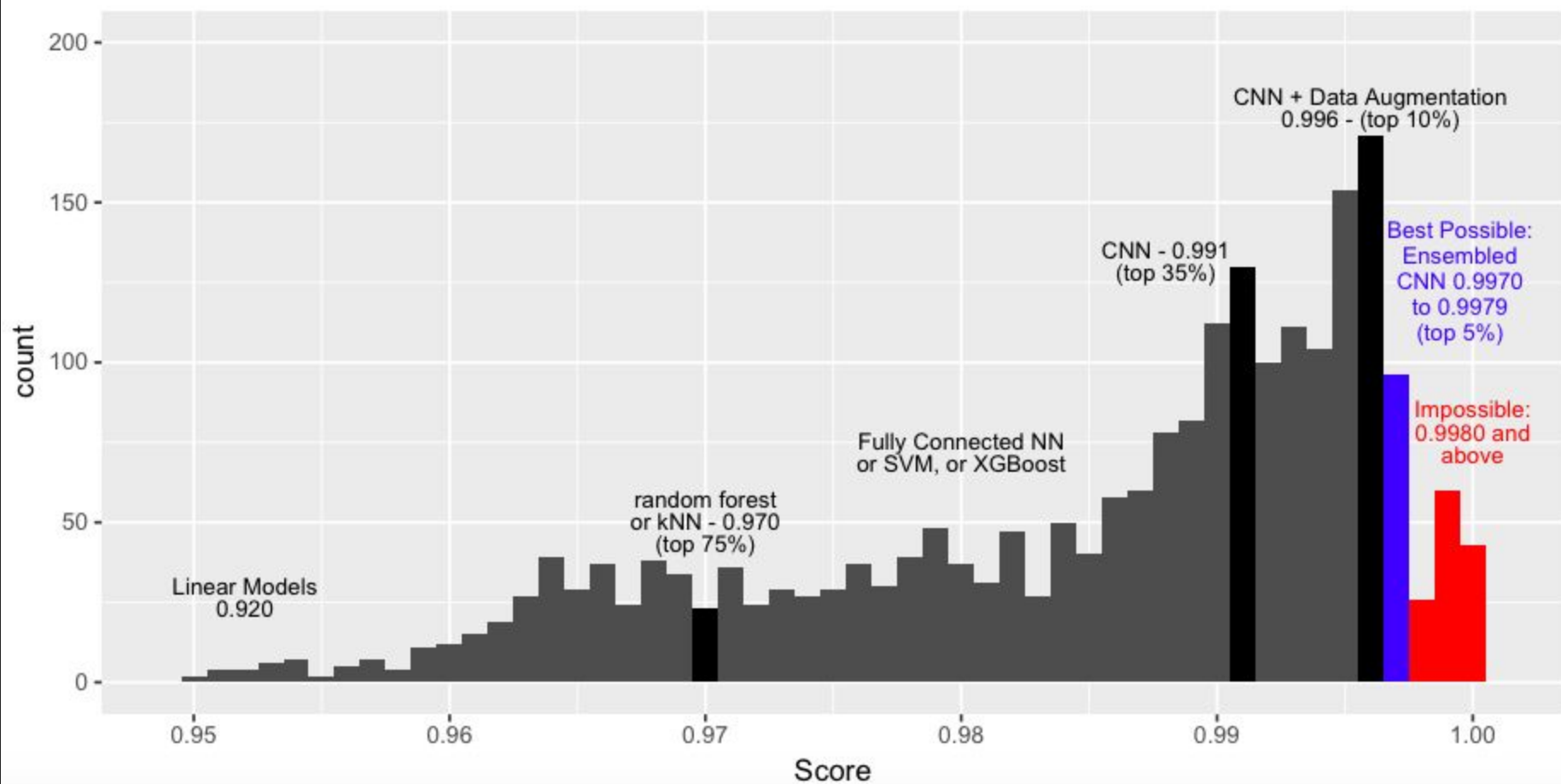


# Convolutional Neural Networks

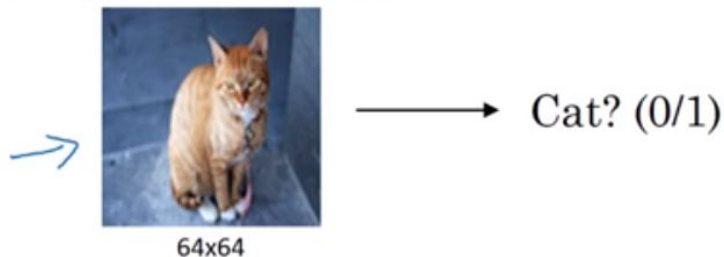
Sources: Elements of Stat. Learning, Andrew Ng, A. Mueller

Histogram of Kaggle MNIST public leaderboard scores, July 15 2018



# Computer Vision Problems:

Image Classification



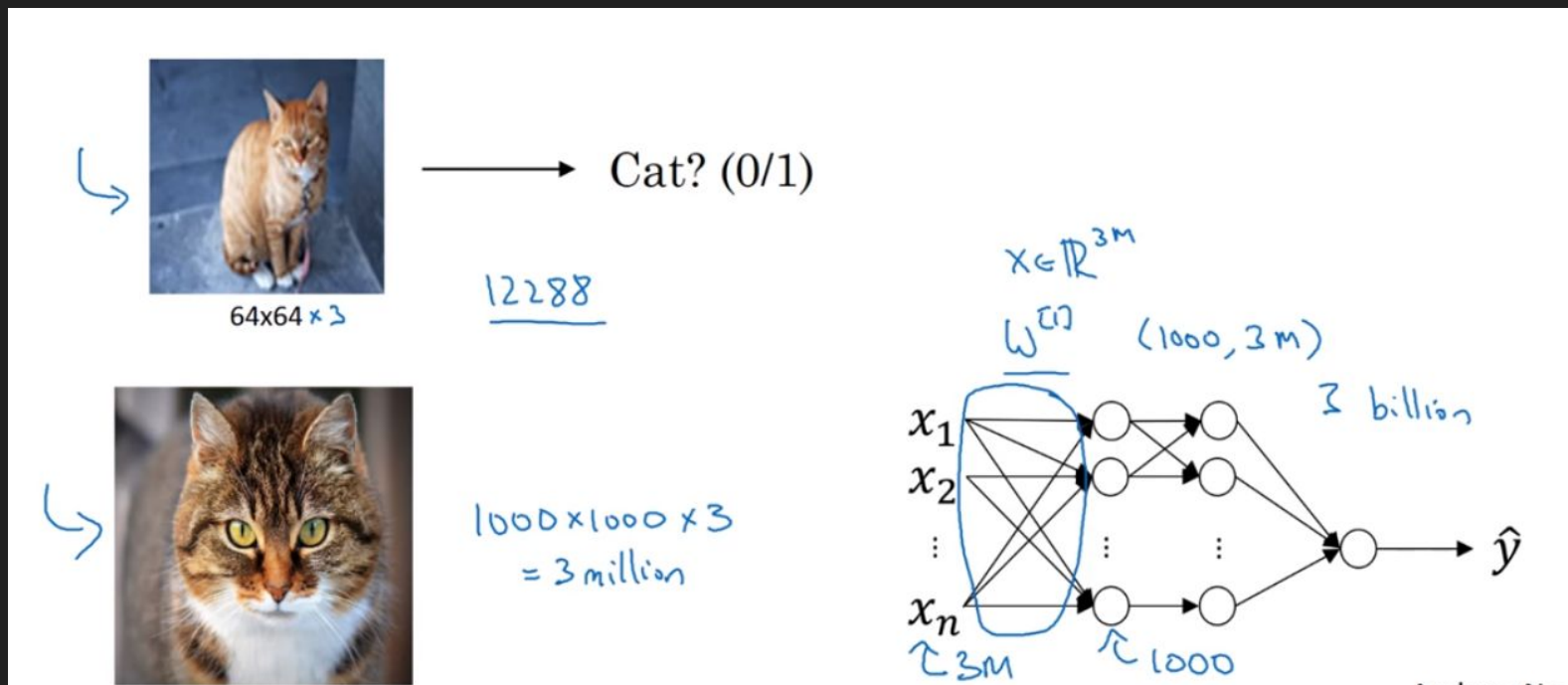
Neural Style Transfer



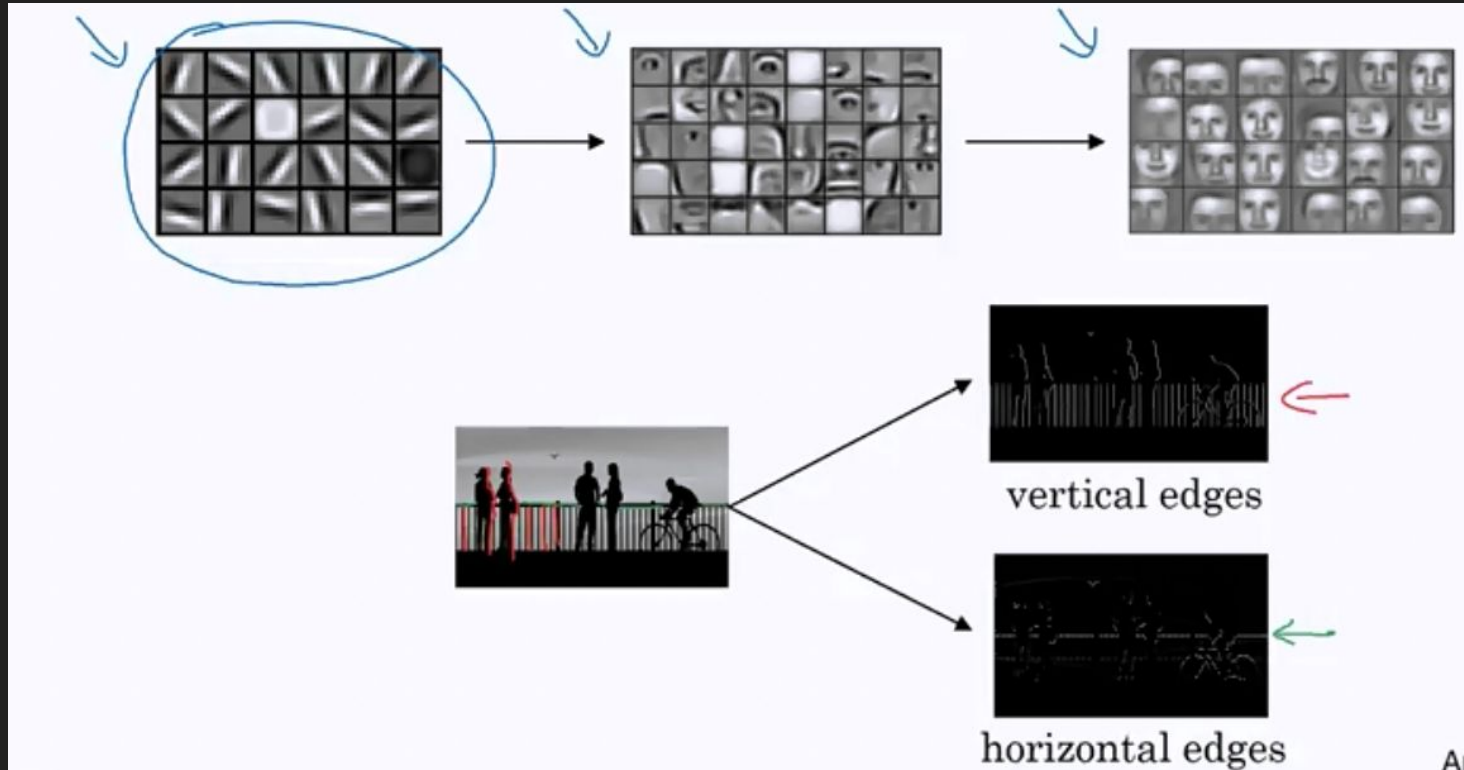
Object detection



# Deep Learning on Larger Images:



# Edge Detection:

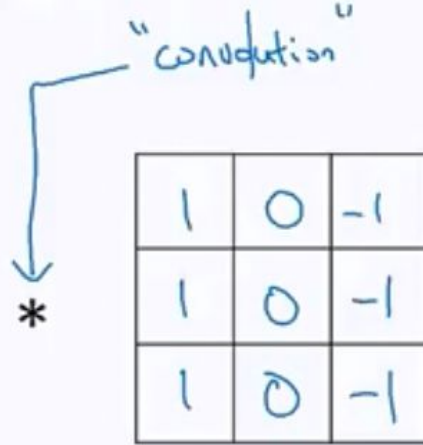


# Vertical edge detection

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

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

6x6



3x3  
filter

=

-5			

4x4

# Vertical edge detection

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

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

6x6

"convolution"

\*

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

3x3  
filter

=

<u>-5</u>	<u>-4</u>		

4x4

# 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	<u>0</u>	<u>1</u>	<u>2</u>	<u>7</u>	<u>4</u>
1	<u>5</u>	<u>8</u>	<u>9</u>	<u>3</u>	<u>1</u>
2	<u>7</u>	<u>2</u>	<u>5</u>	<u>1</u>	<u>3</u>
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

"convolution"  
↓  
\*

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

3x3  
filter

=

<u>-5</u>	<u>-4</u>	0	<u>8</u>
<u>-10</u>	-2	2	3
0	-2	-4	-7
-3	-2	-3	<u>-16</u>

4x4

python: conv-forward  
tensorflow: tf.nn.conv2d  
keras: Conv2D



# 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



\*

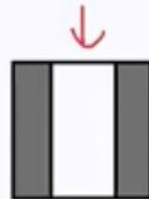
1	0	-1
1	0	-1
1	0	-1

3x3

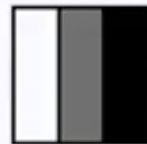
=

0	30	30	0
0	30	30	0
0	30	30	0
0	<u>30</u>	30	0

4x4



\*



↑ ↑ ↑

# More on 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



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

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



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

→

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

Sobel filter



convolution  
\*

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

3x3

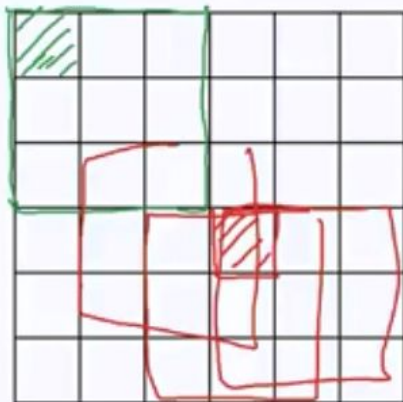
=  
45°  
70°  
73°

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

Scharr filter




# Padding:



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

\*



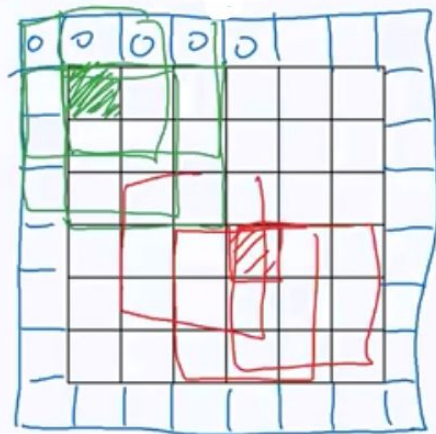
$$\begin{matrix} 3 \times 3 \\ f \times f \end{matrix}$$

=

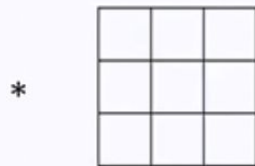
$$\begin{aligned} n-f+1 \times n-f+1 \\ 6-3+1=4 \end{aligned}$$

→ 4 × 4

# Padding:



- shrinky output
- throw away info from edge



$3 \times 3$   
 $f \times f$

=



$6 \times 6$

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

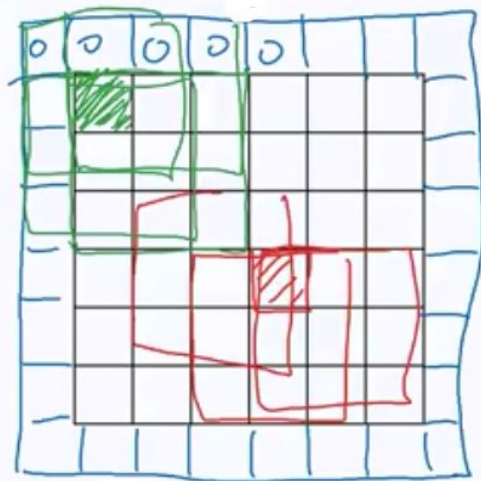
$p = \text{padding} = \underline{1}$

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

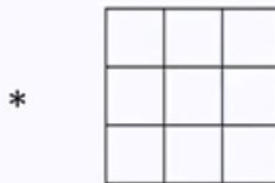
$n + 2p - f + 1 \times n + 2p - f + 1$   
 $6 + 2 - 3 + 1 \times \underline{\quad} = 6 \times 6$

~~$4 \times 4$~~

# Padding:



- shrinking output
- throw away info from edge



$3 \times 3$   
 $f \times f$

=



$6 \times 6$

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

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

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

$$6 - 3 + 1 = 4$$

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

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

~~$4 \times 4$~~

# Valid versus same padding:

→ no padding

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

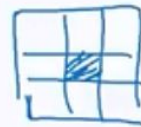
“Same”: Pad so that output size is the same as the input size.

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

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

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

$f$  is usually odd



1x1  
3x3  
5x5  
7x7



# Strided convolution:

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

7x7

\*

3	4	4
1	0	2
-1	0	3

3x3

stride = 2

=

91		

# Strided convolution:

The diagram illustrates a strided convolution operation. On the left is a 7x7 input grid with values: 

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

. A blue 3x3 region is highlighted, starting from the second row and second column, with handwritten superscripts above some cells. A blue arrow indicates a stride of 2 from the top-left of this region to the top-right. Below the grid is the handwritten label "7x7". In the center is a 3x3 kernel grid: 

3	4	4
1	0	2
-1	0	3

, with the handwritten label "3x3" below it. To the left of the kernel is a "\*" symbol, and to the right is an "=" symbol. On the far right is a 3x3 output grid: 

91	100	

. Below the kernel and output grids is the handwritten label "Stride = 2".

# Strided convolution:

Diagram illustrating a strided convolution operation. The input is a 7x7 matrix:

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

The 3x3 region highlighted in blue (rows 1-3, columns 4-6) is the kernel area. A blue arrow indicates a stride of 2 from the top-left of this region to the top-right.

The kernel is a 3x3 matrix:

3	4	4
1	0	2
-1	0	3

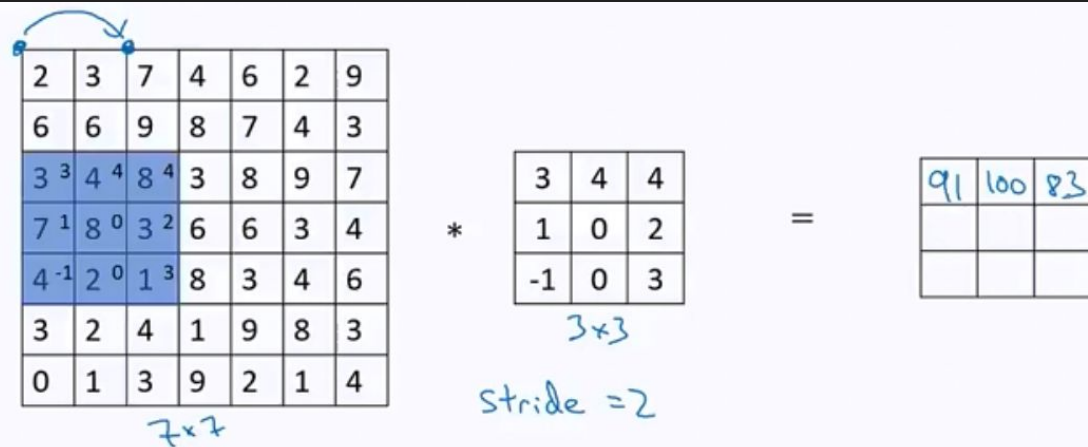
3x3

stride = 2

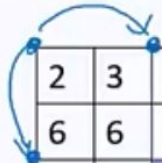
The result is a 3x3 matrix:

91	100	83

# Strided convolution:



# Strided convolution:



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

7x7

\*

3	4	4
1	0	2
-1	0	3

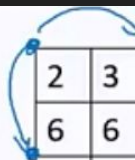
3x3

Stride = 2

=

91	100	83
69	91	

# Strided convolution:



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

7x7

\*

3	4	4
1	0	2
-1	0	3

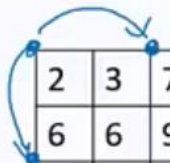
3x3

Stride = 2

=

91	100	83
69	91	12

# Strided convolution:



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

7x7

\*

3	4	4
1	0	2
-1	0	3

3x3

stride = 2

=

91	100	83
69	91	127

# Strided convolution:

The diagram illustrates a strided convolution operation. It shows a 7x7 input matrix, a 3x3 kernel, and a 3x3 output matrix. The current step involves a 3x3 sub-region of the input matrix, highlighted in blue, which is convolved with the kernel. The handwritten notes indicate the kernel size is 3x3, the stride is 2, and the current sub-region is 7x7.

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

7x7

3x3

Stride = 2

\*

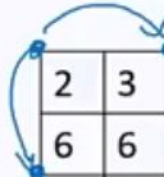
3	4	4
1	0	2
-1	0	3

=

91	100	83
69	91	127
44	7	



# Strided convolution:



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

7x7

\*

3	4	4
1	0	2
-1	0	3

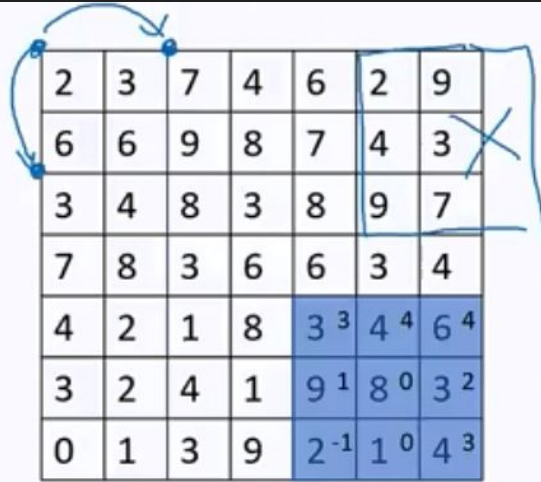
3x3

stride = 2

=

91	100	83
69	91	127
44	72	74

# Strided convolution:



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

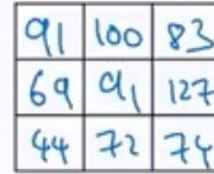
\*

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:

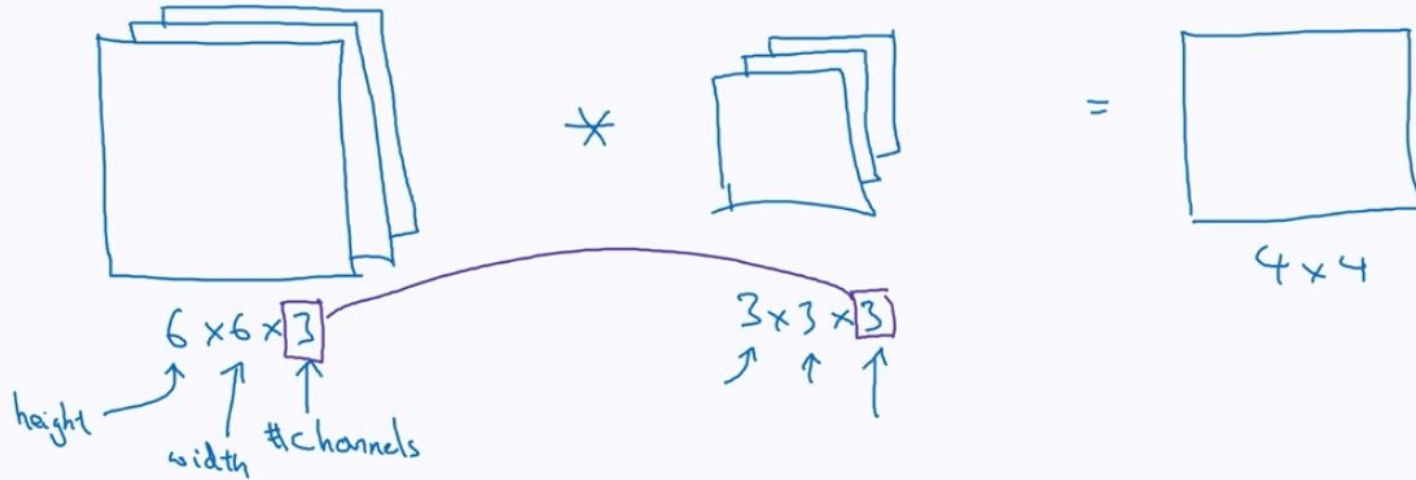
$n \times n$  image       $f \times f$  filter

padding  $p$       stride  $s$

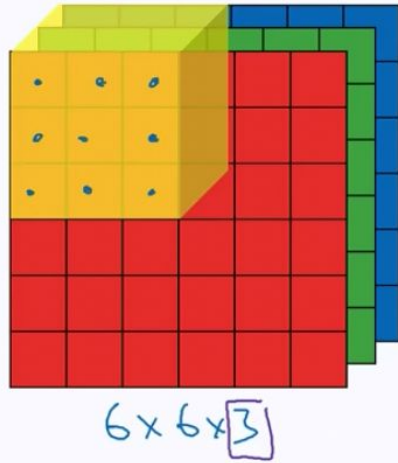
f.

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

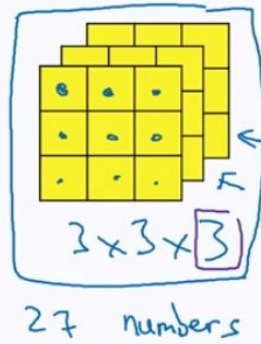
# Convolutions on RGB images



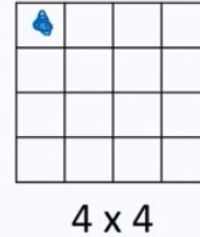
# Convolutions on RGB image



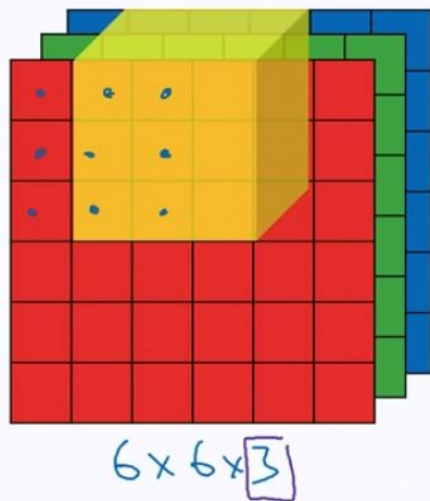
\*



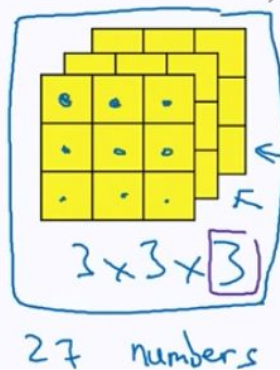
=



# Convolutions on RGB image



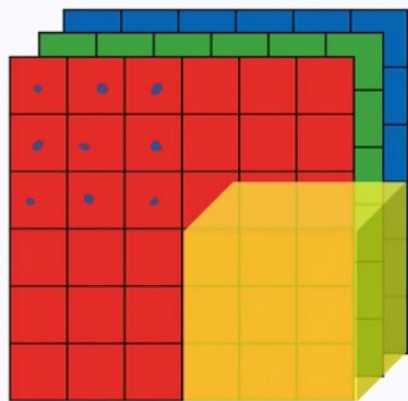
\*



=

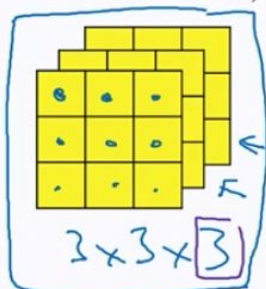


# Convolutions on RGB image

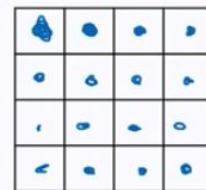


$6 \times 6 \times 3$

\*



=



$4 \times 4$

27 numbers

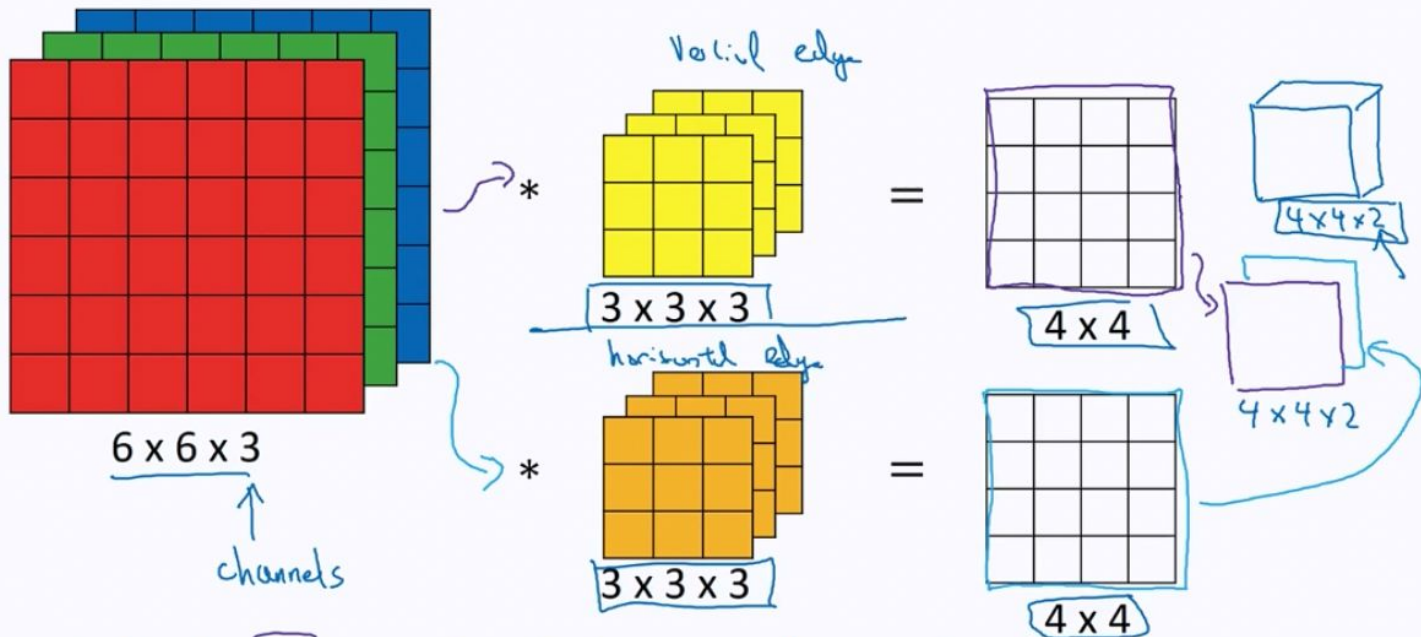


$\rightarrow 3 \times 3 \times 3$



$\rightarrow 3 + 3 + 3$

# Multiple filters

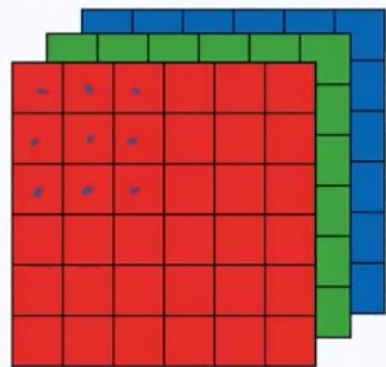


Summary:  $n \times n \times n_c$   $\times$   $f \times f \times n_c$   $\rightarrow$   $\frac{n-f+1}{4} \times \frac{n-f+1}{4} \times \frac{n_c}{2}$   $\uparrow$  #filters

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



# Example of a layer



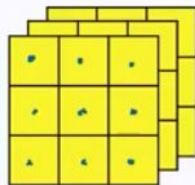
6 x 6 x 3

$a^{[1]}$

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

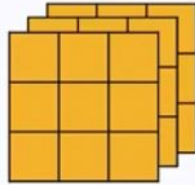
$$a^{[1]} = g(z^{[1]})$$

\*



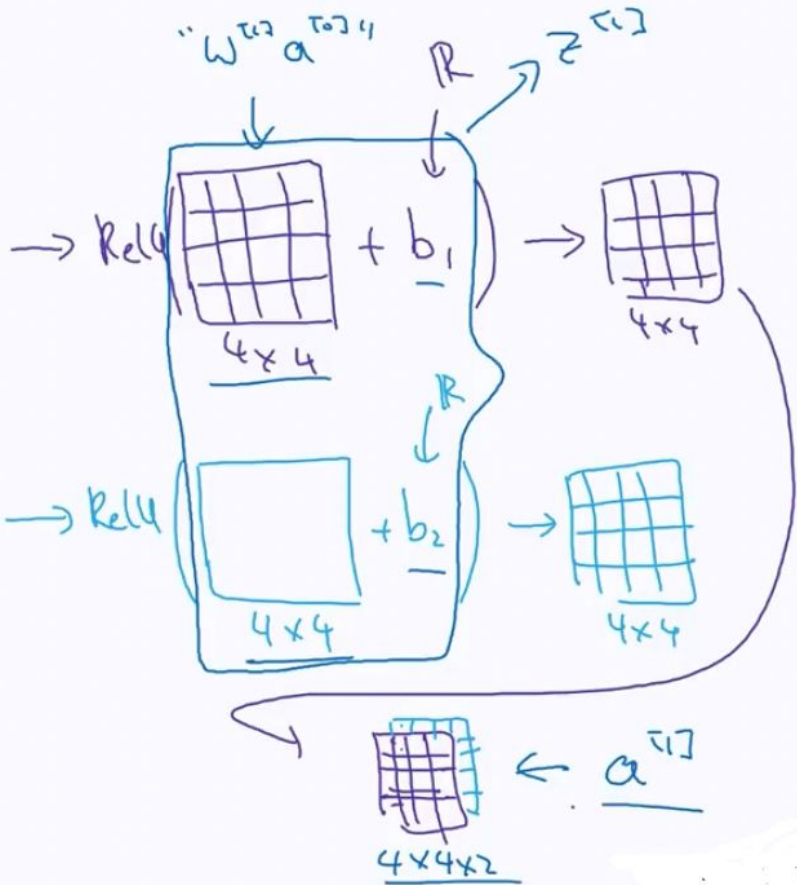
3 x 3 x 3

\*



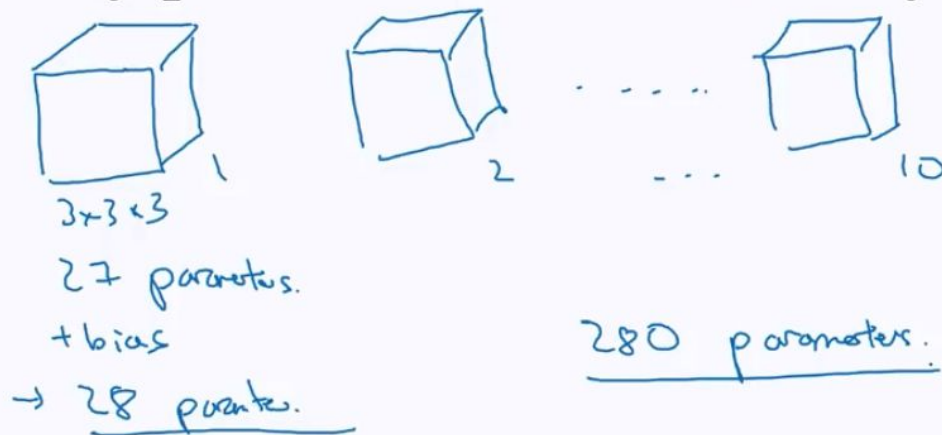
3 x 3 x 3

" $W^{[1]}$ "



# 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

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

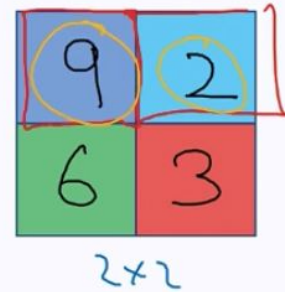
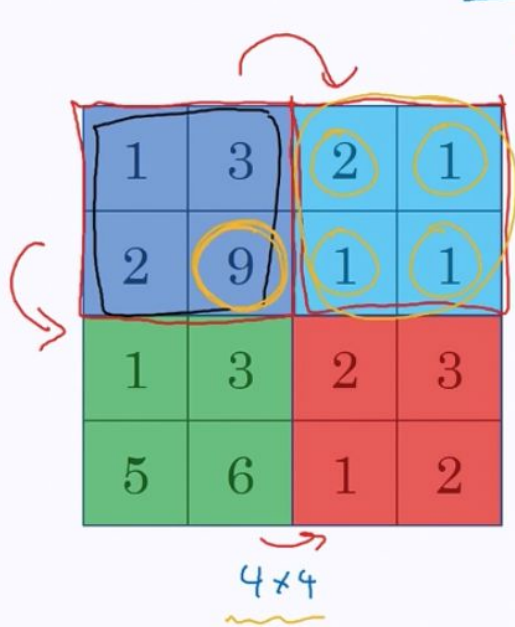
Output:  $n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$

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

## Types of layer in a convolutional network:

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

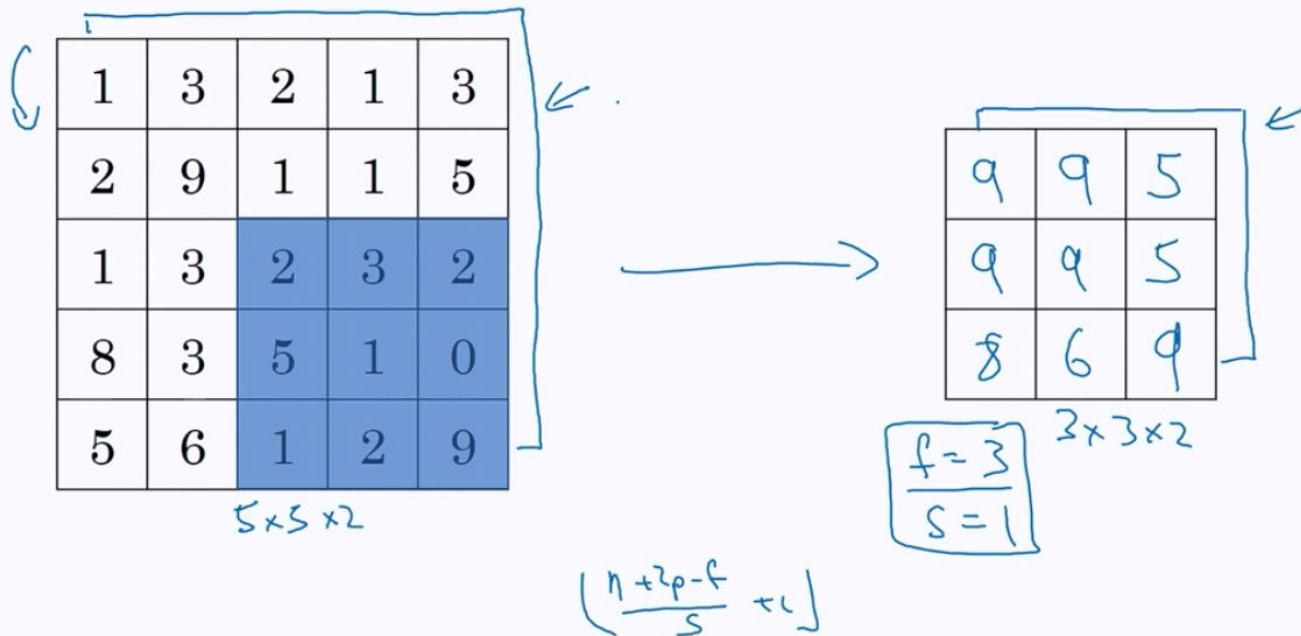
# Pooling layer: Max pooling



Hyporparameters:  
 $f = \underline{2}$   
 $s = \underline{2}$

No parameters.!

## 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

$$f=2$$
$$s=2$$

# Summary of pooling

Hyperparameters:

f : filter size

$f=2, s=2$

$f=3, s=2$

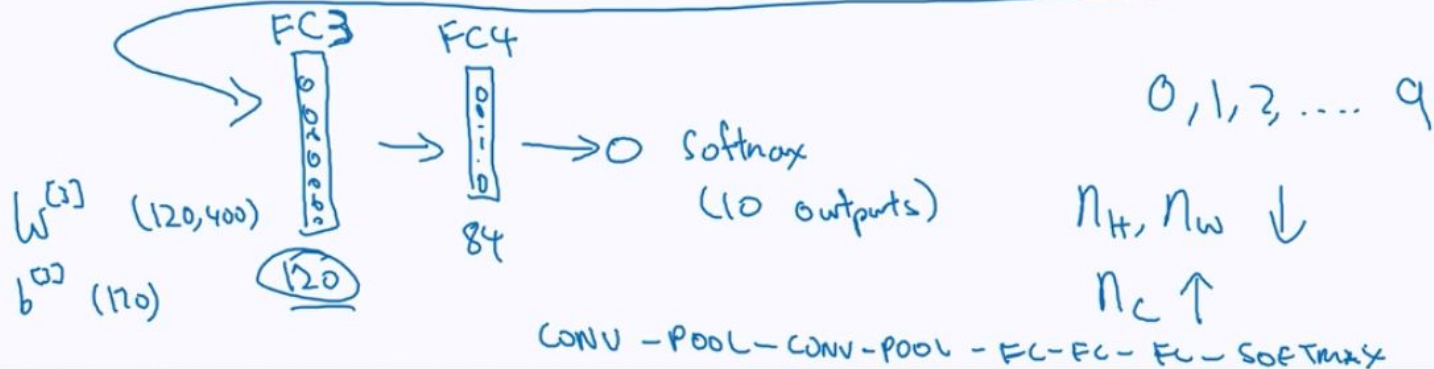
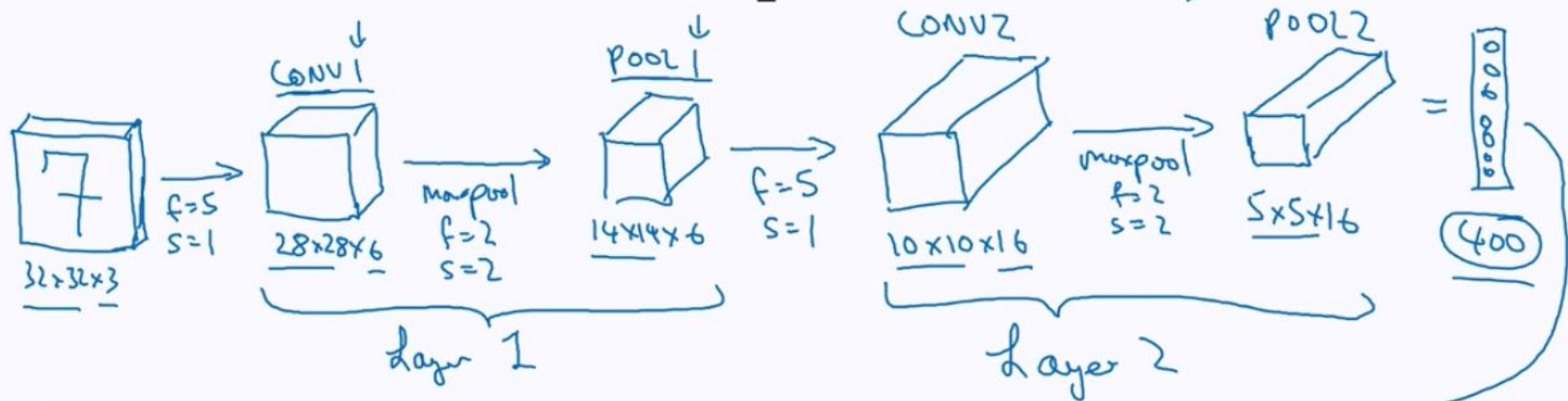
s : stride

Max or average pooling

→ p: padding.



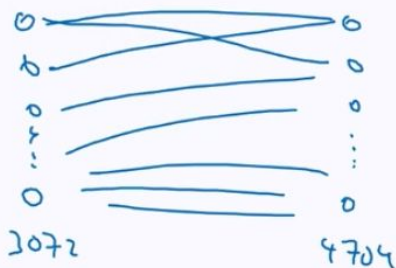
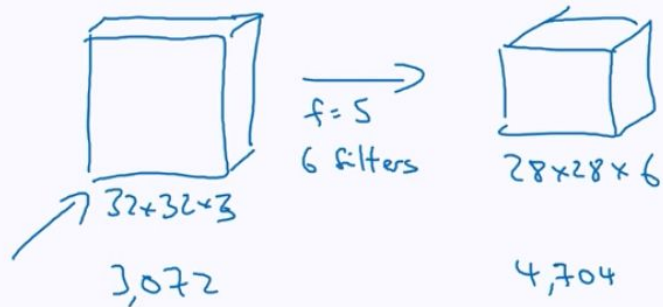
# Neural network example (LeNet-5)



# Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	— 3,072 $a^{col}$	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

# Why convolutions



$$5 \times 5 = 25$$

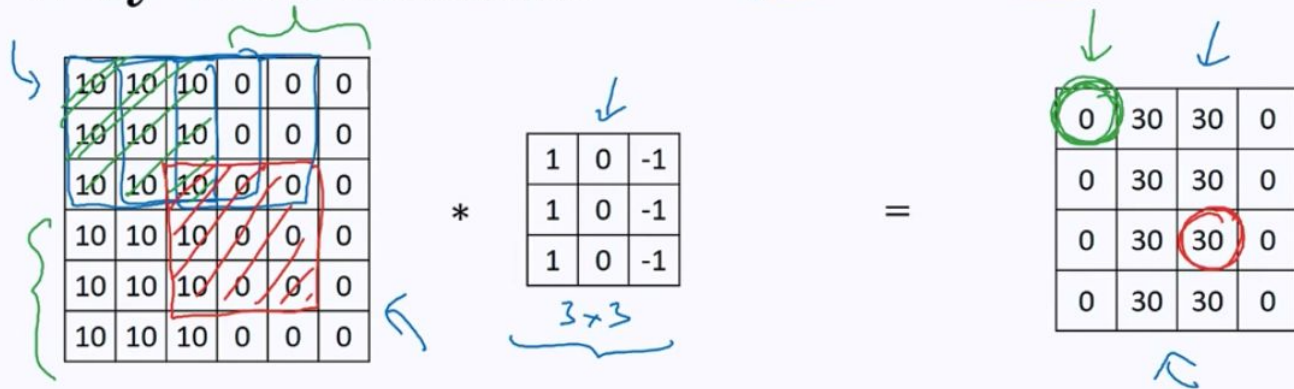
$$26$$

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

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

# Why convolutions

translation invariance

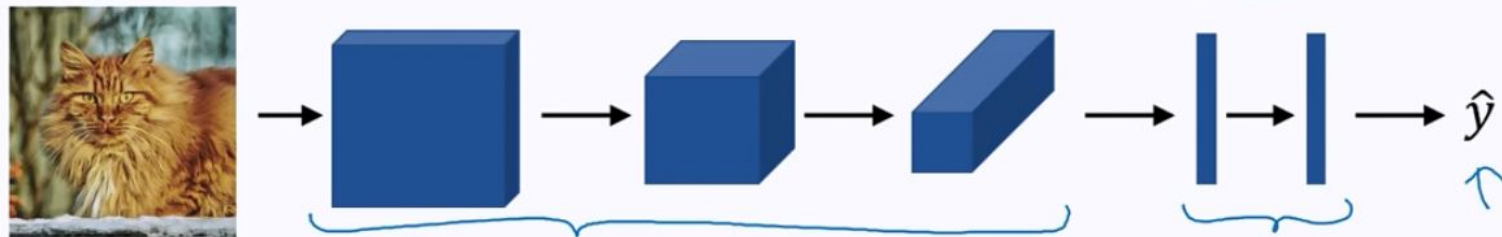


**Parameter sharing:** A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **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  $J$