

# Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

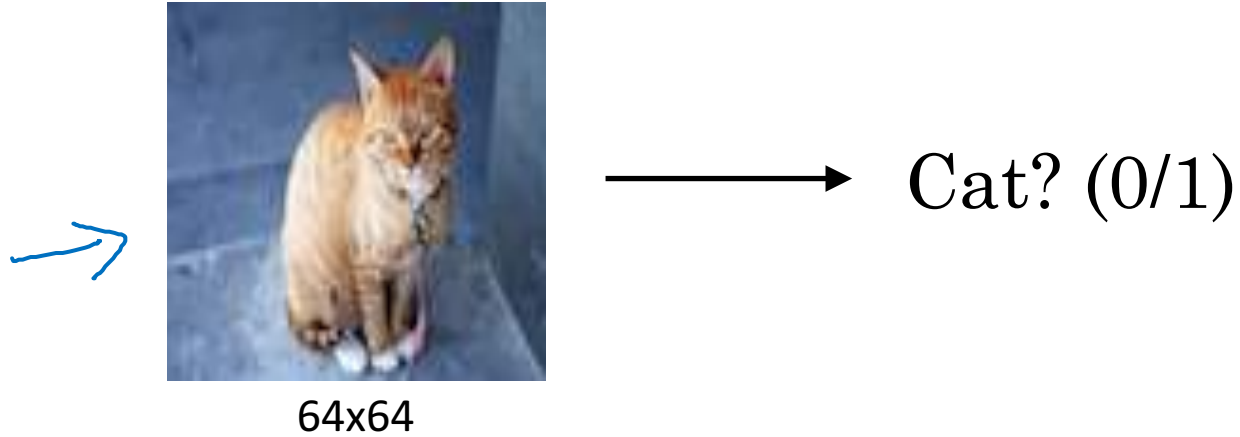
# Convolutional Neural Networks

---

Computer **vision**

# Computer Vision Problems

## Image Classification



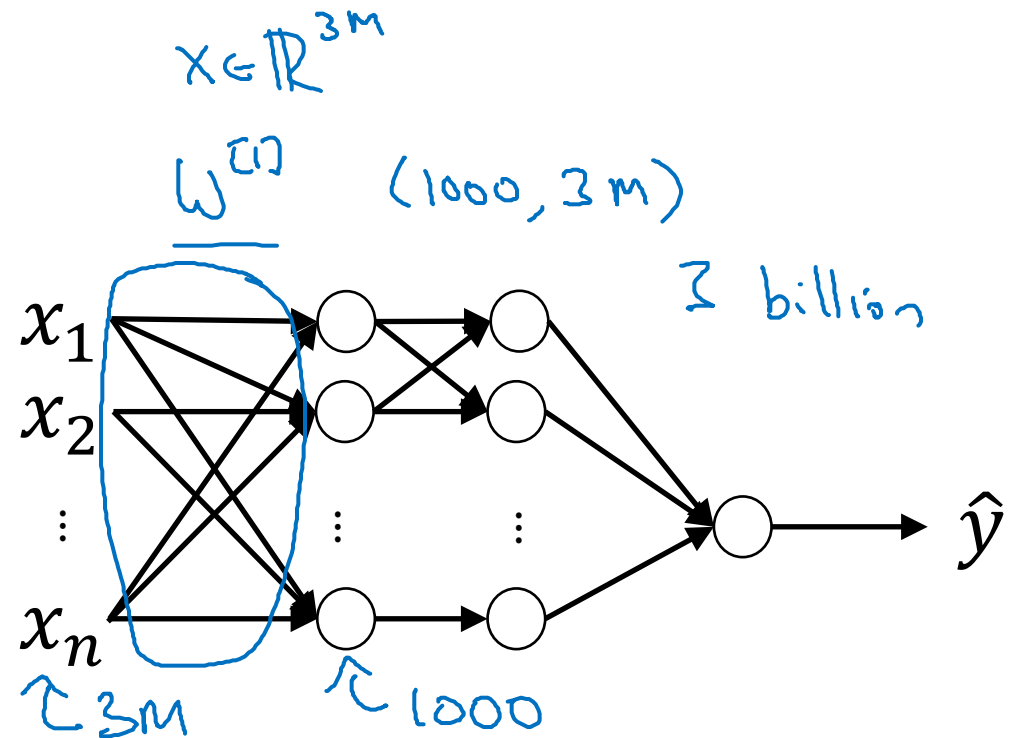
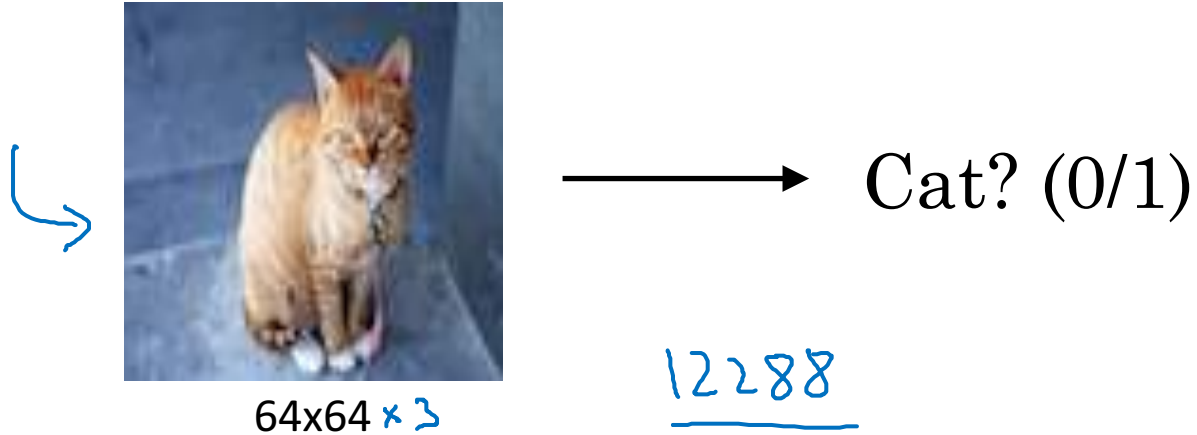
## Neural Style Transfer



## Object detection



# Deep Learning on large images





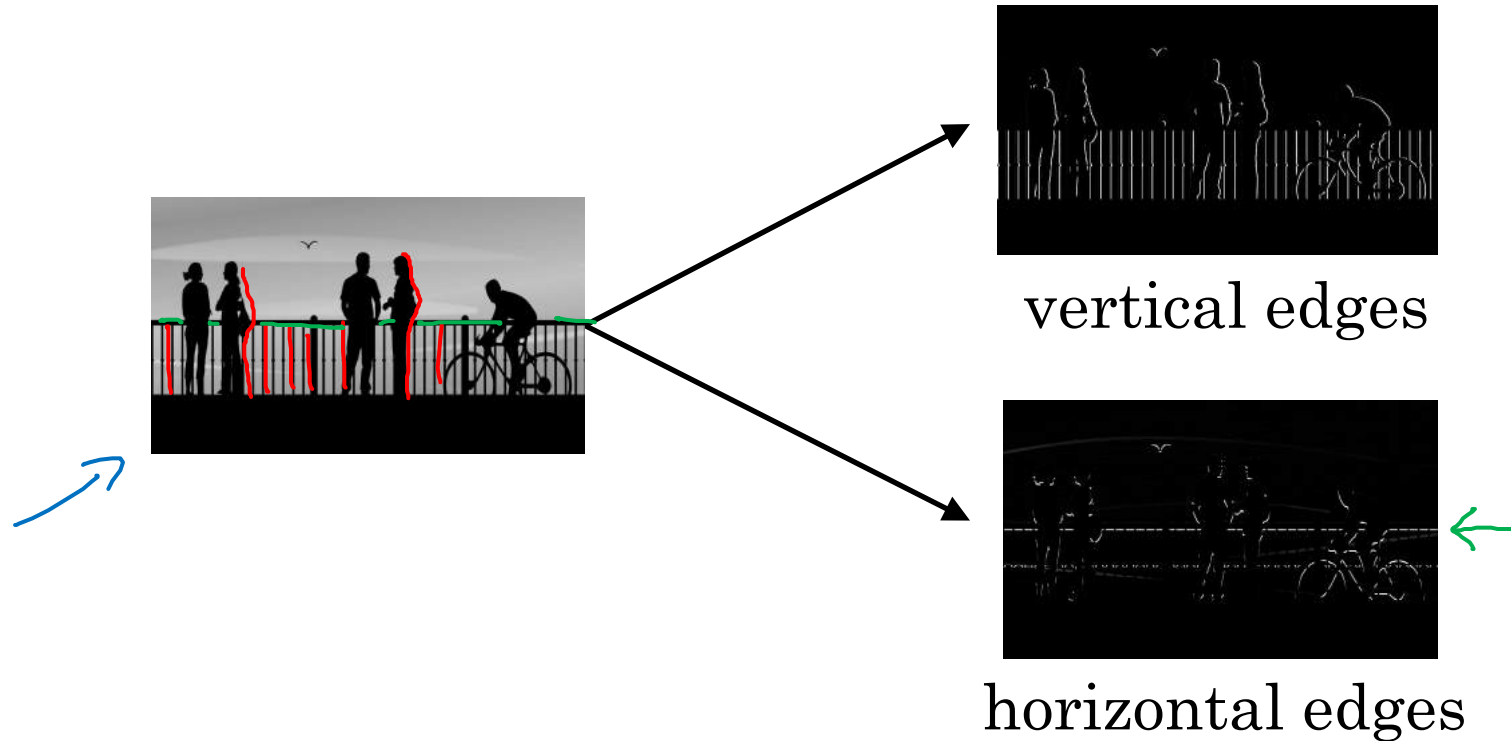
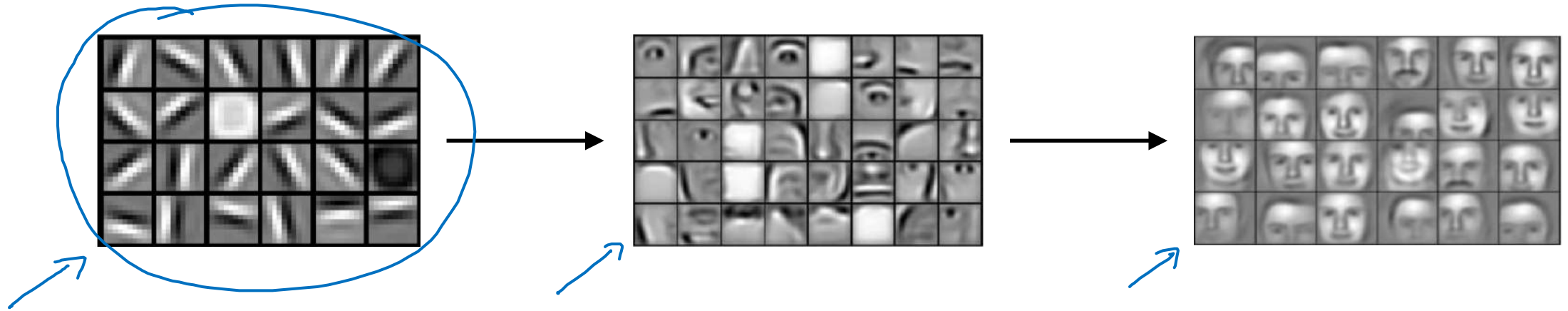
deeplearning.ai

# Convolutional Neural Networks

---

Edge **detection**  
example

# Computer Vision Problem

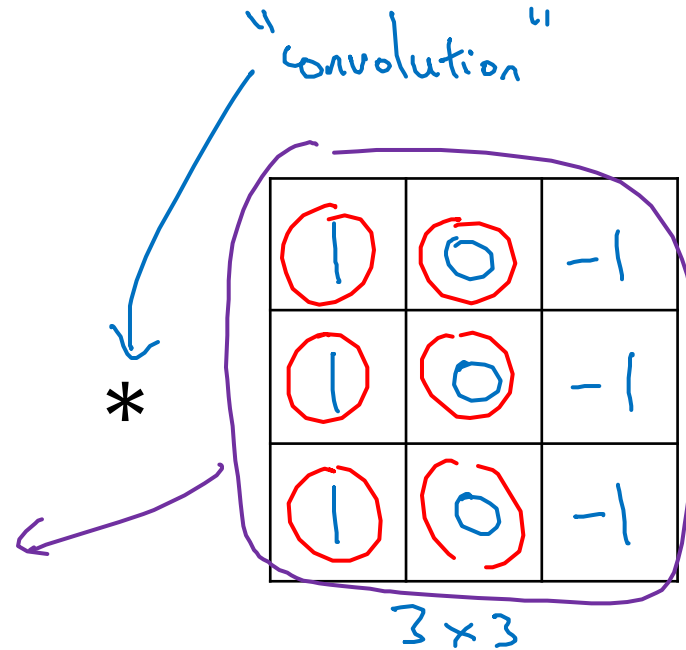


# Vertical edge detection

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

3 <sup>1</sup>	0 <sup>1</sup>	1 <sup>-1</sup>	2 <sup>-1</sup>	7 <sup>-0</sup>	4 <sup>-1</sup>
1 <sup>1</sup>	5 <sup>1</sup>	8 <sup>-1</sup>	9 <sup>-1</sup>	3 <sup>-0</sup>	1 <sup>-1</sup>
2 <sup>1</sup>	7 <sup>1</sup>	2 <sup>-1</sup>	5 <sup>-1</sup>	1 <sup>-0</sup>	3 <sup>-1</sup>
0 <sup>1</sup>	1 <sup>1</sup>	3 <sup>-1</sup>	1 <sup>-1</sup>	7 <sup>-0</sup>	8 <sup>-1</sup>
4	2	1	6	2	8
2	4	5	2	3	9

6x6



=

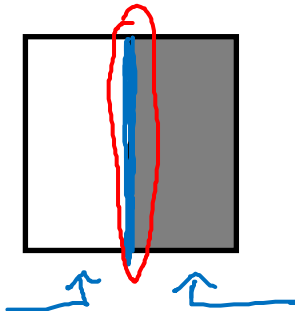
-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4x4

# Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0
10	<u>10</u>	<u>10</u>	<u>0</u>	0	0

6x6



\*

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

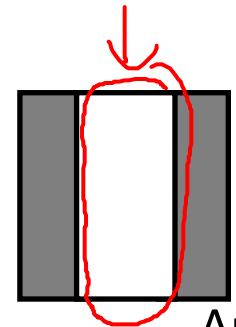
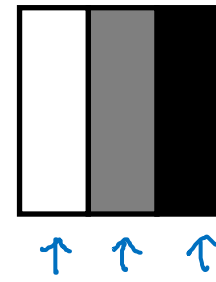
3x3

=

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

4x4

\*







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



\*

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	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




# 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

6x6

\*



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



Sobel filter

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

=

45°  
70°  
73°






deeplearning.ai

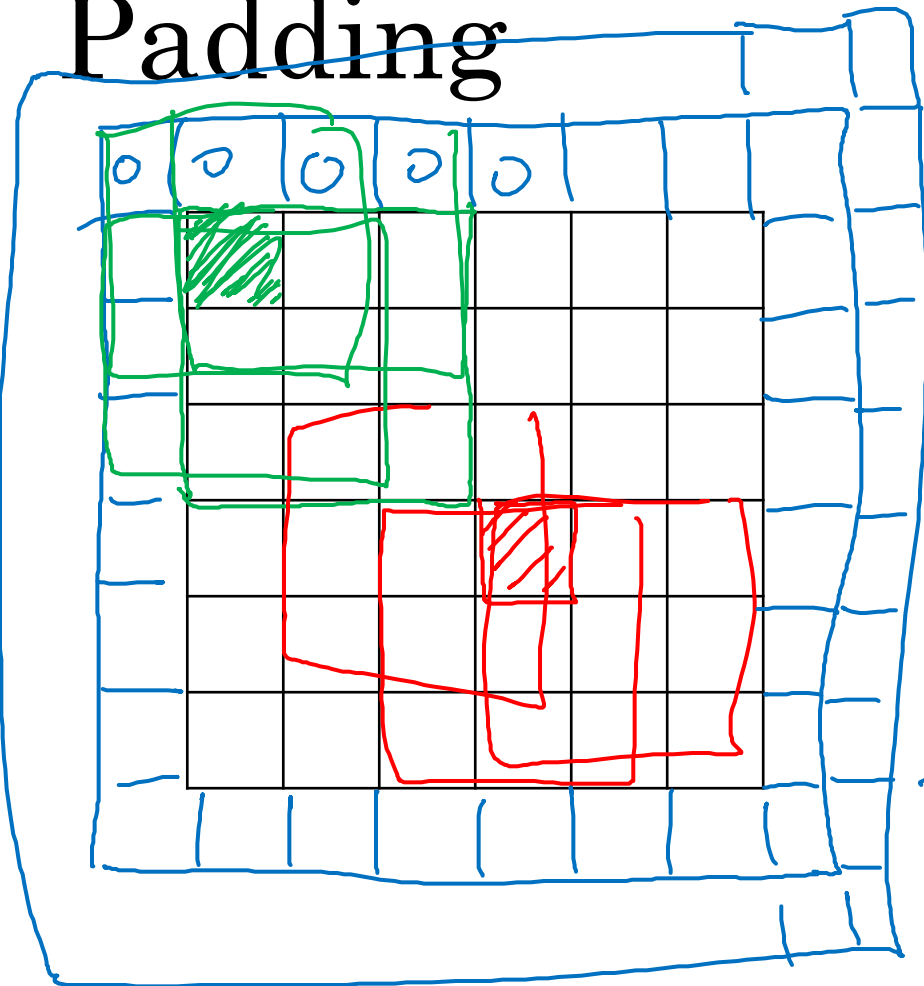
# Convolutional Neural Networks

---

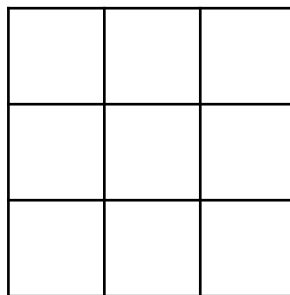
**Padding**

# Padding

- shrinky output
- throw away info from edge

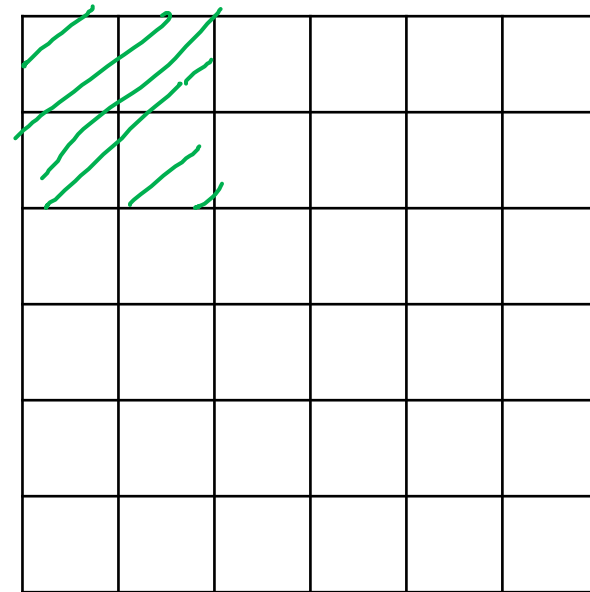


\*



3x3  
f x f

=



6x6

6x6 → 8x8  
n x n

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

$6 - 3 + 1 = 4$

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

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

$6 + 2 - 3 + 1 \times \underline{\underline{4}} = 6 \times 6$

# Valid and Same convolutions

→ no padding

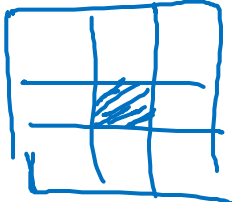
“Valid”:  $n \times n$   $\times$   $f \times f$   $\rightarrow \frac{n-f+1}{1} \times n-f+1$   
 $6 \times 6$   $\times$   $3 \times 3$   $\rightarrow 4 \times 4$

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

$$n+2p-f+1 \times n+2p-f+1$$
$$\cancel{n+2p-f+1} = \cancel{n} \Rightarrow \boxed{p = \frac{f-1}{2}}$$
$$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad \left| \begin{array}{l} 5 \times 5 \\ f=5 \end{array} \right.$$

$f$  is usually odd

1x1  
3x3  
5x5  
7x7



$p=2$



deeplearning.ai

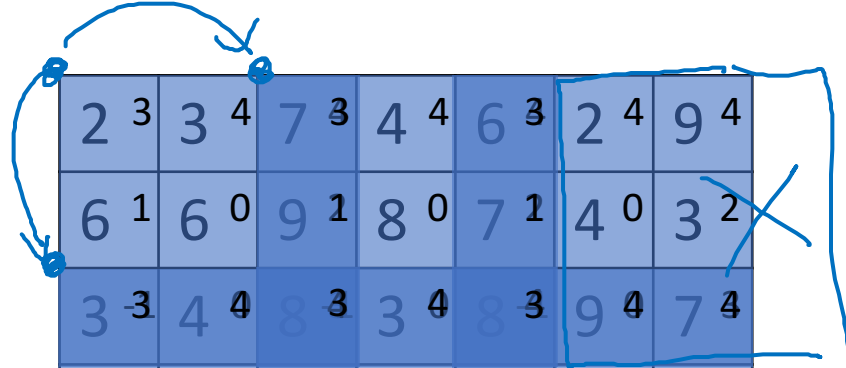
# Convolutional Neural Networks

---

Strided  
convolutions



# Strided convolution



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

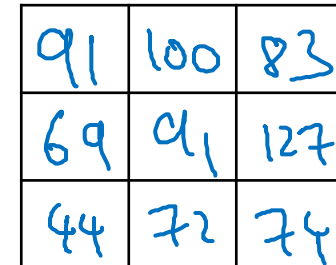
7x7

3	4	4
1	0	2
-1	0	3

3x3

Stride = 2

=



91	100	83
69	91	127
44	72	74

3x3

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

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

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

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

# Summary of convolutions

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

padding  $p$       stride  $s$

Output Size:

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

# Technical note on cross-correlation vs. convolution

Convolution in math textbook:

2 <sup>7</sup>	3 <sup>2</sup>	7 <sup>5</sup>	4	6	2
6 <sup>9</sup>	6 <sup>0</sup>	9 <sup>4</sup>	8	7	4
3 <sup>-1</sup>	4 <sup>1</sup>	8 <sup>3</sup>	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


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



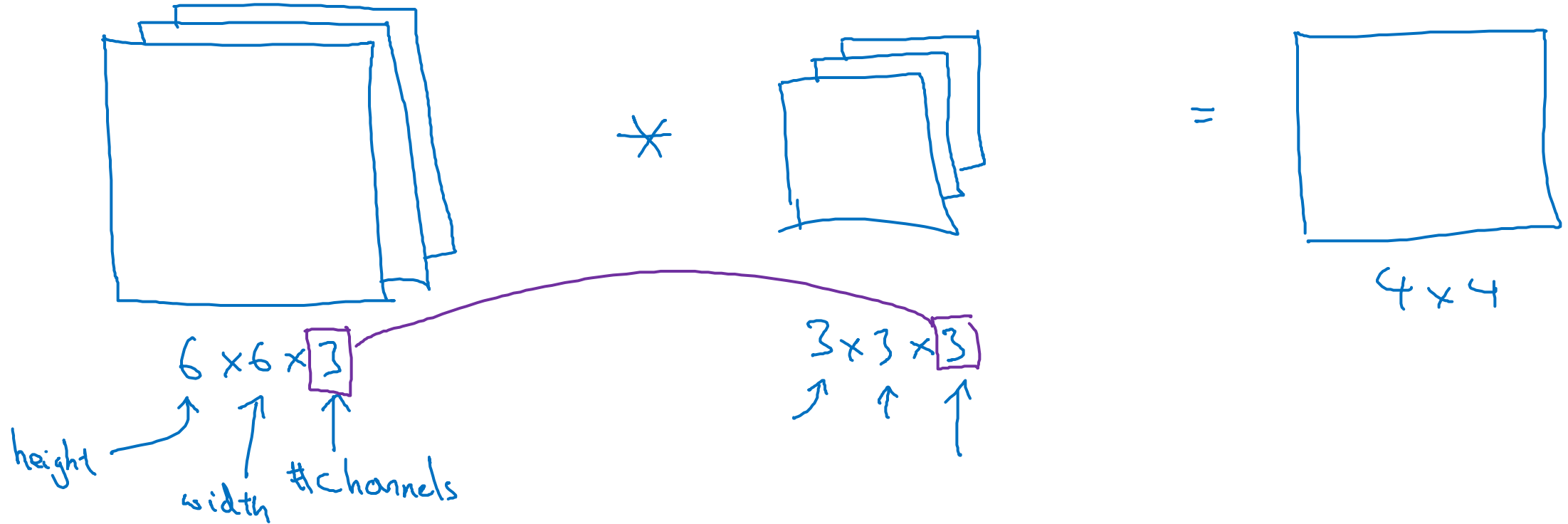
deeplearning.ai

# Convolutional Neural Networks

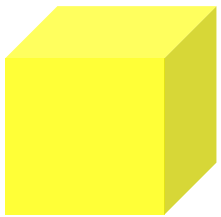
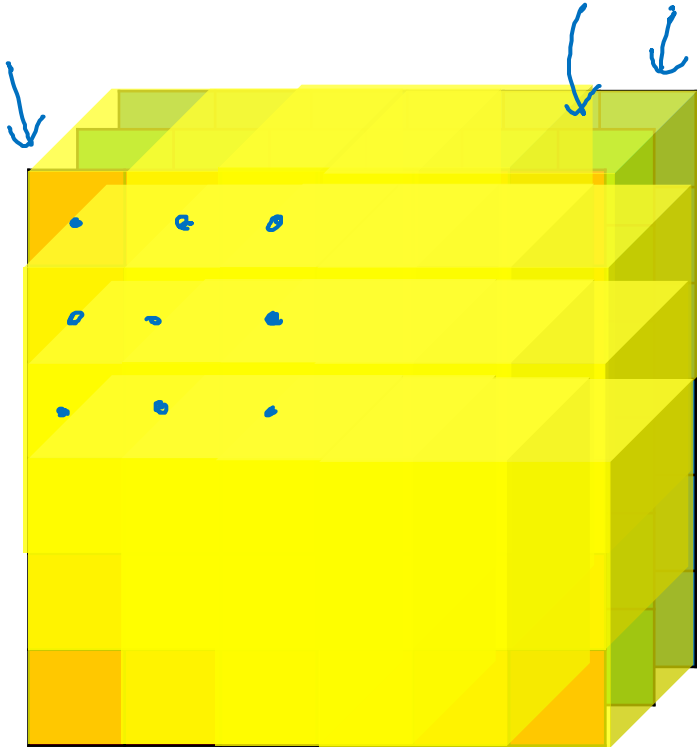
---

Convolutions over  
**volumes**

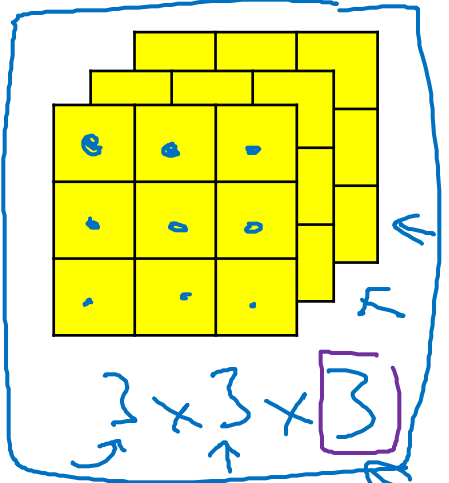
# Convolutions on RGB images



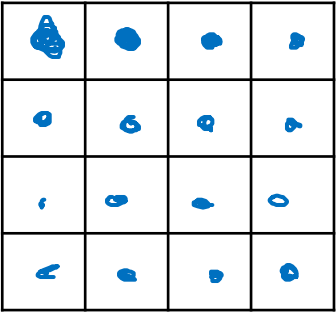
# Convolutions on RGB image



\*

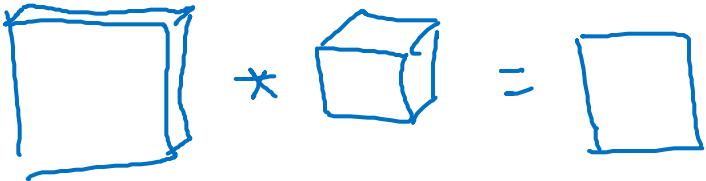


=

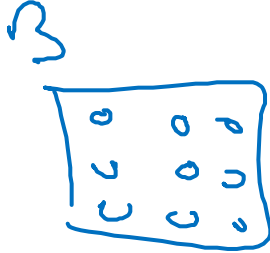
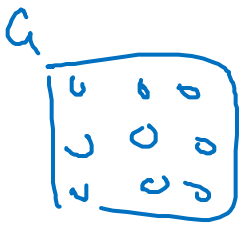
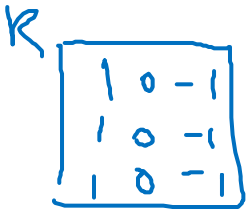


4 x 4

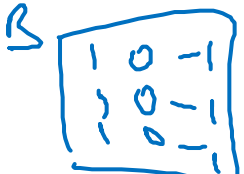
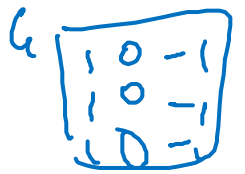
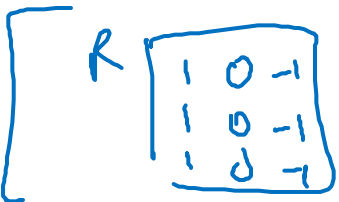
$6 \times 6 \times 3$   
↑    ↑    ↑



27 numbers

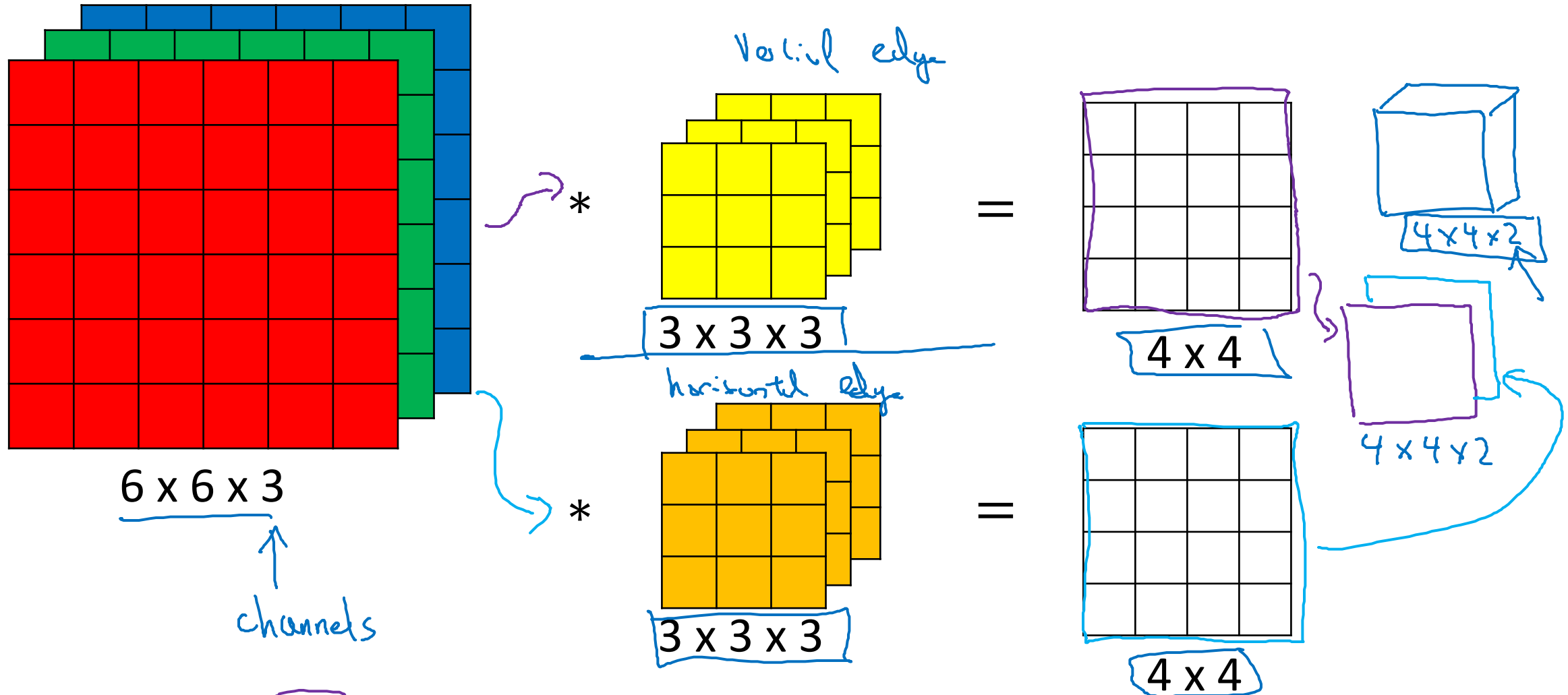


→ 3x3x3



→ 3x3x3

# Multiple filters



Summary:

$$n \times n \times n_c \times f \times f \times n_c \rightarrow \frac{n-f+1}{4} \times \frac{n-f+1}{4} \times n_c' \times 2$$

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



deeplearning.ai

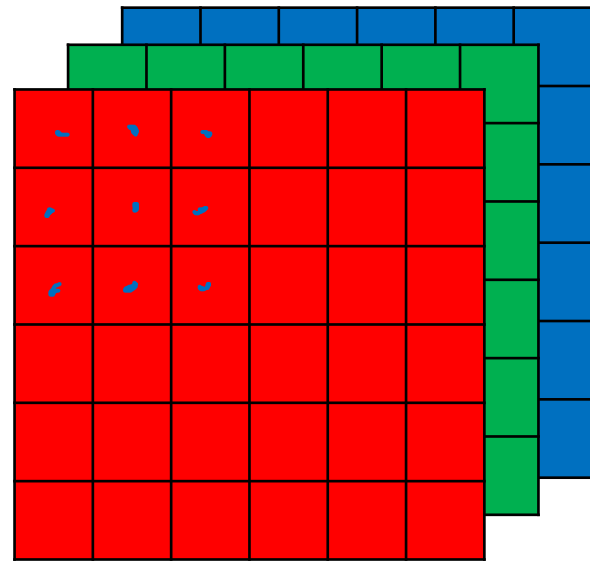
# Convolutional Neural Networks

---

One **layer** of a  
convolutional  
network



# Example of a layer



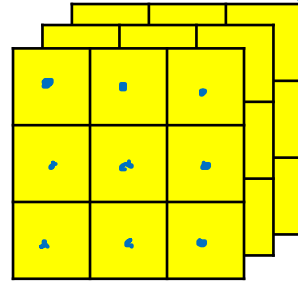
6 x 6 x 3

$a^{[0]}$

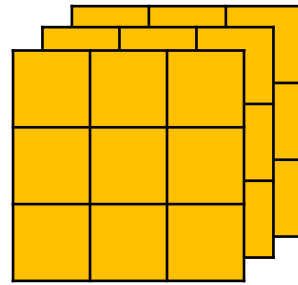
$$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

\*  
\*

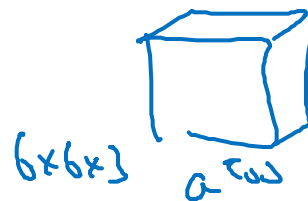


3 x 3 x 3



3 x 3 x 3

" $W^{[1]}$ "



6 x 6 x 3



4 x 4 x 2

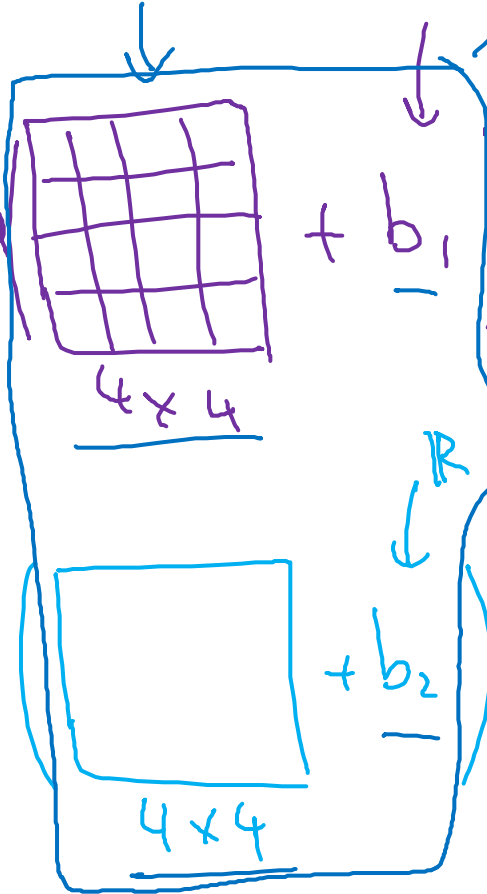
10 filters

" $W^{[1]} a^{[0]}$ "

$R$

$z^{[1]}$

Relu



+  $b_1$

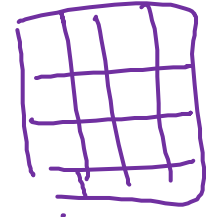
4 x 4

$R$

+  $b_2$

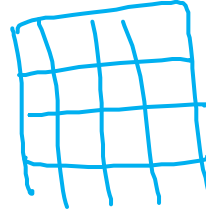
4 x 4

→

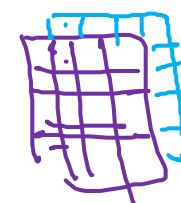


4 x 4

→



4 x 4



4 x 4 x 2

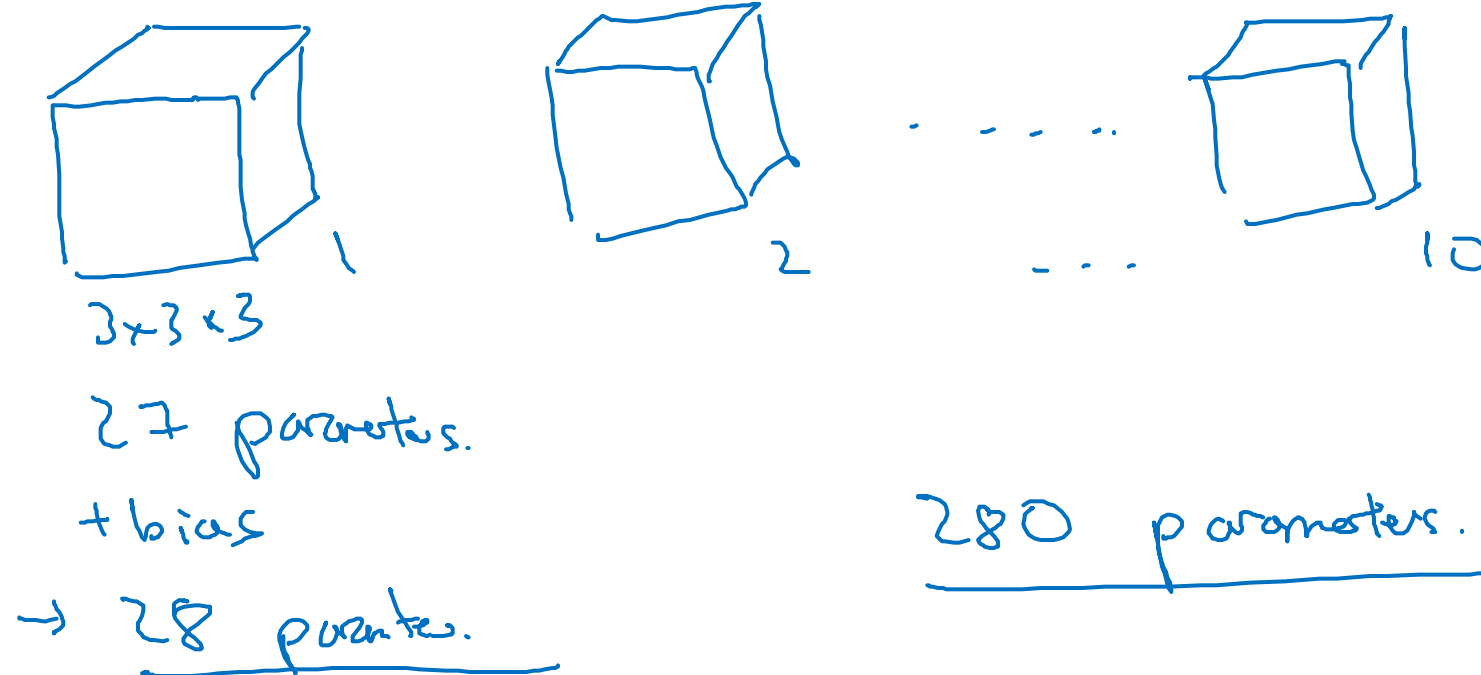
←

$a^{[1]}$

4 x 4 x 10

# 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?



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

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

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

bias:  $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$  ← #filters in layer l.

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

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

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

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



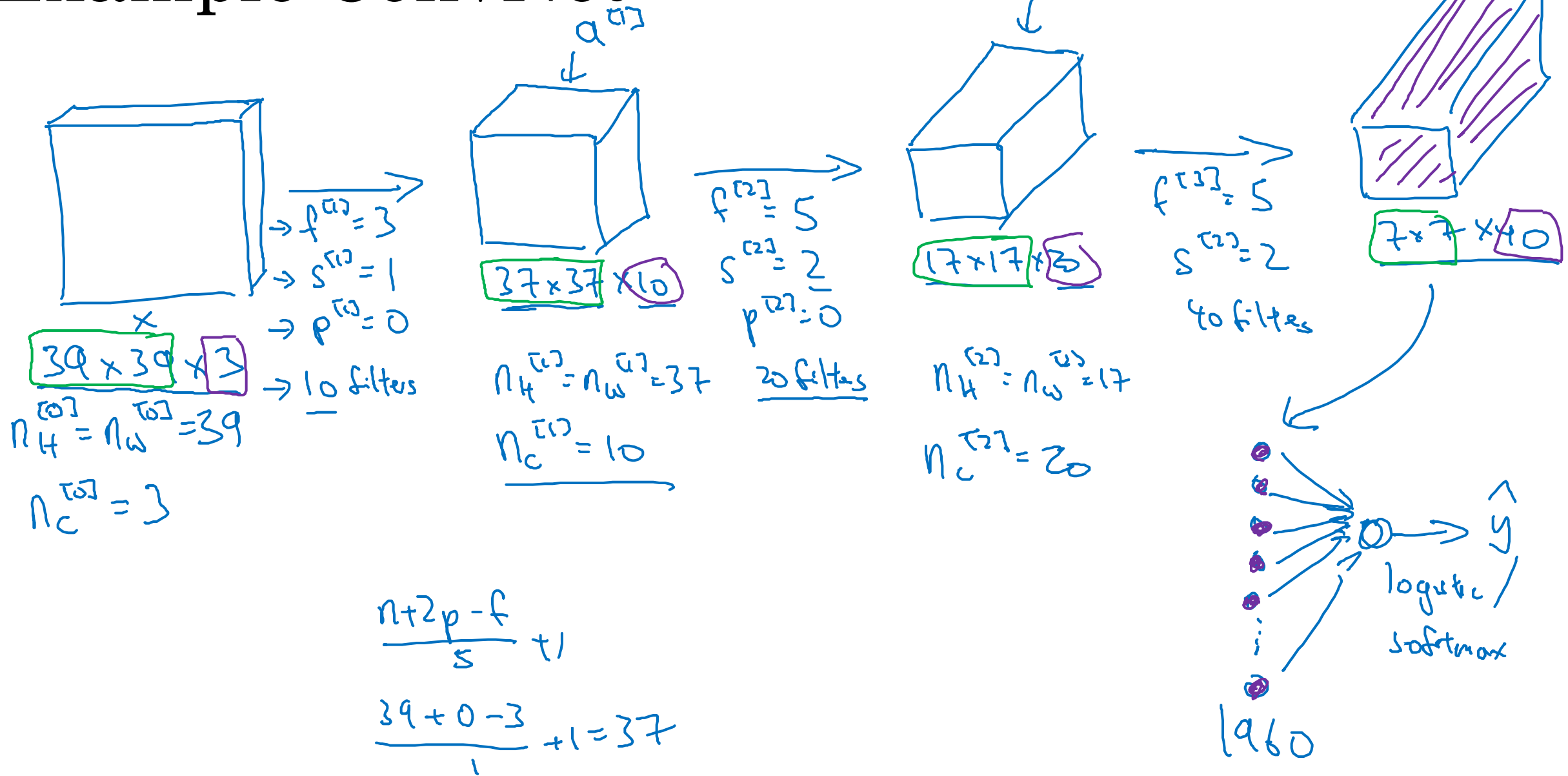
deeplearning.ai

# Convolutional Neural Networks

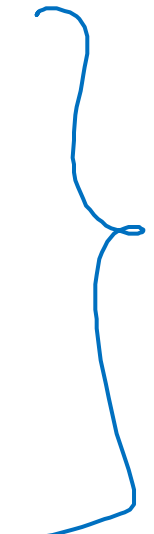
---

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

---

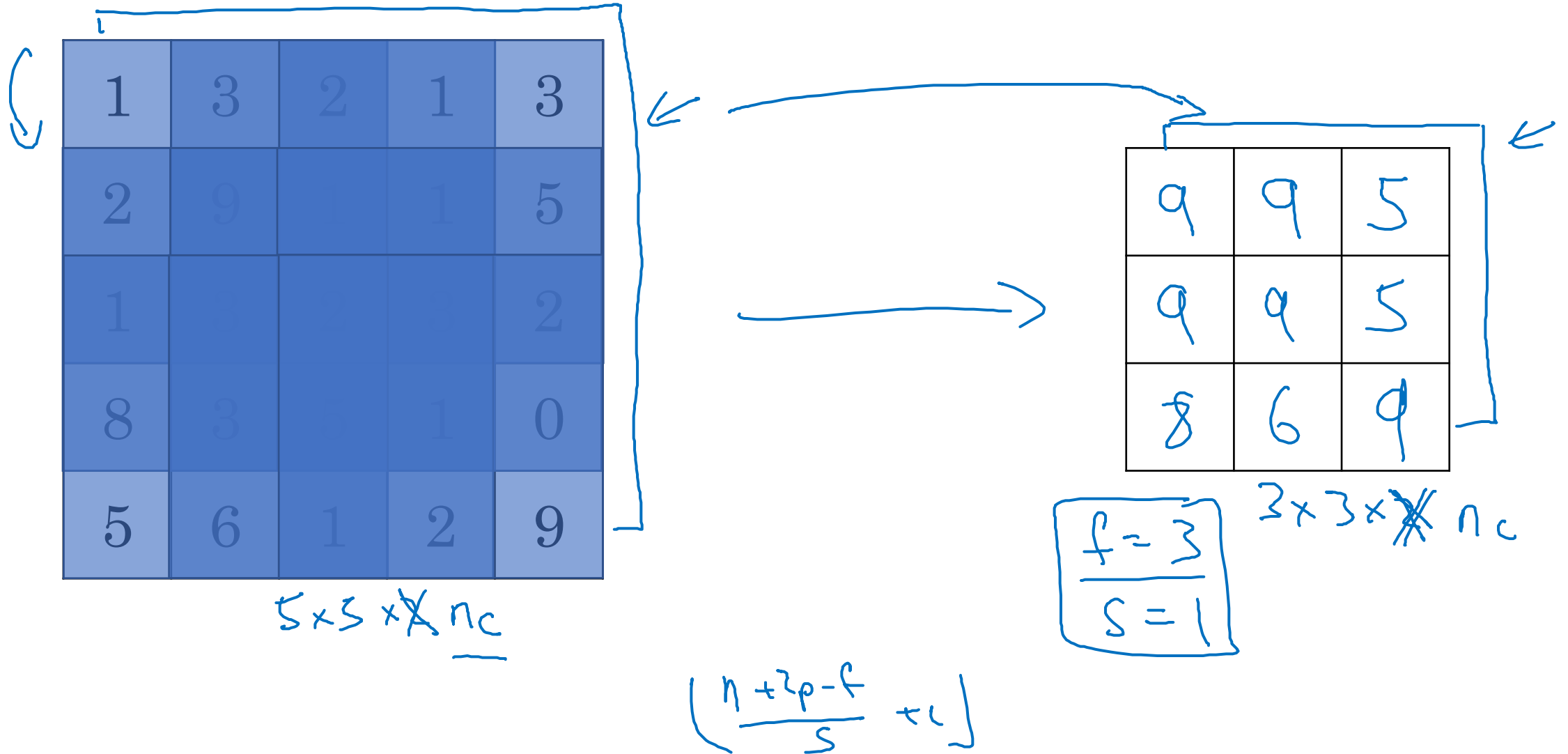
## Pooling layers

# 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

$$f=2$$

$$s=2$$

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

# 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!

$$n_H \times n_W \times \underline{n_C}$$



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



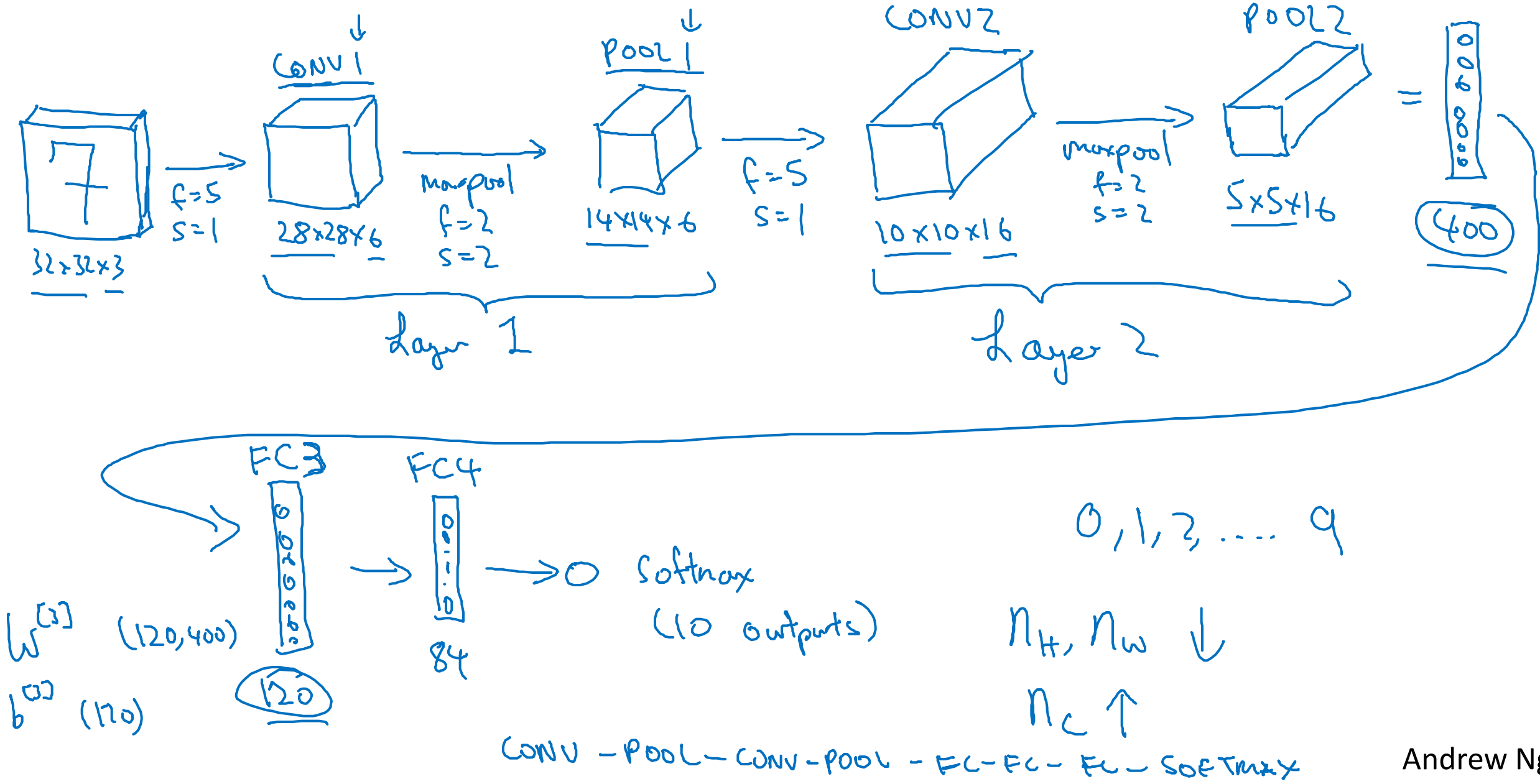
deeplearning.ai

# 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 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	<u>6,272</u>	608 ←
POOL1	(14,14,8)	<u>1,568</u>	0 ←
CONV2 (f=5, s=1)	(10,10,16)	<u>1,600</u>	3216 ←
POOL2	(5,5,16)	<u>400</u>	0 ←
FC3	(120,1)	<u>120</u>	48120 }
FC4	(84,1)	<u>84</u>	10164 }
Softmax	(10,1)	<u>10</u>	850



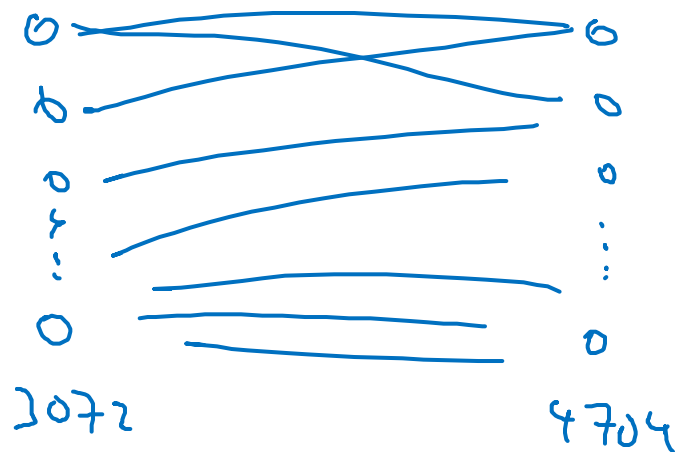
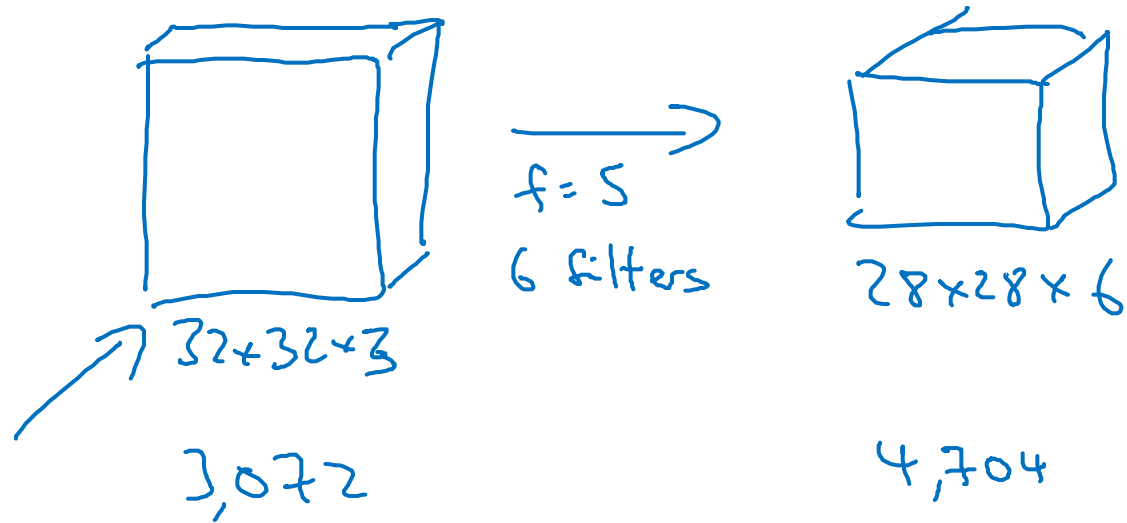
deeplearning.ai

# Convolutional Neural Networks

---

## Why convolutions?

# Why convolutions



$$5 \times 5 = 25$$

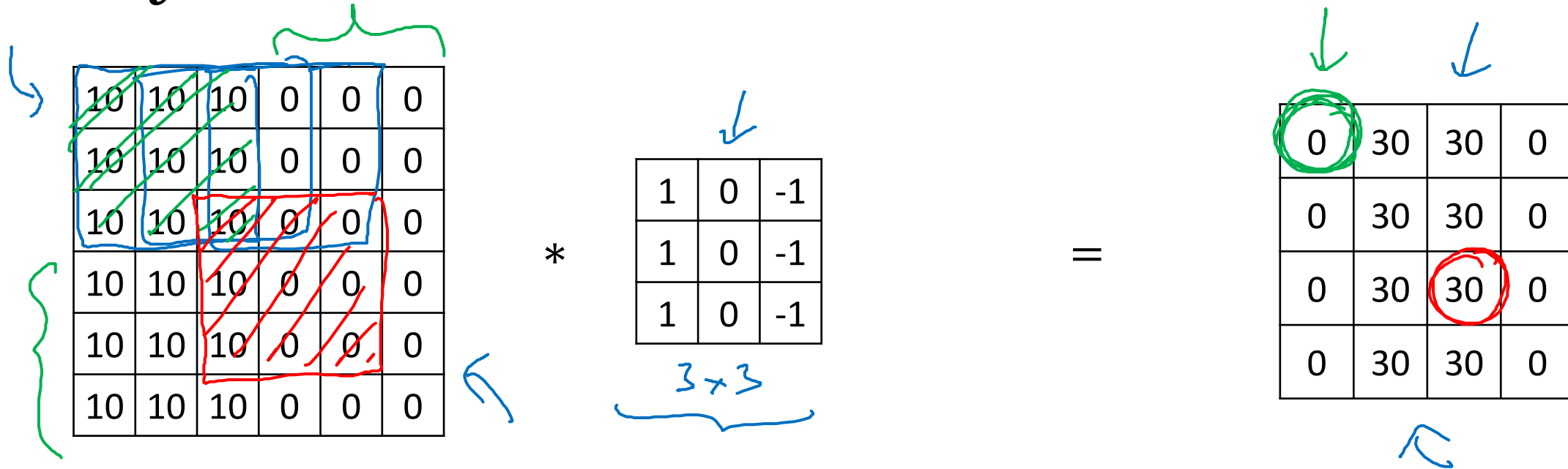
$$26$$

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

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



# Why convolutions

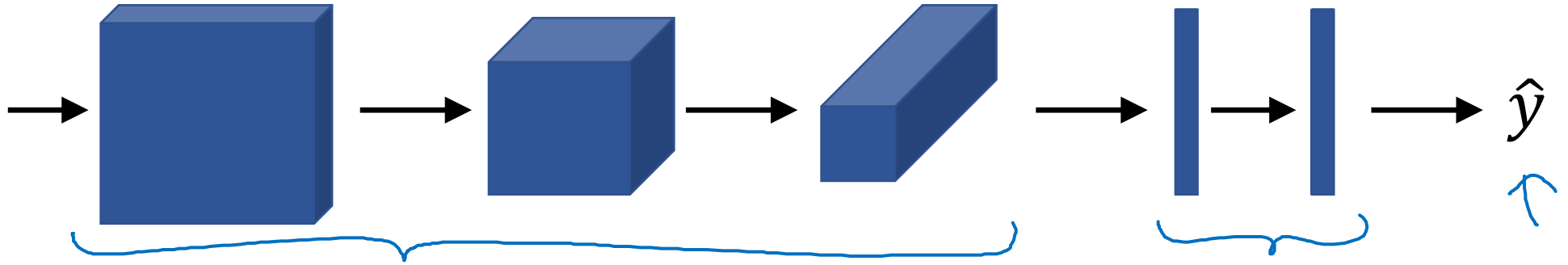


**Parameter sharing:** A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **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  $J$