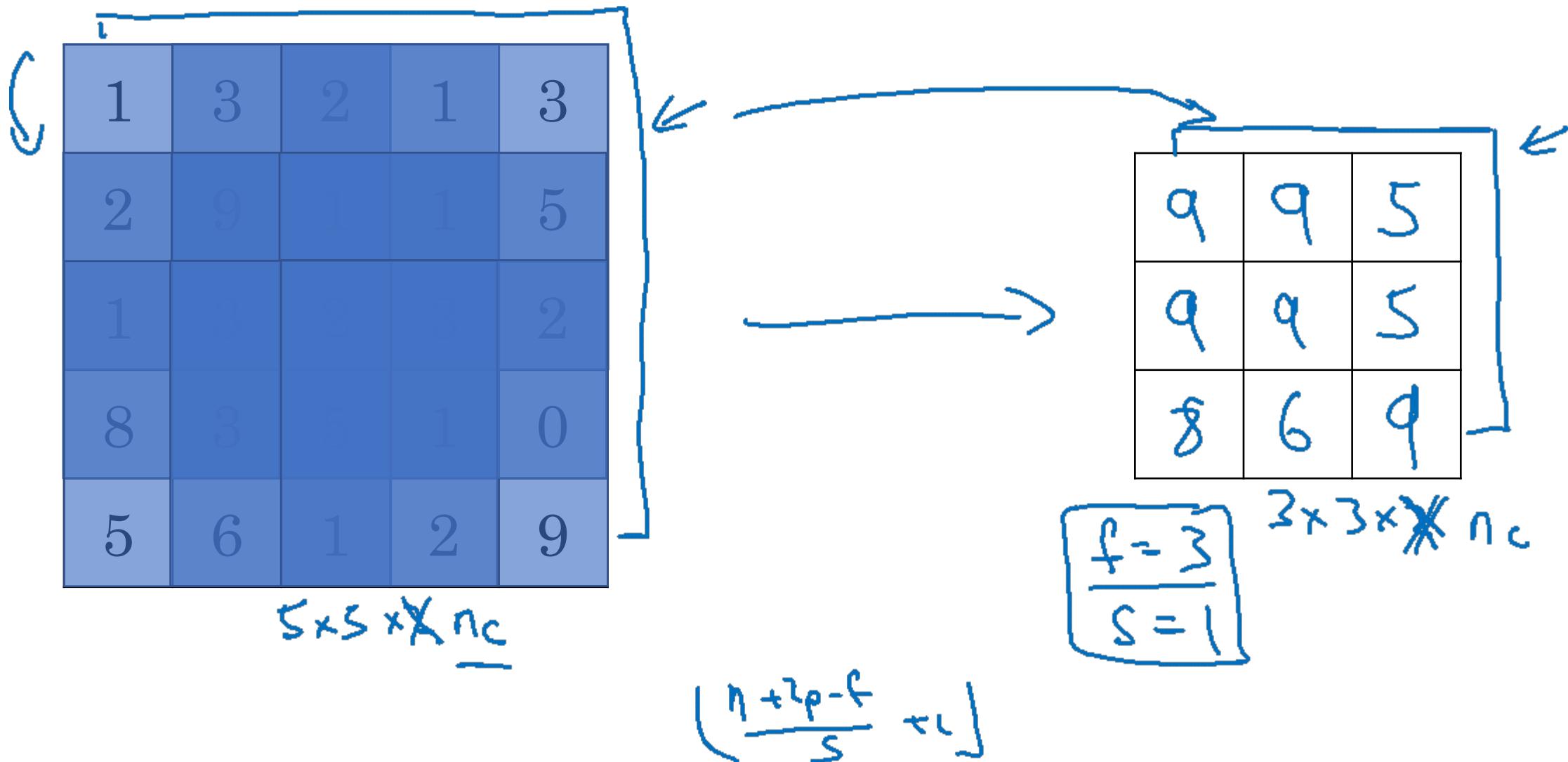
## Pooling layers

# 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

~~p: padding~~

No parameters to learn!

$$n_H \times n_w \times n_c$$

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



deeplearning.ai

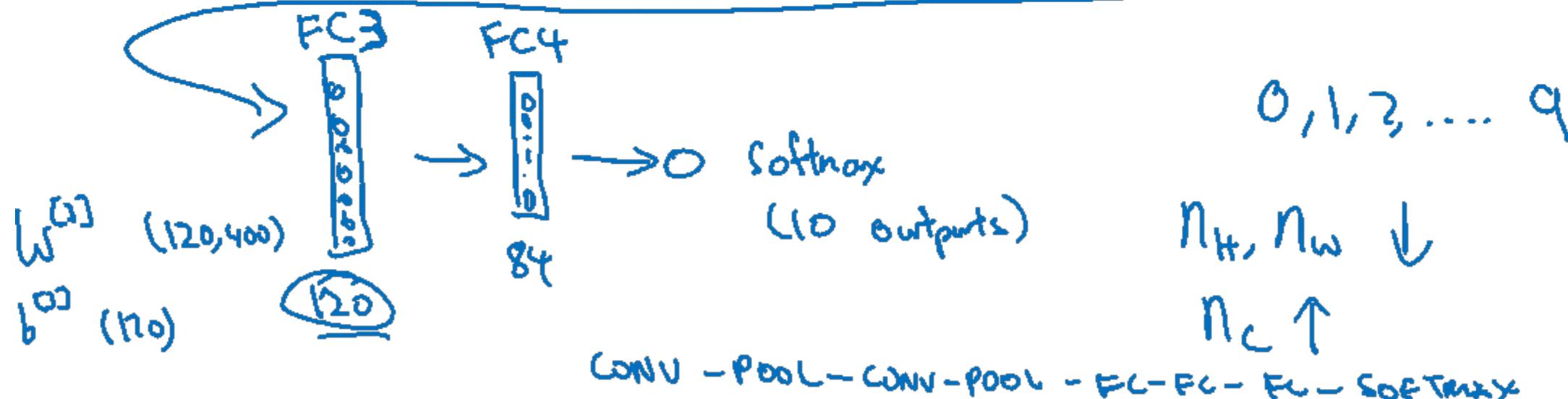
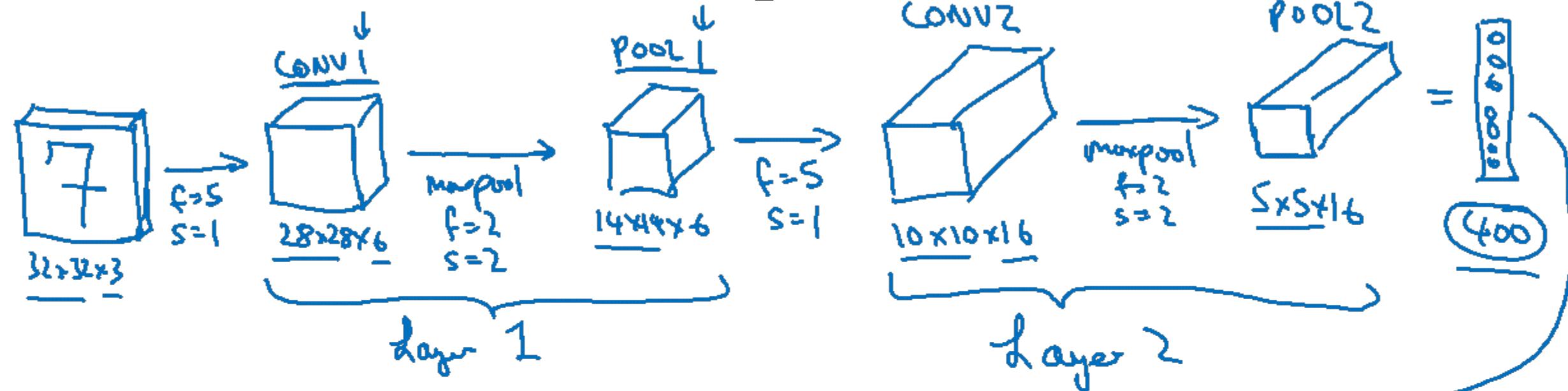
# Convolutional Neural Networks

---

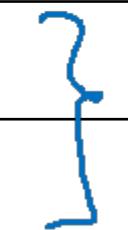
## Convolutional neural network example

# Neural network example

(LeNet-5)



# Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[1]}$	0
			
			
			
			
			



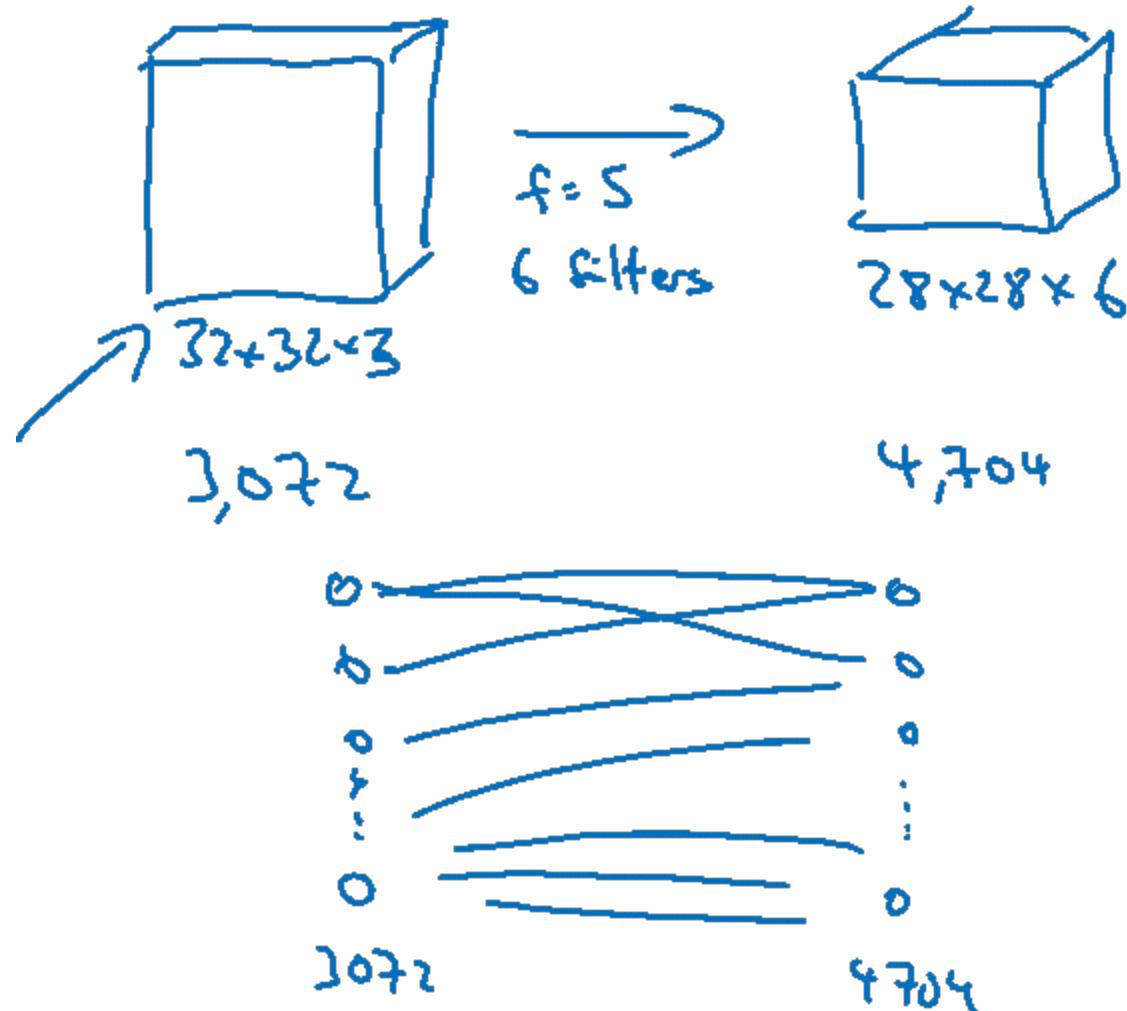
deeplearning.ai

# Convolutional Neural Networks

---

## Why convolutions?

# Why convolutions

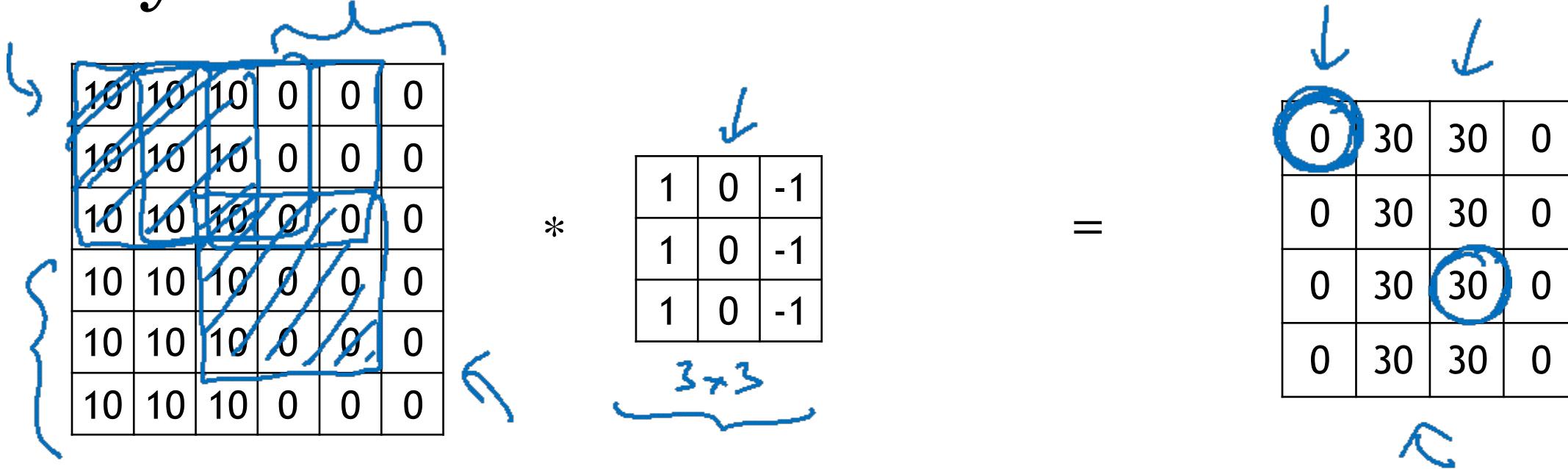


$$5 \times 5 = 25$$

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

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

# Why convolutions

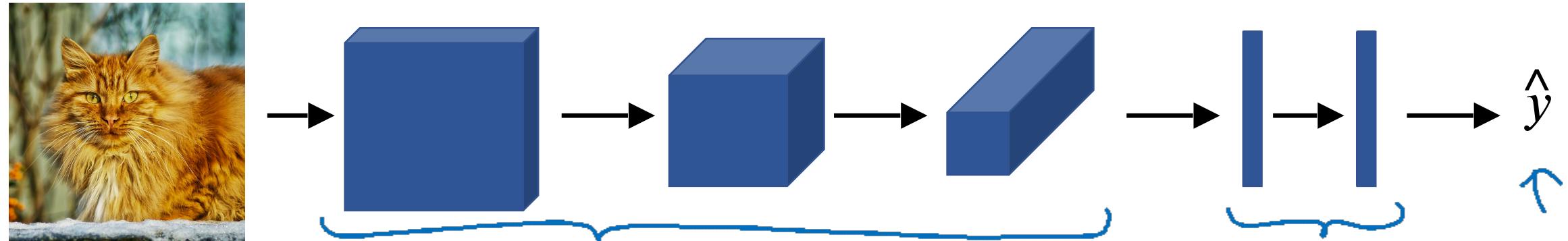


**Parameter sharing:** A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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

# Putting it together

Training set .



Cost

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

Use gradient descent to optimize parameters to reduce