deeplearning.ai

Convolutional Neural Networks

Computer vision

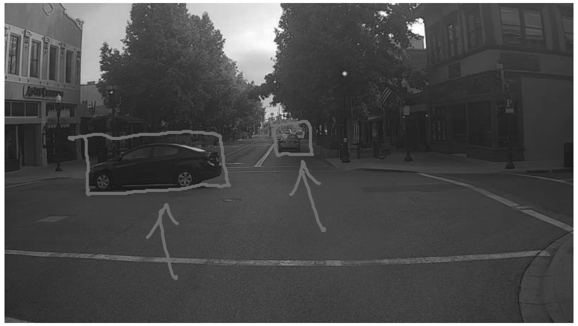
Computer Vision Problems

Image Classification

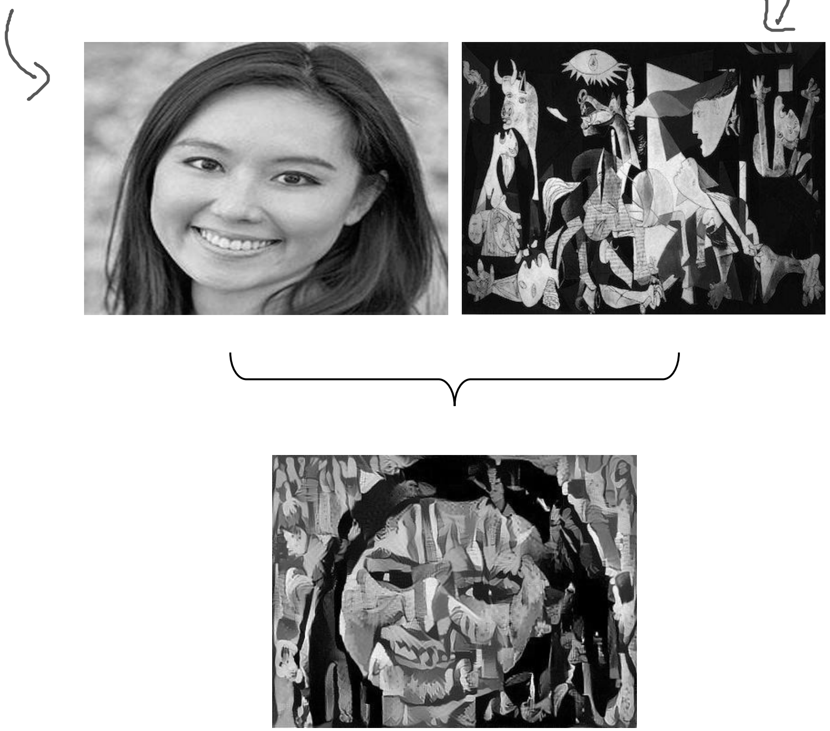
64x64

→ Cat? (0/1)

Object detection



Neural Style Transfer



Andrew Ng

Deep Learning on large images

→ Cat? (0/1)

64x64 × 3

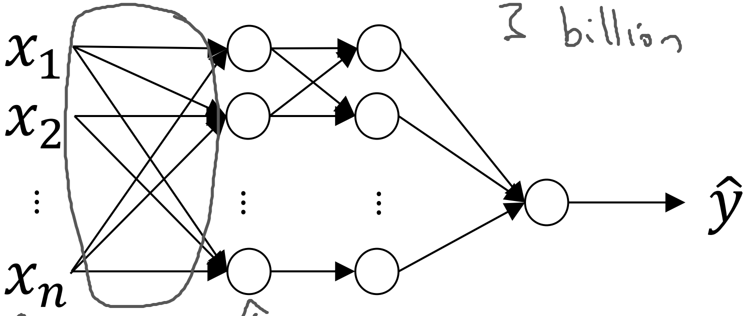
$\underline{12288}$

1000×1000×3
= 3 million

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

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

3 billion



↑3M

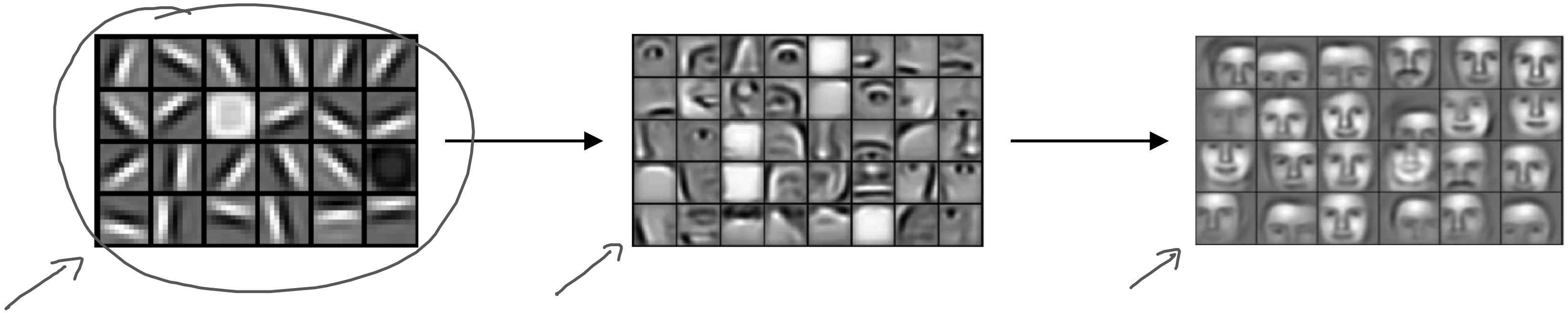
↑1000

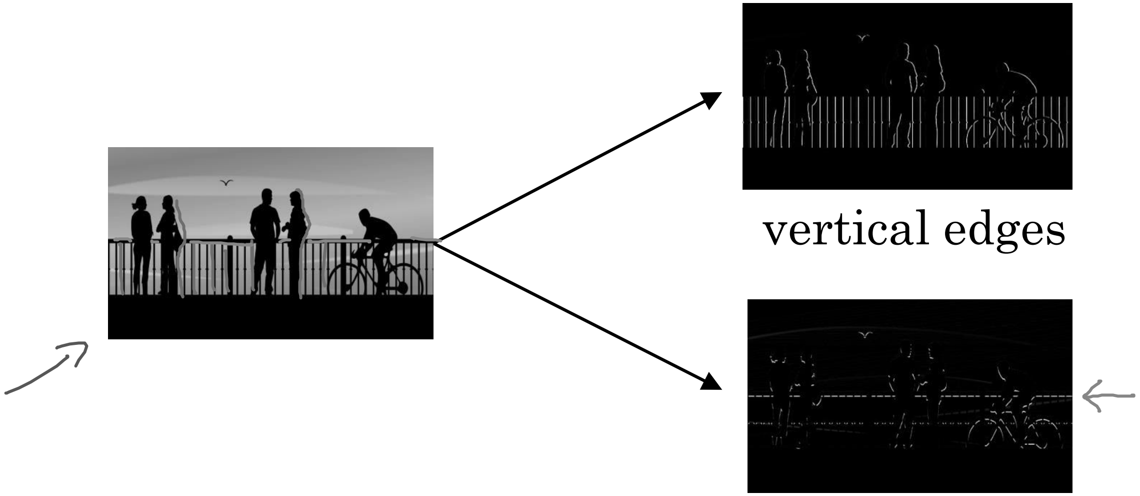
deeplearning.ai

Convolutional Neural Networks

Edge detection example

Computer Vision Problem





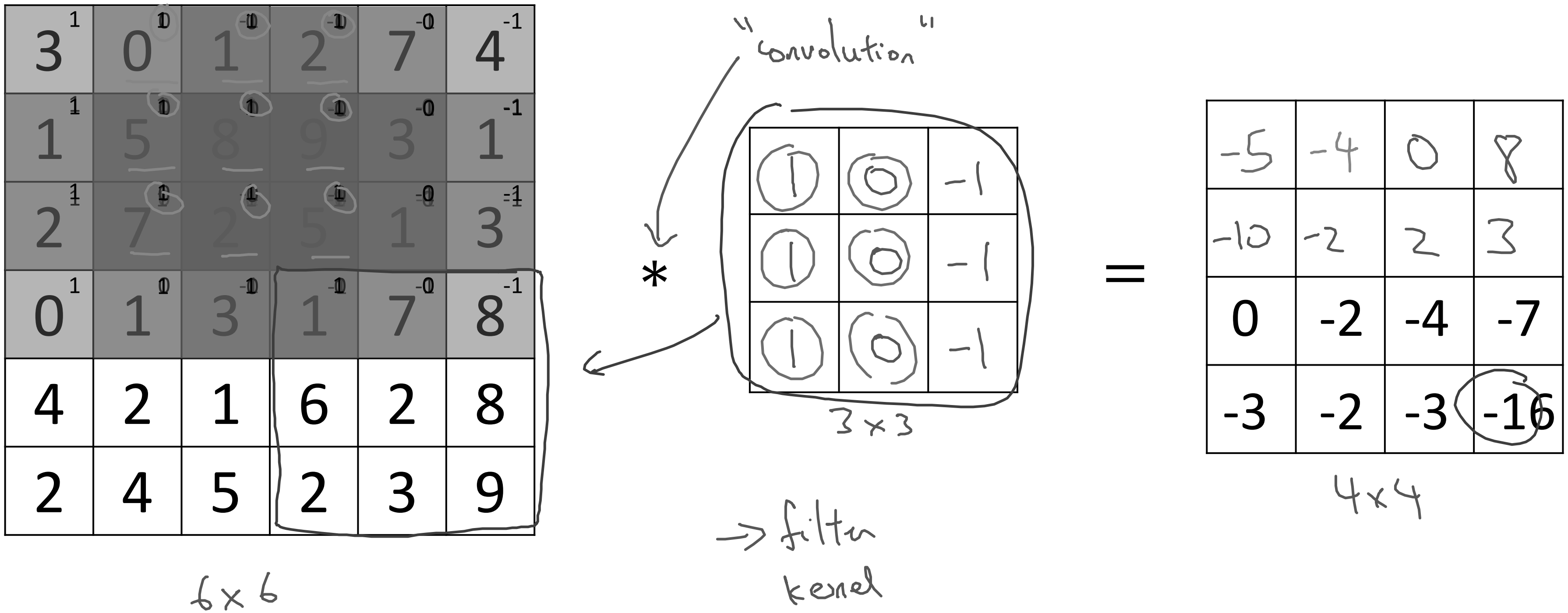
vertical edges

horizontal edges

Andrew Ng

Vertical edge detection

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



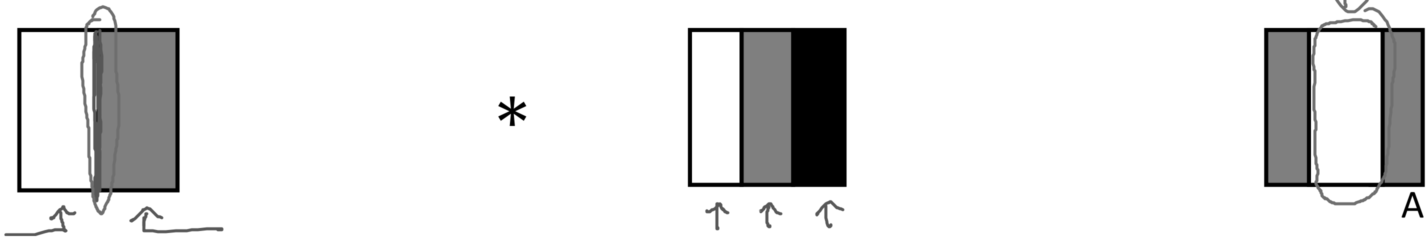
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

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

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

Vertical edge detection

$$\underset{6\times6}{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}} * \underset{3\times3}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{4\times4}{\begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}$$



Andrew Ng

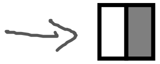
deeplearning.ai

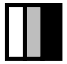
Convolutional Neural Networks

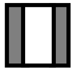
More edge detection

Vertical edge detection examples

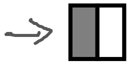
$$\begin{array}{|c|c|c|c|c|c|}\hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline \end{array}$$

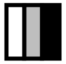


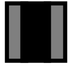




$$\begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & -30 & -30 & 0 \\ \hline 0 & -30 & -30 & 0 \\ \hline 0 & -30 & -30 & 0 \\ \hline 0 & -30 & -30 & 0 \\ \hline \end{array}$$







Andrew Ng

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} = \text{general filter}$$

Vertical and Horizontal Edge Detection

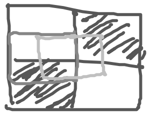
$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad\qquad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Vertical

filters

Horizontal

$$\underset{6\times6}{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix}} * \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 30 & 10 & -10 & -30 \\ 30 & 10 & -10 & -30 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



Andrew Ng

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

3×3

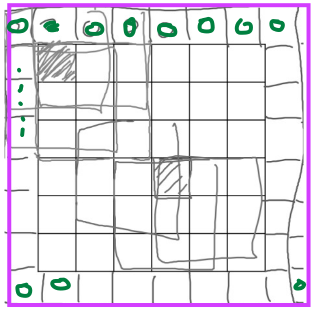
=

45°
70°
73°

deeplearning.ai

Convolutional Neural Networks

Padding



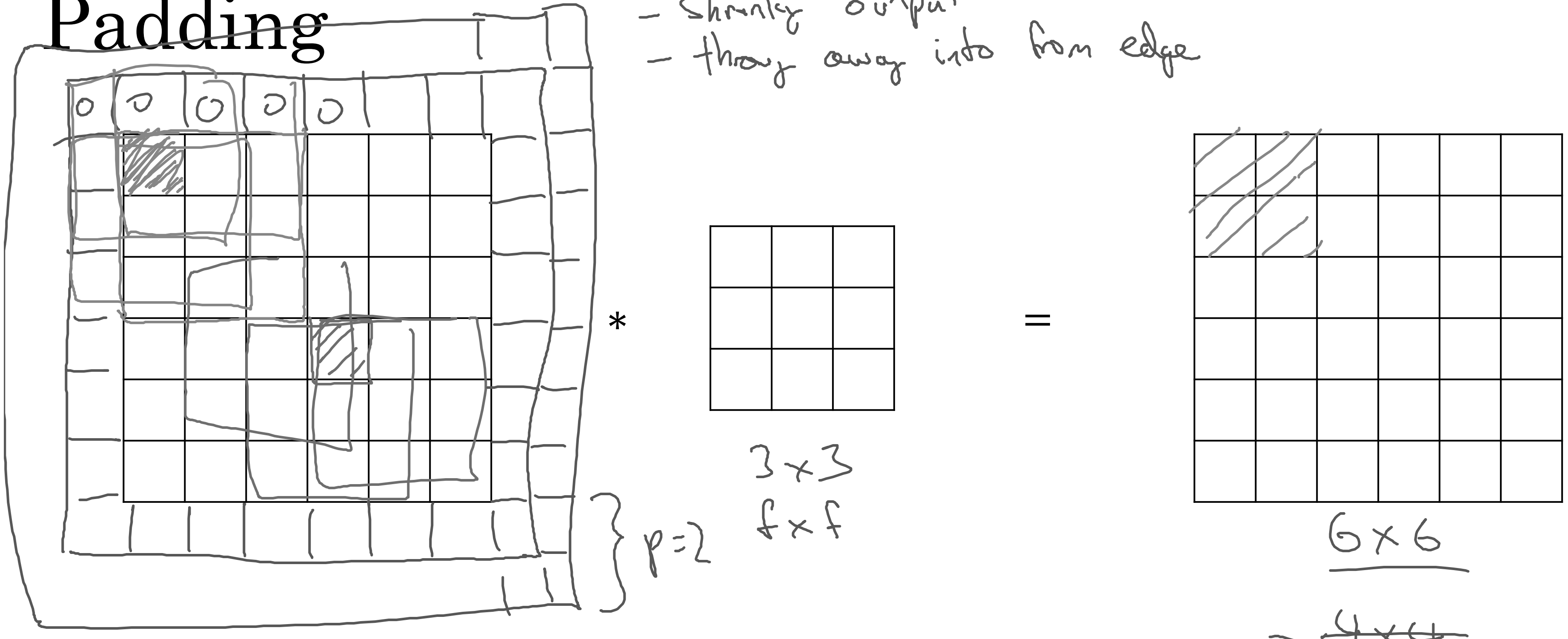
$\Rightarrow P=1$

$(n\times n) * (f\times f) \xRightarrow{P} (n+2P-f-1) * (n+2P-f-1)$

اثر پیکسل‌های حاشیه‌ای را متعادل(↑) می‌کنند.

Padding

- shrinky output
- throw away info from edge



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

$n \times n$

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

$6-3+1=4$

p = padding = $\underline{1}$

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

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

Andrew Ng

Valid and Same convolutions

→ no padding

"Valid": $n \times n \quad * \quad f \times f \quad \longrightarrow \quad \underline{n-f+1} \times n-f+1$

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

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

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

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

$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad \Big| \quad \underset{f=5}{5 \times 5} \quad p=2$

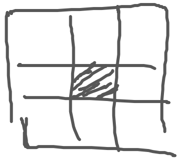
f is usually odd

1×1

3×3

5×5

7×7



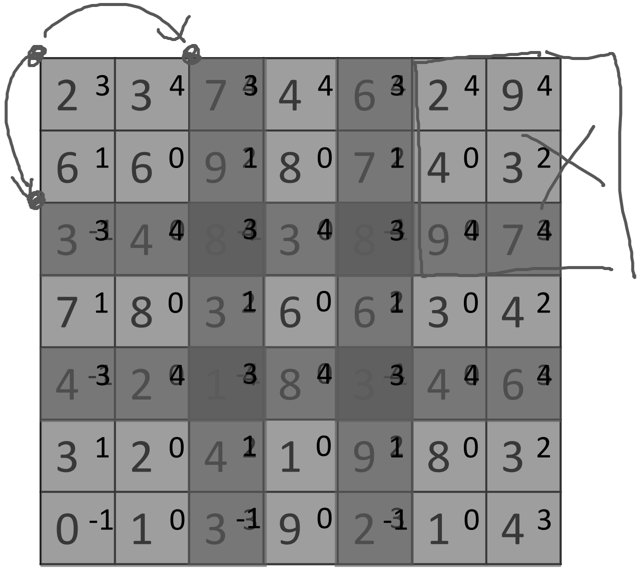
deeplearning.ai

$$(n \times n) * (f \times f) \xrightarrow{f,s} \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Convolutional Neural Networks

Strided convolutions

Strided convolution



7×7

$*$

3	4	4
1	0	2
-1	0	3

3×3

stride $= 2$

$=$

91	100	83
69	91	127
44	72	74

3×3

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

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

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 $\quad f \times f$ filter

padding $p \quad$ stride s

Output size:

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

Technical note on cross-correlation vs. convolution

⇝ Math

⇝ DL

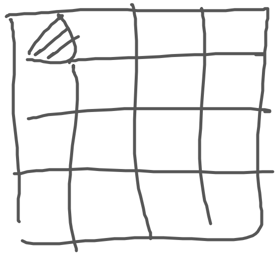
Convolution in math textbook:

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

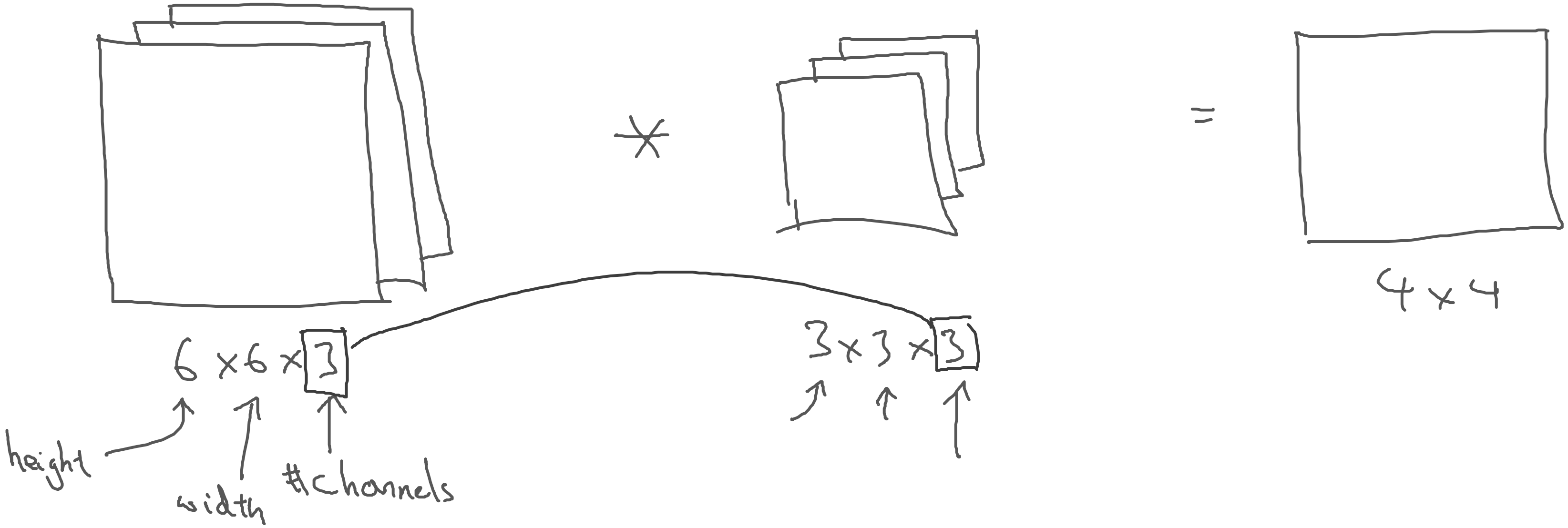
$$(A*B)*C = A*(B*C)$$

deeplearning.ai

Convolutional Neural Networks

Convolutions over volumes

Convolutions on RGB images

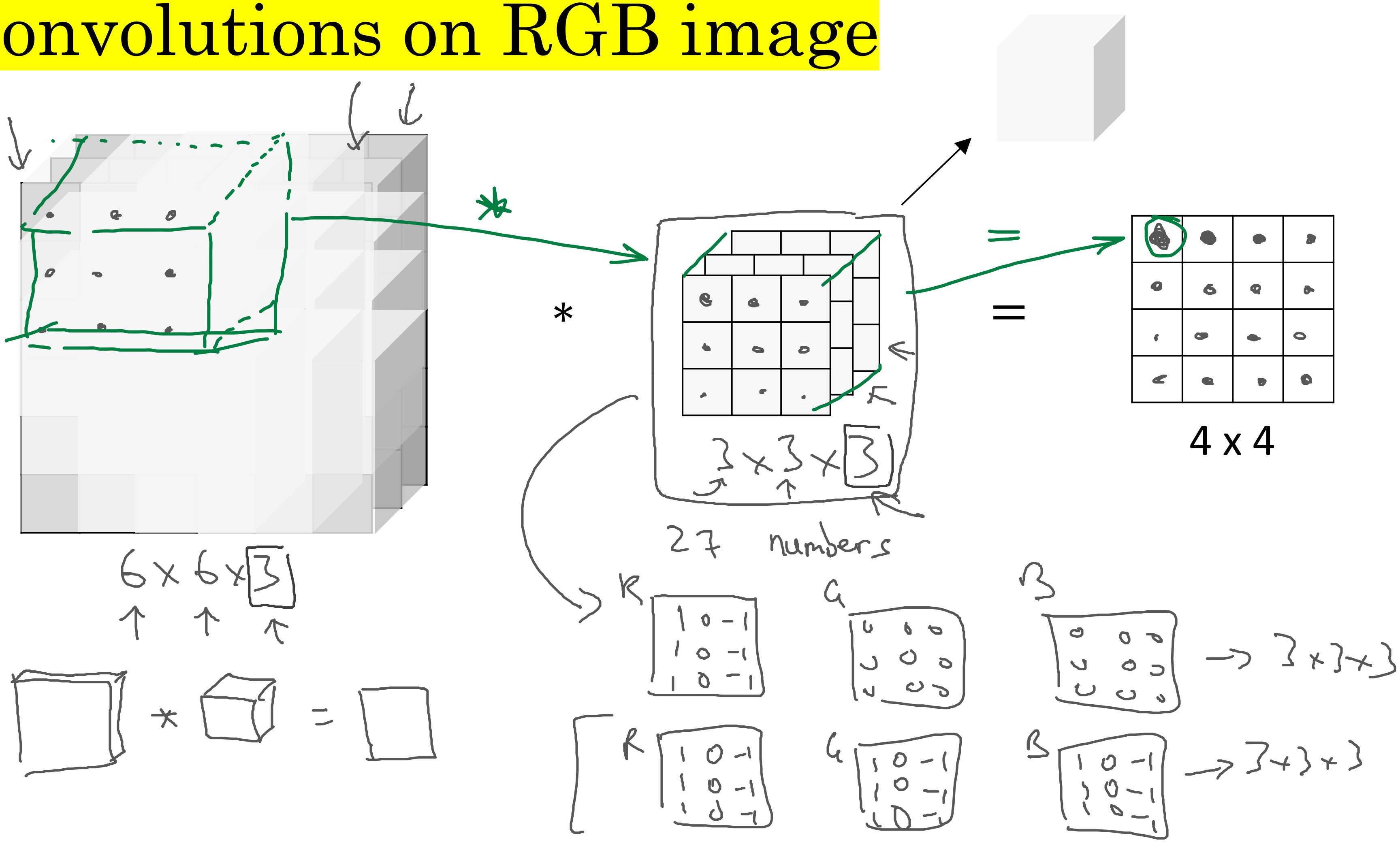


Andrew Ng

یک عکس RGB ابعاد 6×6×3 دارد که # color channel = 3 است و فیلترهایی که با ماتریس‌های سه رنگ زیر نمایش داده می‌شوند، از طریق

اعمال محاسبات حجمی انجام می‌شوند ⇐ Conv over volume هم به آن می‌گویند.

Convolutions on RGB image



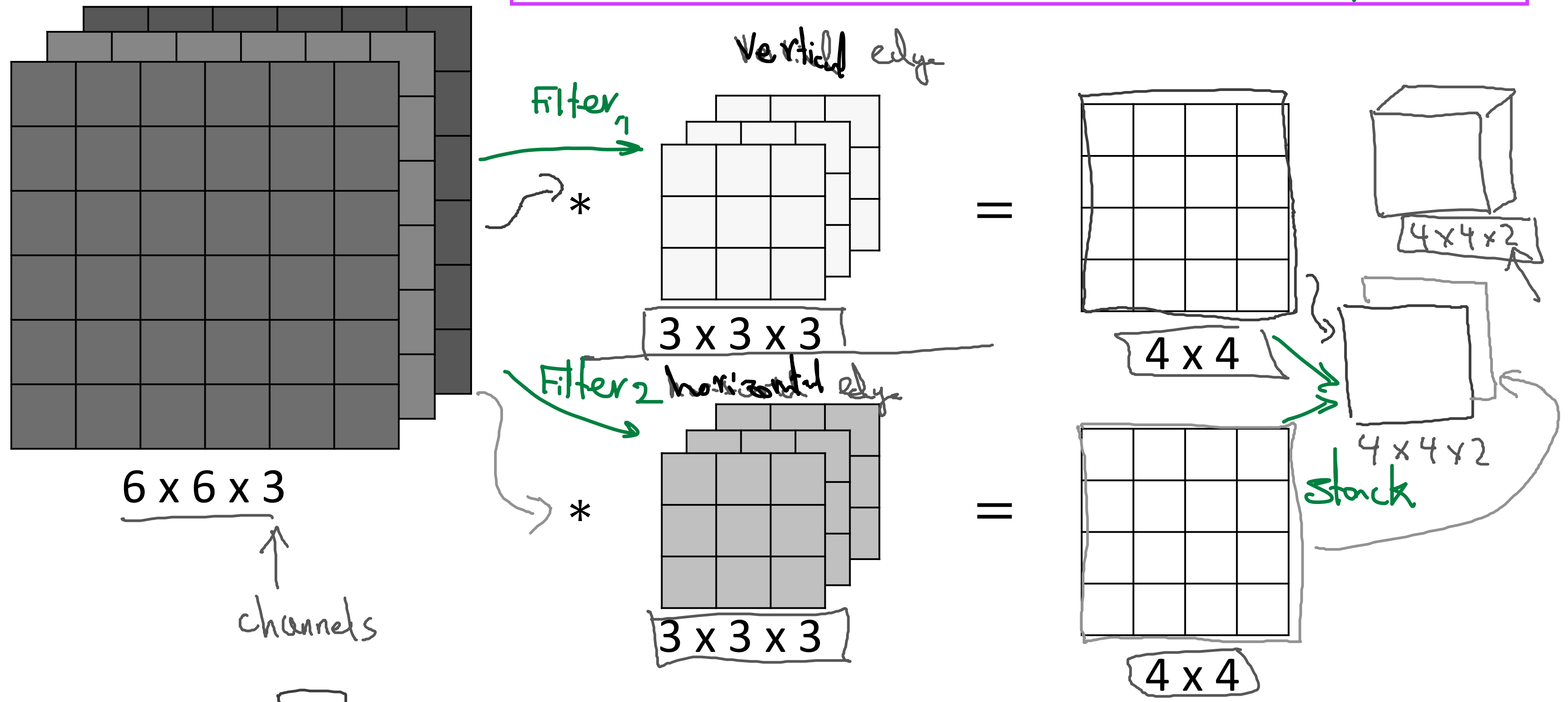
می‌توان همزمان چندین فیلتر را اعمال کرد و سپس نتایج را stack up کرد.

Multiple filters

$$(n \times n \times \underline{n_c}) * (f \times f \times \underline{n_c}) \rightarrow (n-f+1 \times n-f+1 \times n_c')$$

باید برابر باشد

n_c' → # filters



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

$6 \times 6 \times 3 \qquad 3 \times 3 \times 3 \qquad 4 \times 4 \times 2$

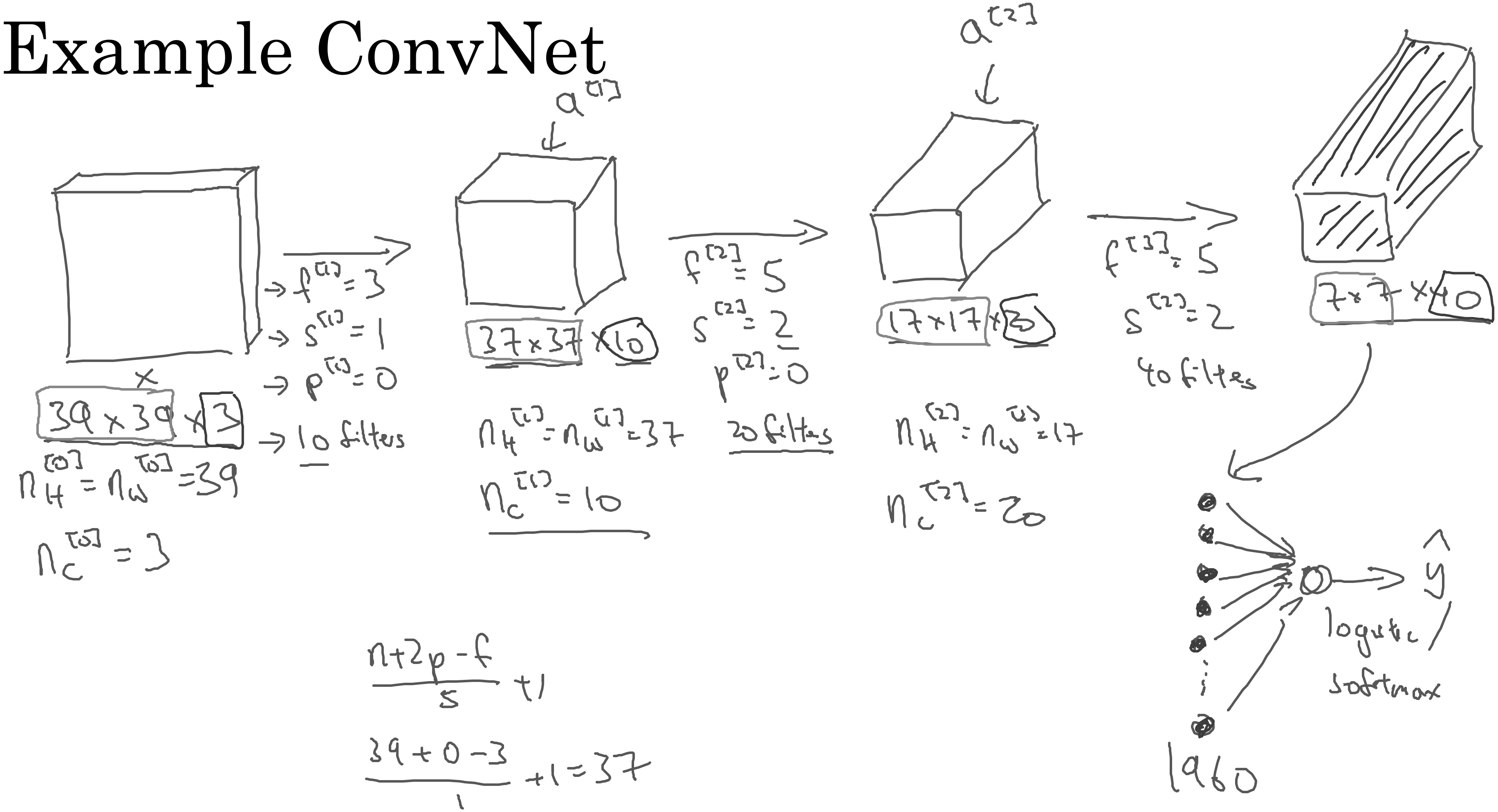
deeplearning.ai

Convolutional Neural Networks

A simple convolution network example

If 10 filters each 3x3x3 ⇒ # parameters $= 10(\underbrace{3\times 3\times 3}_{w}+\underset{b}{\overset{\uparrow}{1}}) = 280$

Example ConvNet



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

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

Andrew Ng

Types of layer in a convolutional network:

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

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

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

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

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

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

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

bias: $1 \times 1 \times 1 \times n_c^{[l]}$

activations $a^{[l]} = n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Vectorized $A^{[l]} = \underset{\text{↓ \# samples}}{m} \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

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

Notation Summary

deeplearning.ai

Convolutional Neural Networks

Pooling layers

max مرسوم‌تره

average

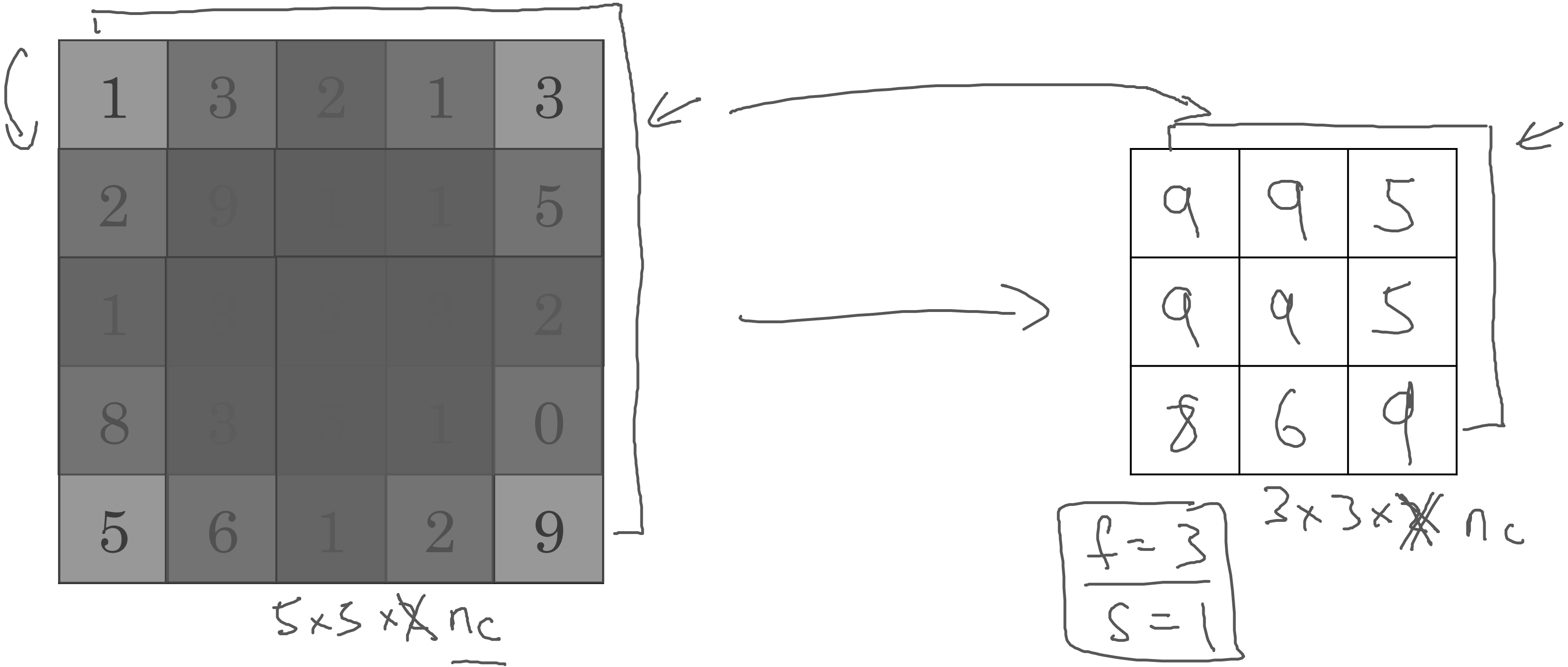
#Parameters = 0

#Hyperparameter= f, s (usually p=0)

باعث کاهش $n_W^{[l]}$ و $n_H^{[l]}$ می‌شود و افزایش عمق $n_C^{[l]}$ می‌شود. ⇐ کاربرد Pool و Fc: سایز (represent ↓ و سرعت ↑ می‌دهند.

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

Pooling layer: Max pooling



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

$5 \times 5 \times \cancel{1}\ \underline{n_c}$

9	9	5
9	9	5
8	6	9

$3 \times 3 \times \cancel{1}\ n_c$

$$\boxed{\frac{f=3}{s=1}}$$

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

Andrew Ng

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

$f=2$

$s=2$

$\underline{7\times7}\times 1000 \rightarrow 1\times1\times1000$

Andrew Ng

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!

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

$$\downarrow$$

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

$$\times \underline{n_C}$$

deeplearning.ai

Convolutional
Neural Networks

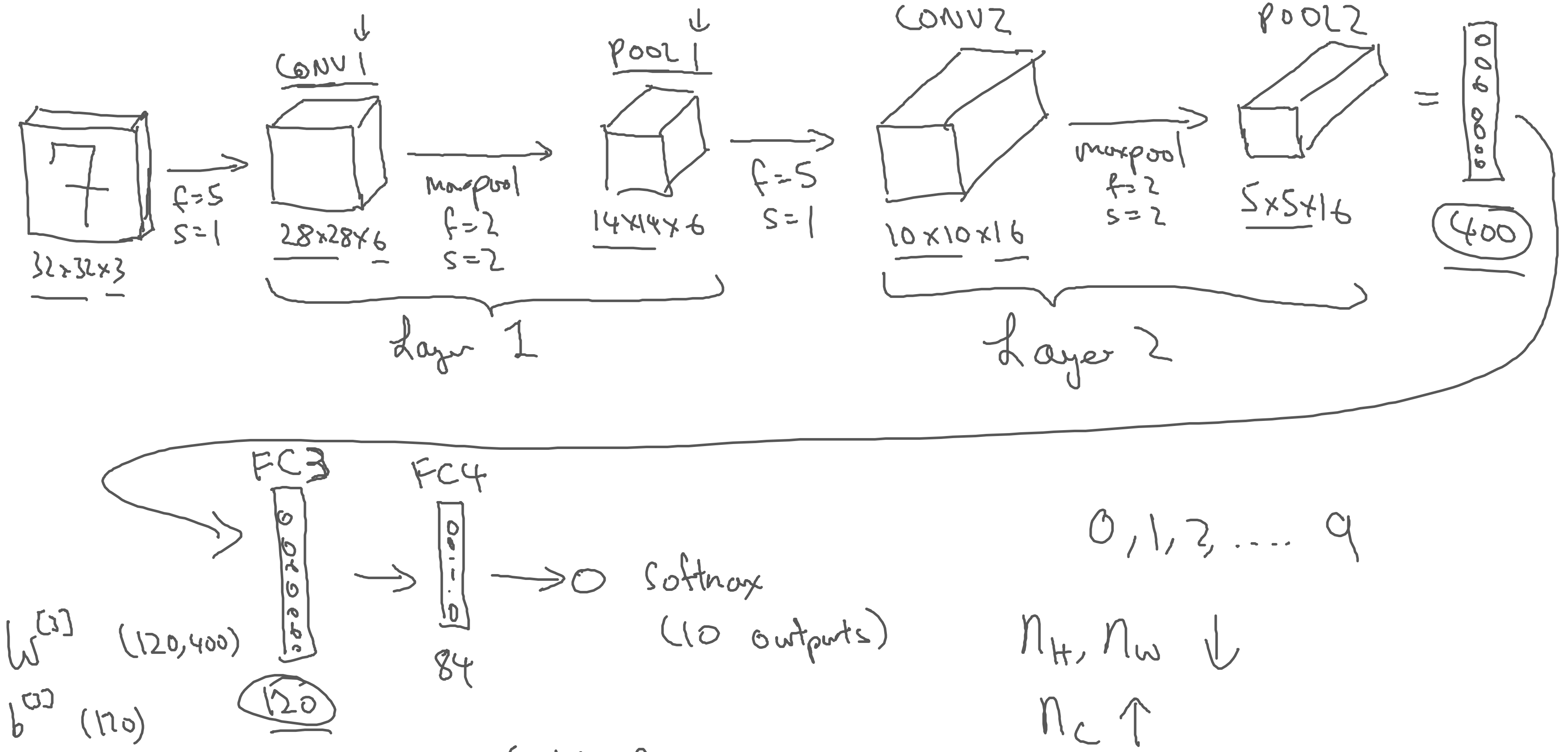
Convolutional neural network example

LeNet-5 = $\overbrace{\text{conv\_pool}}^{\text{layer}_1} - \overbrace{\text{cov\_pool}}^{\text{layer}_2} - \overbrace{\text{FC}}^{l_3} - \overbrace{\text{FC}}^{l_4} - \overbrace{\text{FC}}^{l_5} - \text{softmax}$

flattening

Neural network example

(LeNet-5)



CONV - POOL - CONV - POOL - FC - FC - FC - SOFTMAX

Neural network example

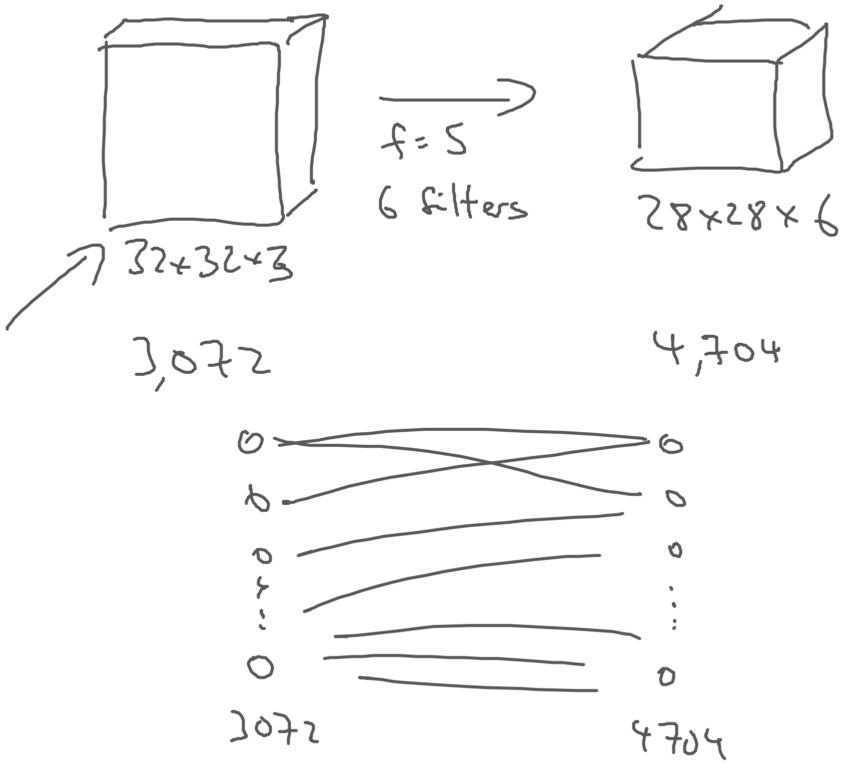
	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	– 3,072 $a^{[0]}$	0
Conv1 (f= 5, s=1)	(28, 28, 8)	6272	$8\times(5\times5\times3+1) = 608$ ←
Pool1	(14, 14, 8)	1568	0 ←
Conv2 (f=5, s=1)	(10, 10, 16)	1600	$16\times(5\times5\times8+1) = 3216$ ←
pool2	(5, 5, 16)	400	0 ←
FC3	(120, 1)	120	$400\times120 + 120 =$ 48120
FC4	(84, 1)	84	$120\times84+84 =$ 10164
Softmax	(10, 1)	10	$84\times10 + 10 = 850$

deeplearning.ai

Convolutional Neural Networks

Why convolutions?

Why convolutions



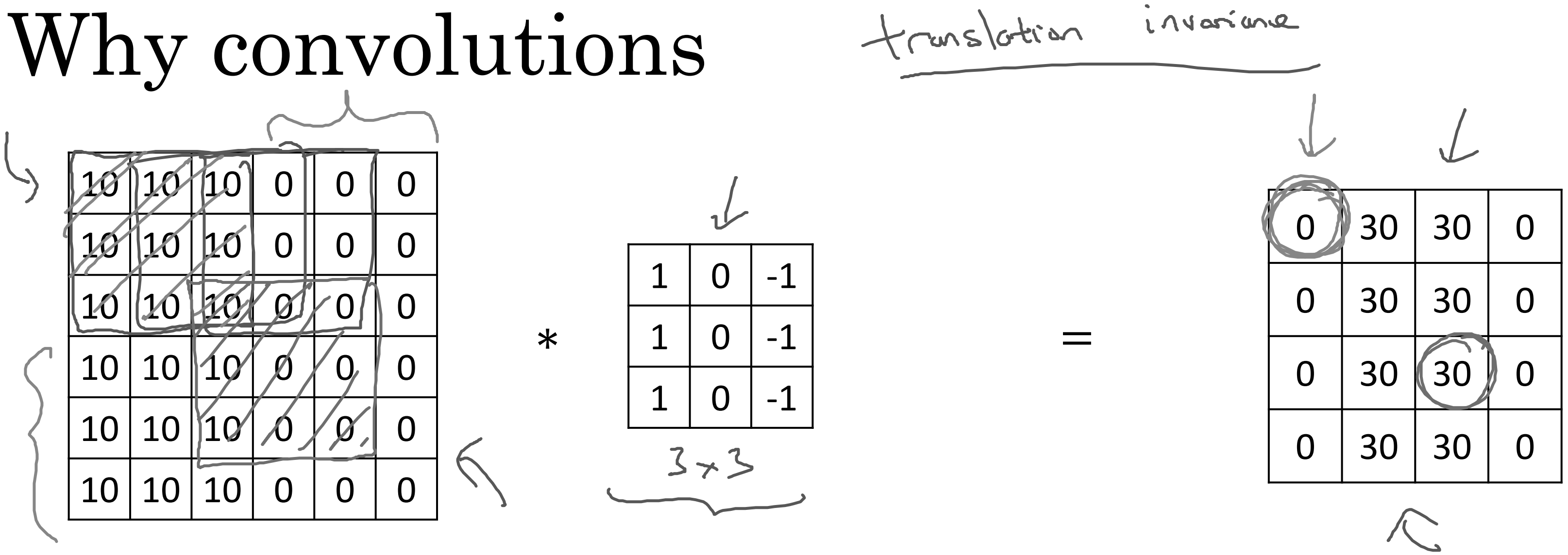
$5 \times 5 - 25$

26

$6 \times 26 = 156$ parameters

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

Why convolutions



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

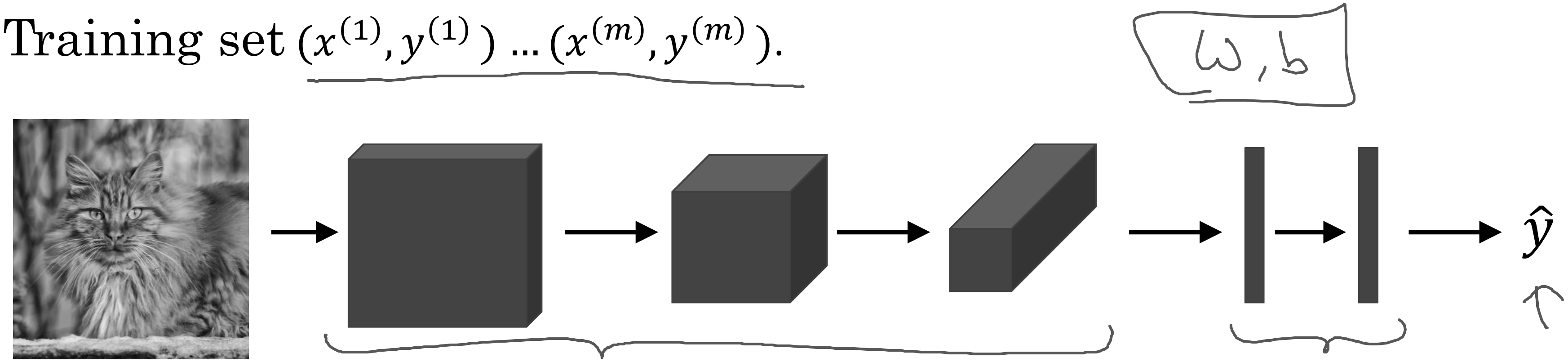
دو برتری Convnet
برای کار با عکس

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