كلية الهندسة
قسم هندسة الروبوت و الأنظمة الذكية

ذكاء صنعي 2

محاضرة 7

Convolutional Neural Networks

د. فادي متوج

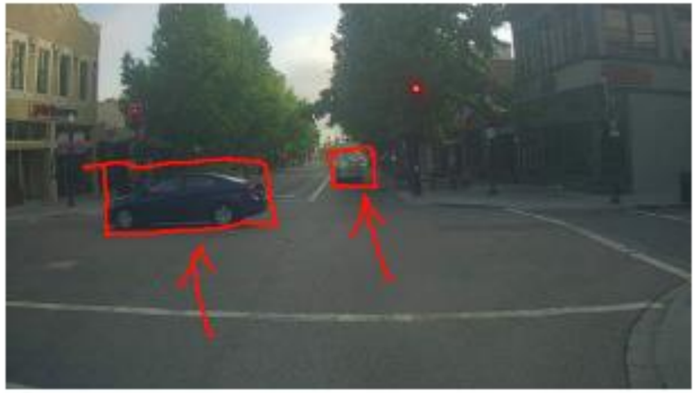
Computer Vision Problems

Image Classification

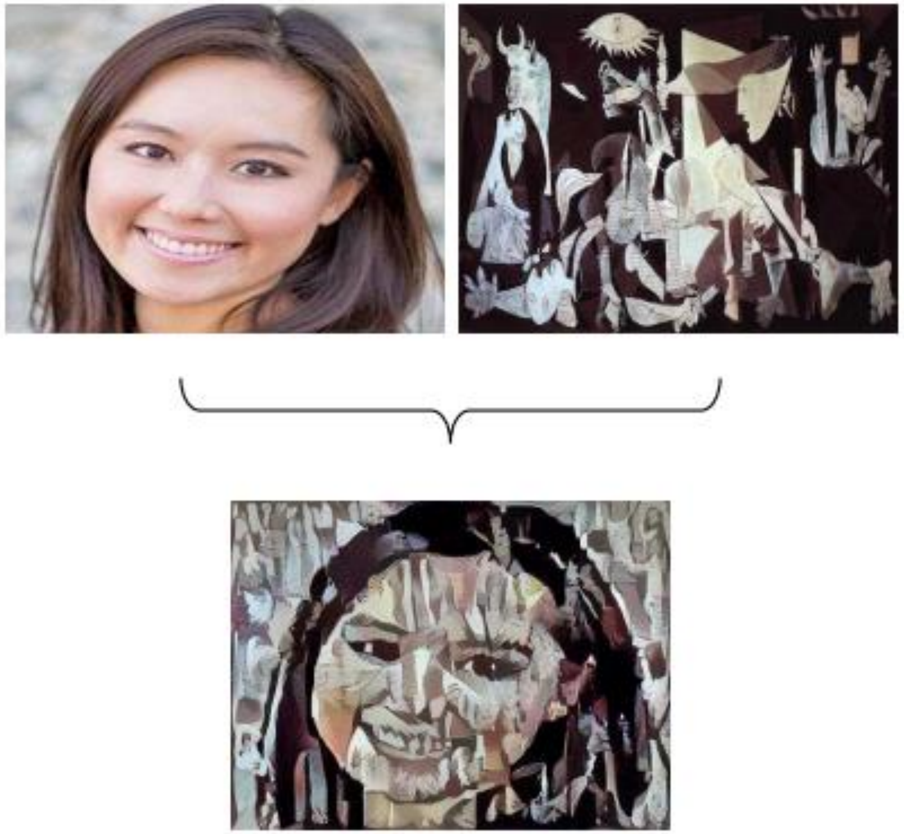
64x64

→ Cat? (0/1)

Object detection



Neural Style Transfer



Deep Learning on large images

64x64 × 3

→ Cat? (0/1)

12288

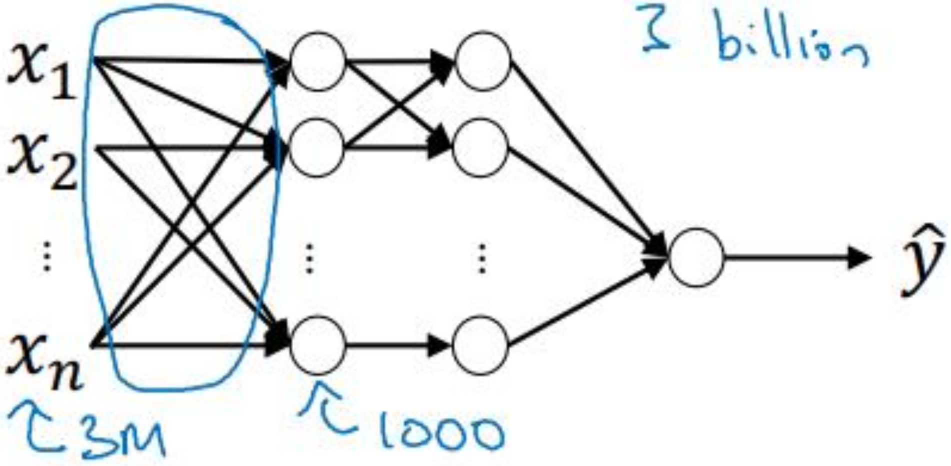
1000×1000×3

= 3 million

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

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

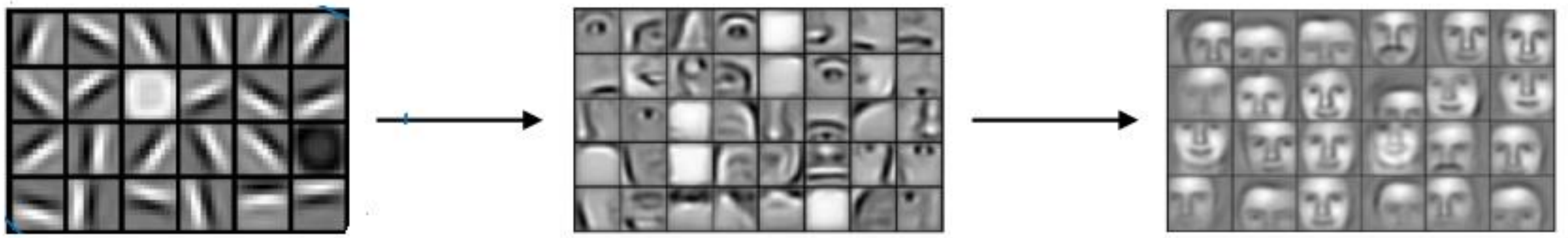
3 billion

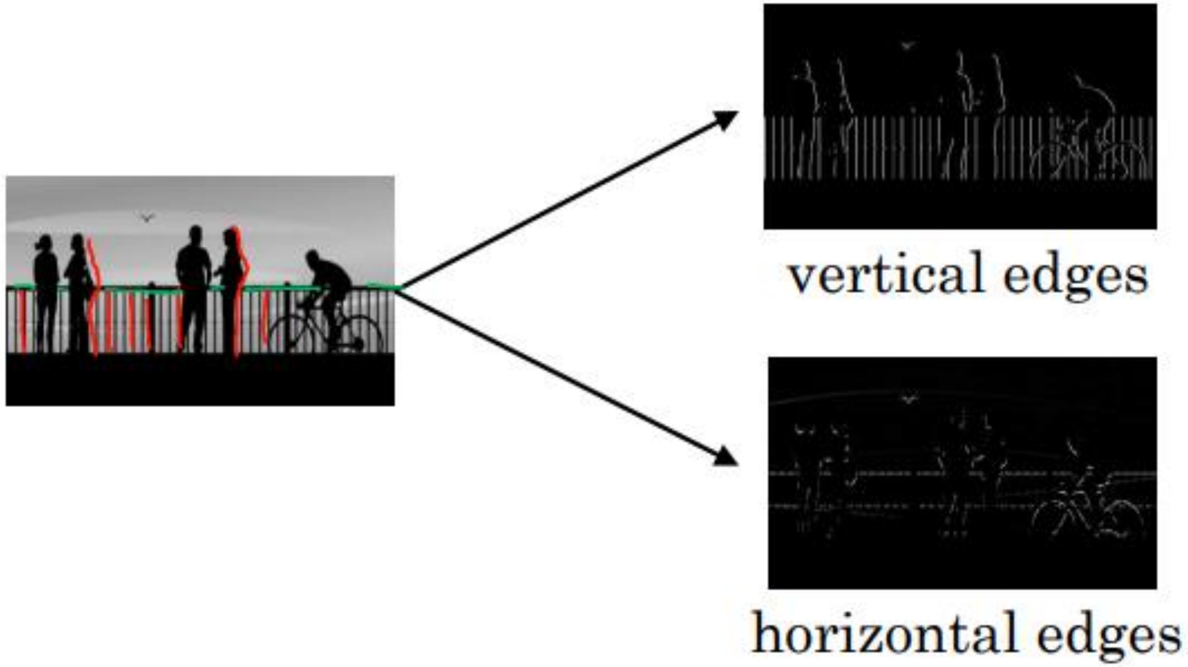


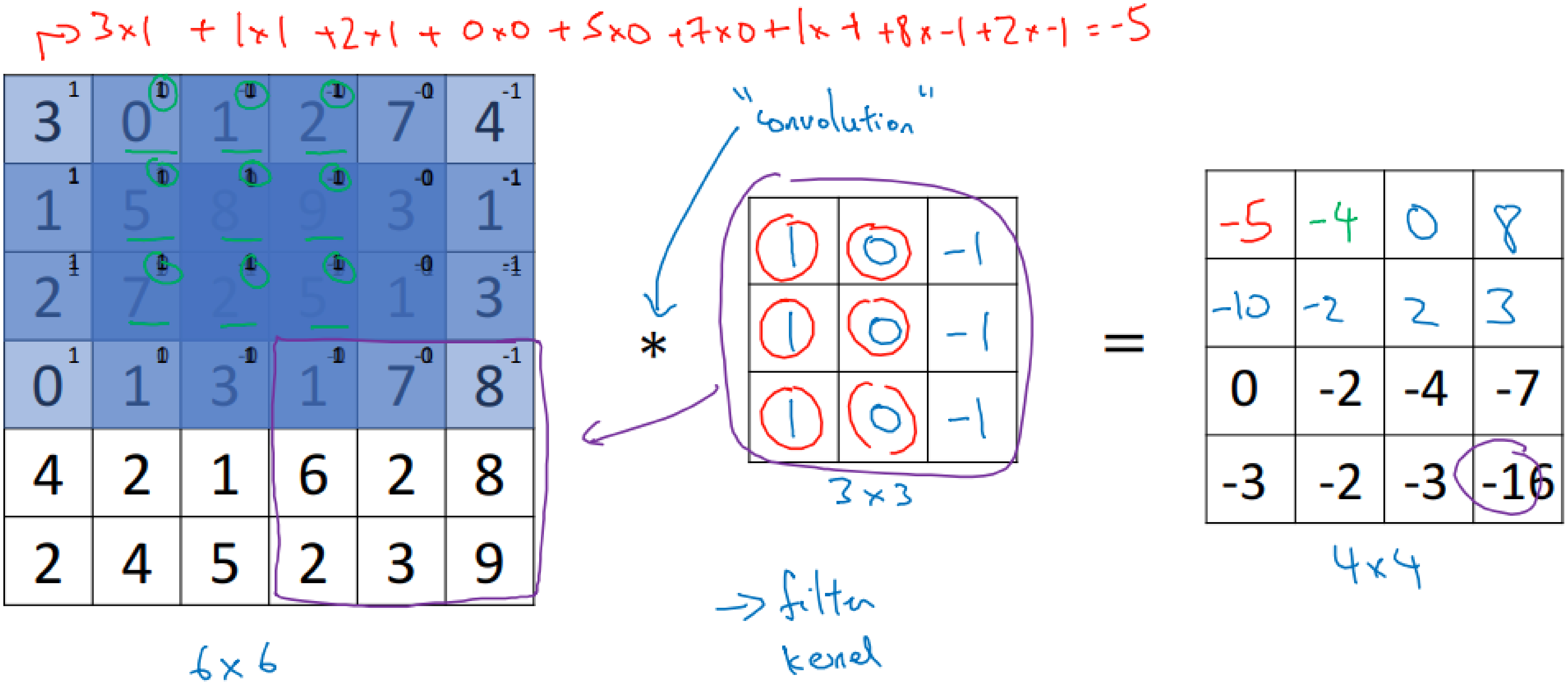
Convolutional Neural Networks

Edge detection example

Edge detection example







3×1 + 1×1 + 2×1 + 0×0 + 5×0 + 7×0 + 1×-1 + 8×-1 + 2×-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
"convolution"
*
1 0 -1
1 0 -1
1 0 -1
3x3
→ filter
kernel
=
-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	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6x6

*

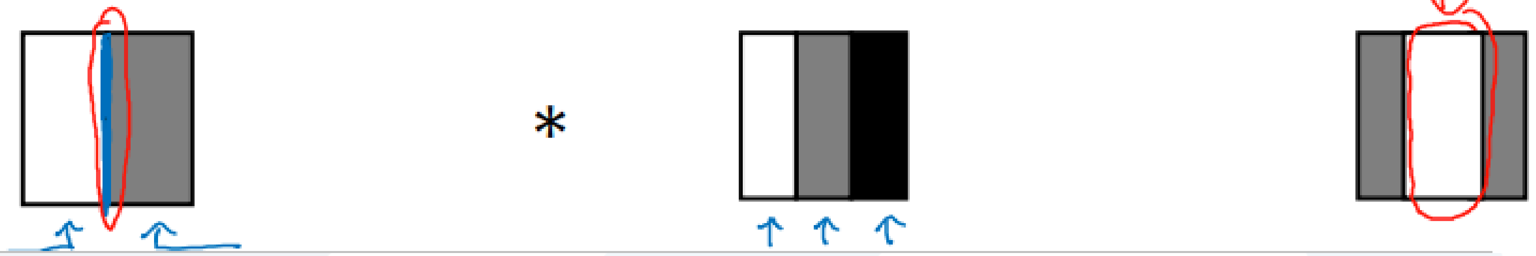
1	0	-1
1	0	-1
1	0	-1

3x3

=

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

4x4



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

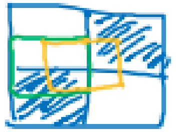
6x6

*

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

جامعة المنارة

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

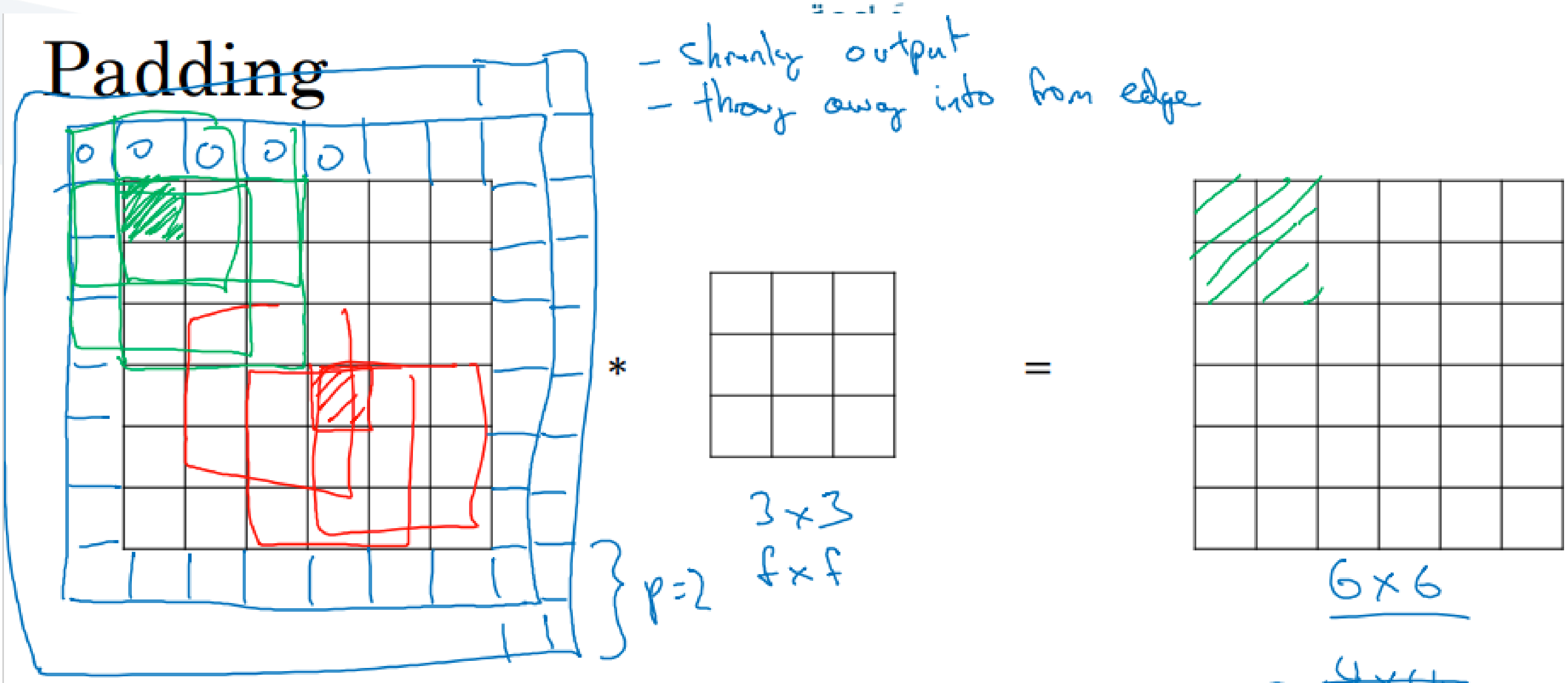
$=$

45°
70°
73°

Convolutional Neural Networks

Padding

Padding



6×6 → 8×8

n×n

n-f+1 × n-f+1

6-3+1=4

n+2p-f+1 × n+2p-f+1

6+2-3+1 × ___ = 6×6

p = padding = 1

Valid and Same convolutions

جامعة المنارة

(→ no padding)

"Valid": $n \times n \; * \; f \times f \rightarrow \underline{n-f+1} \times n-f+1$

$6 \times 6 \; * \; 3 \times 3 \rightarrow 4 \times 4$

"Same": Pad so that output size is the <u>same</u> as the input size.

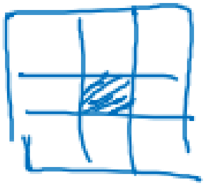
$n+2p-f+1 \times n+2p-f+1$

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

$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad | \quad 5 \times 5 \; (f=5) \quad p=2$

f is usually odd

1×1, 3×3, 5×5, 7×7



Convolutional Neural Networks

Strided convolutions

Strided convolution

جَامعة
المَنارة

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}

7×7

*

3	4	4
1	0	2
-1	0	3

3×3

=

91	100	83
69	91	127
44	72	74

3×3

stride = 2

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

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

padding p $\qquad$ 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:

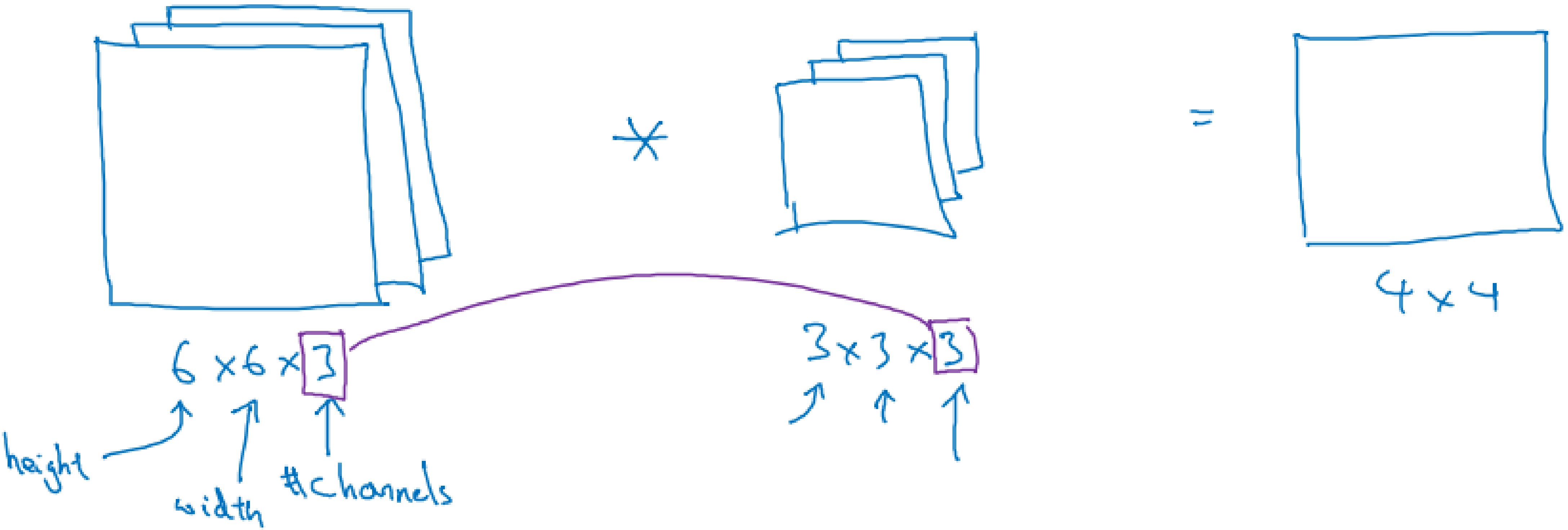
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

جامعة المنارة
MANARA UNIVERSITY

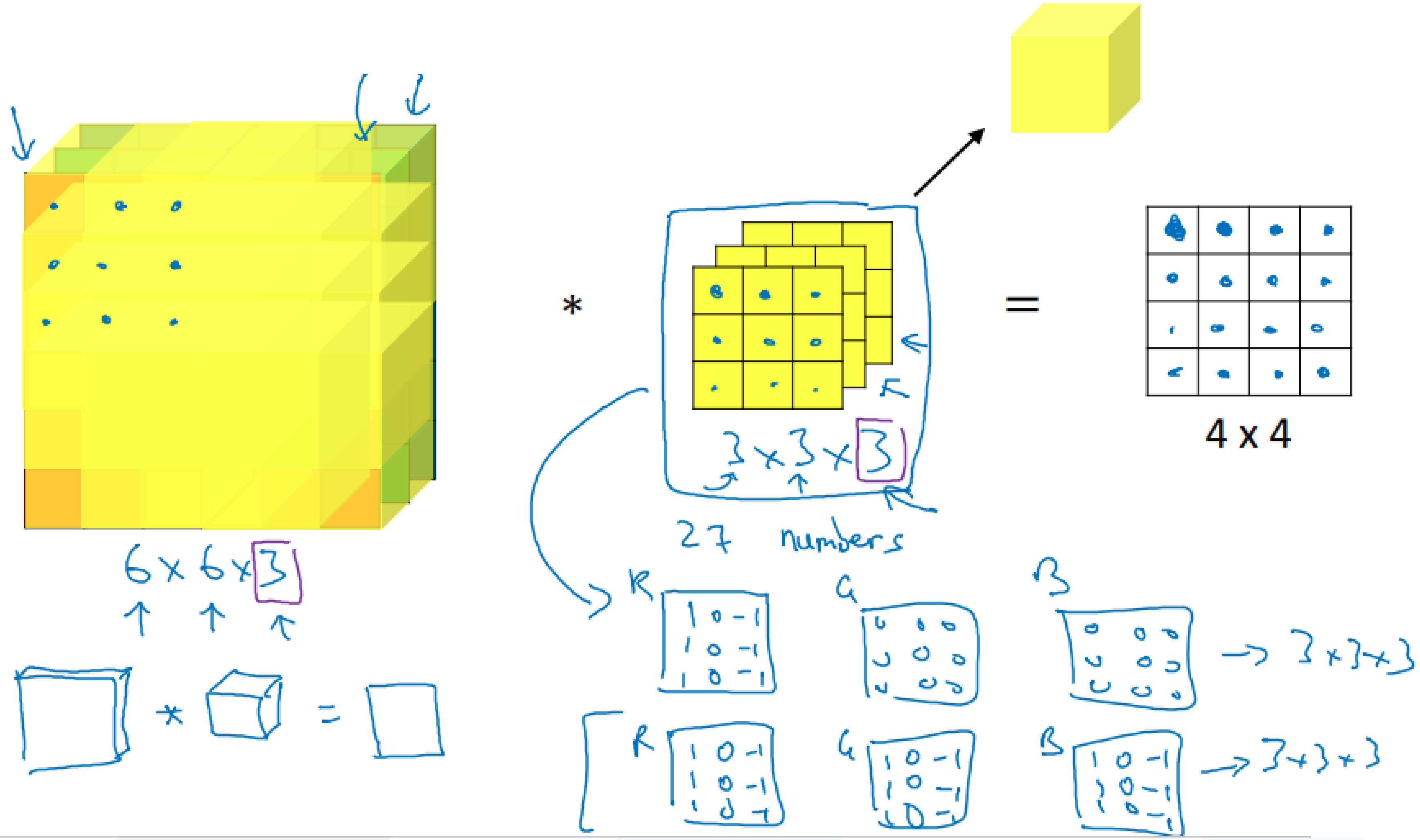
Convolutional Neural Networks

Convolutions over volumes

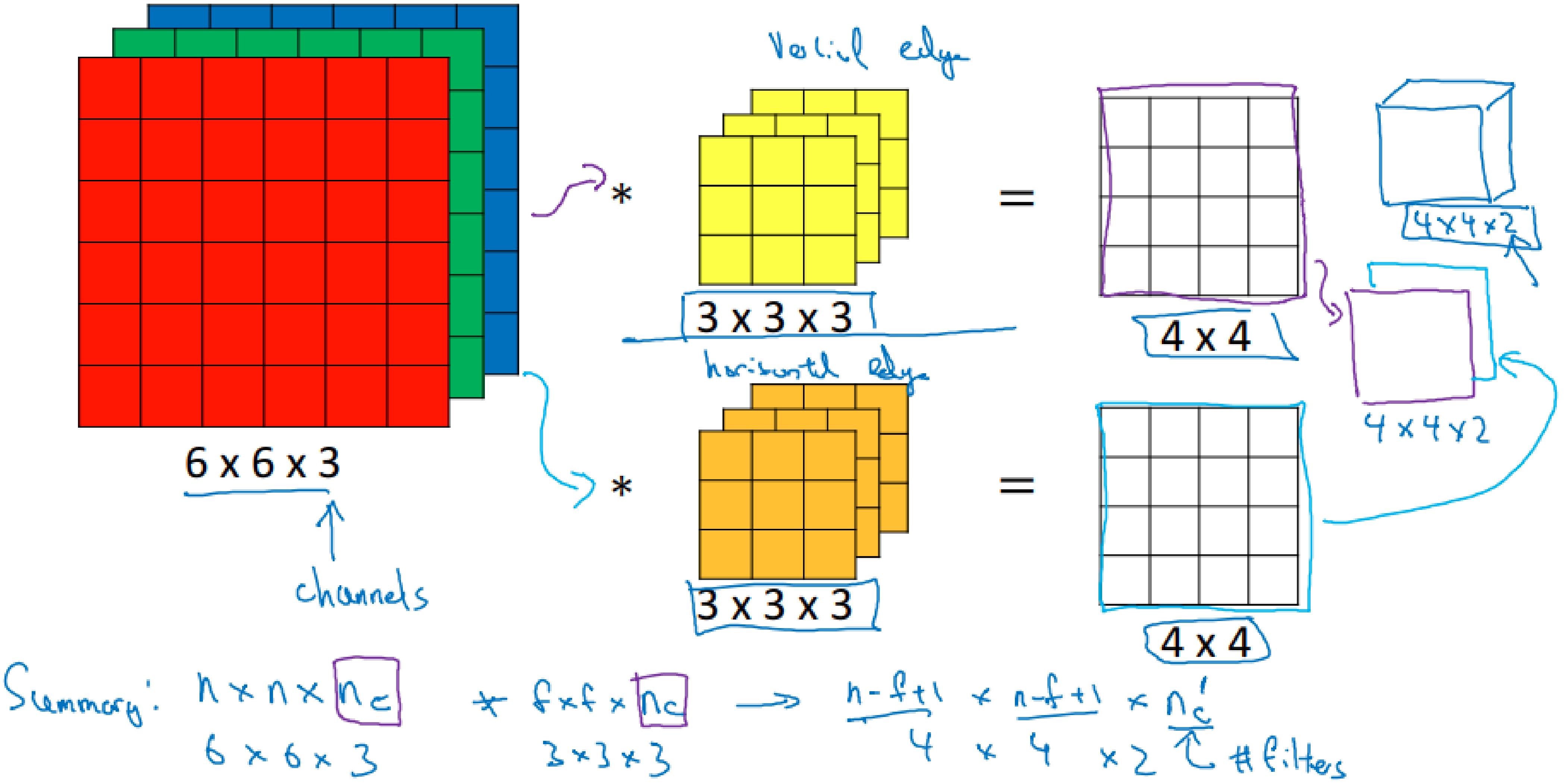
Convolutions on RGB images



Convolutions on RGB image



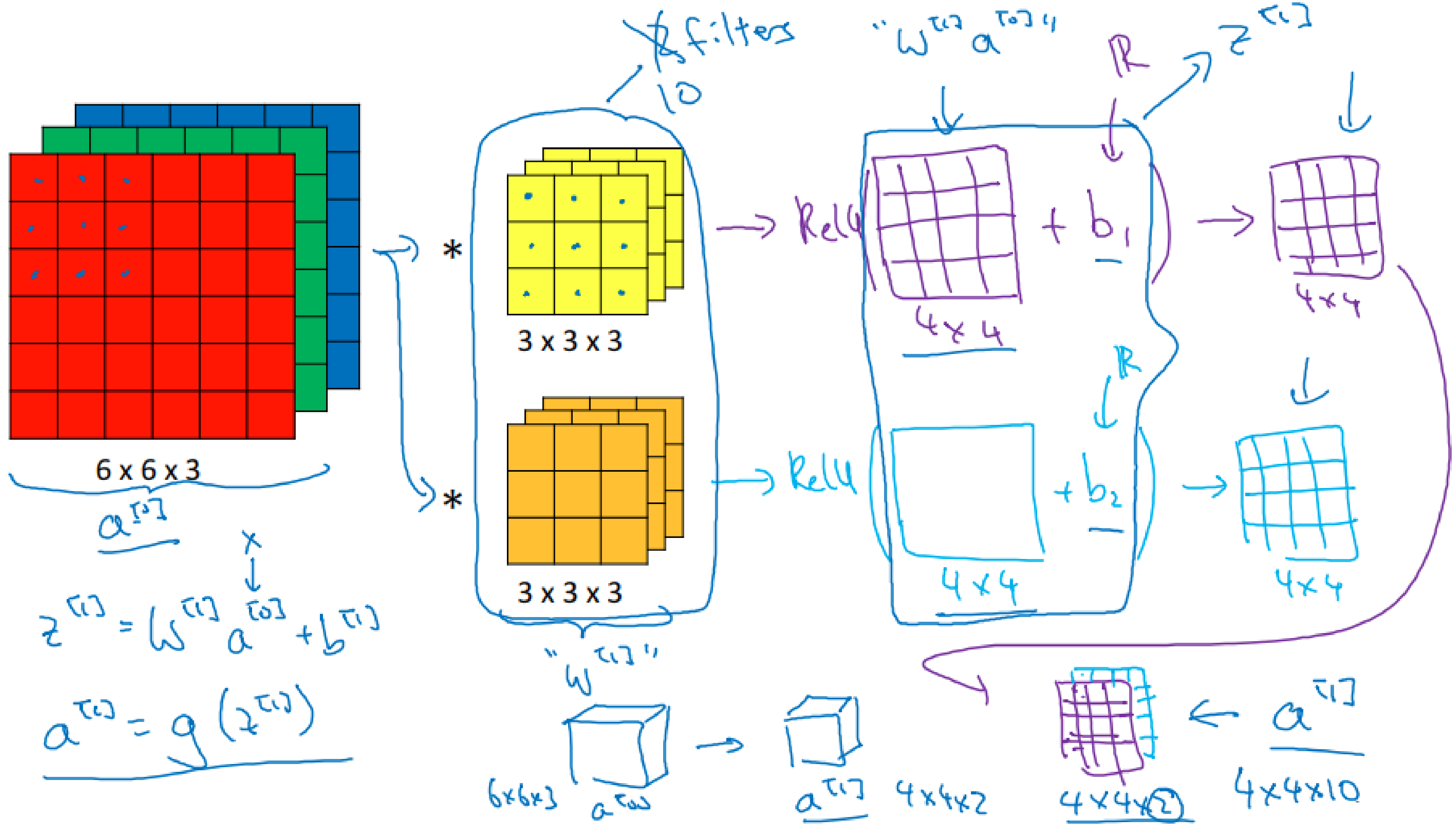
Multiple filters



Convolutional Neural Networks

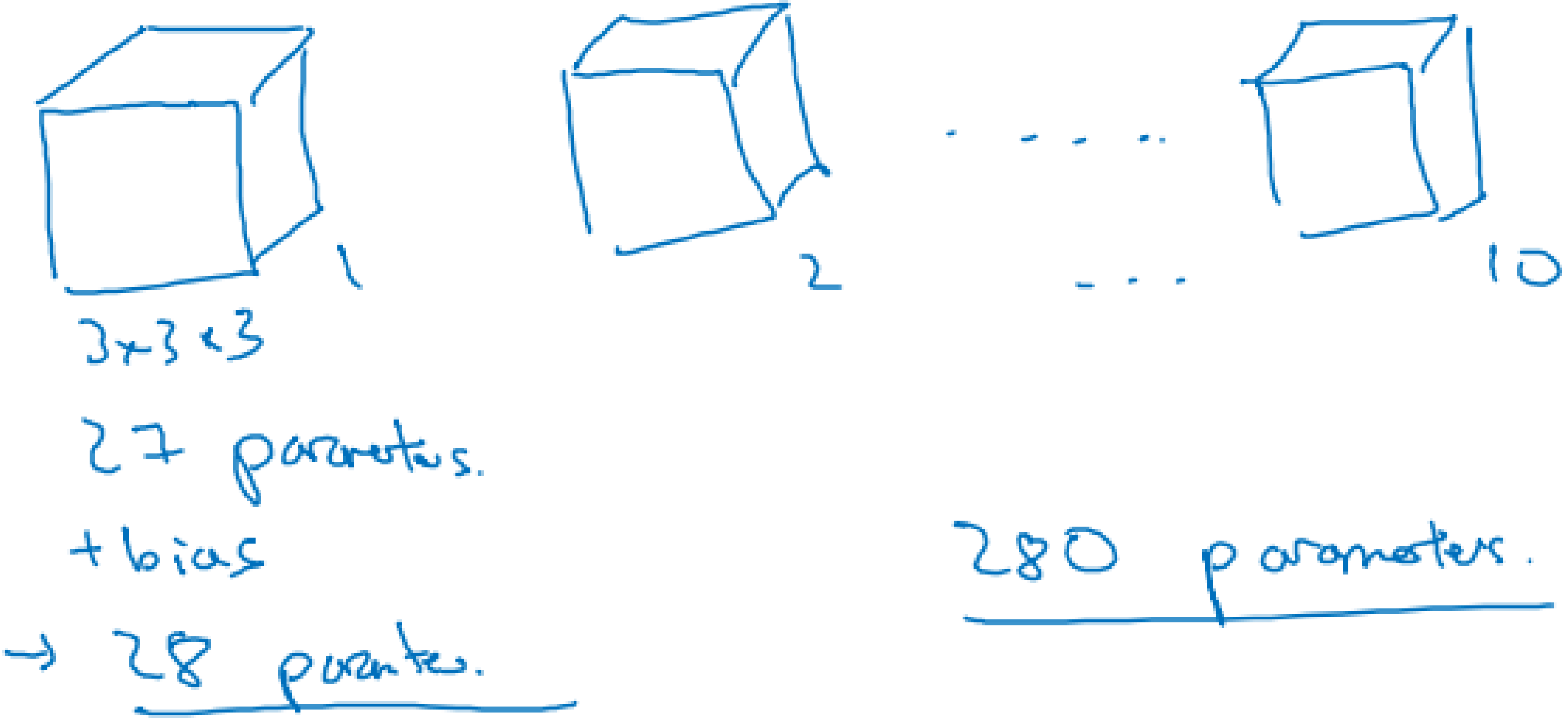
One layer of a convolutional network

Example of a layer



Number of parameters in one layer

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



Summary of notation

جَامعة المَنارة
MANARA UNIVERSITY

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

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

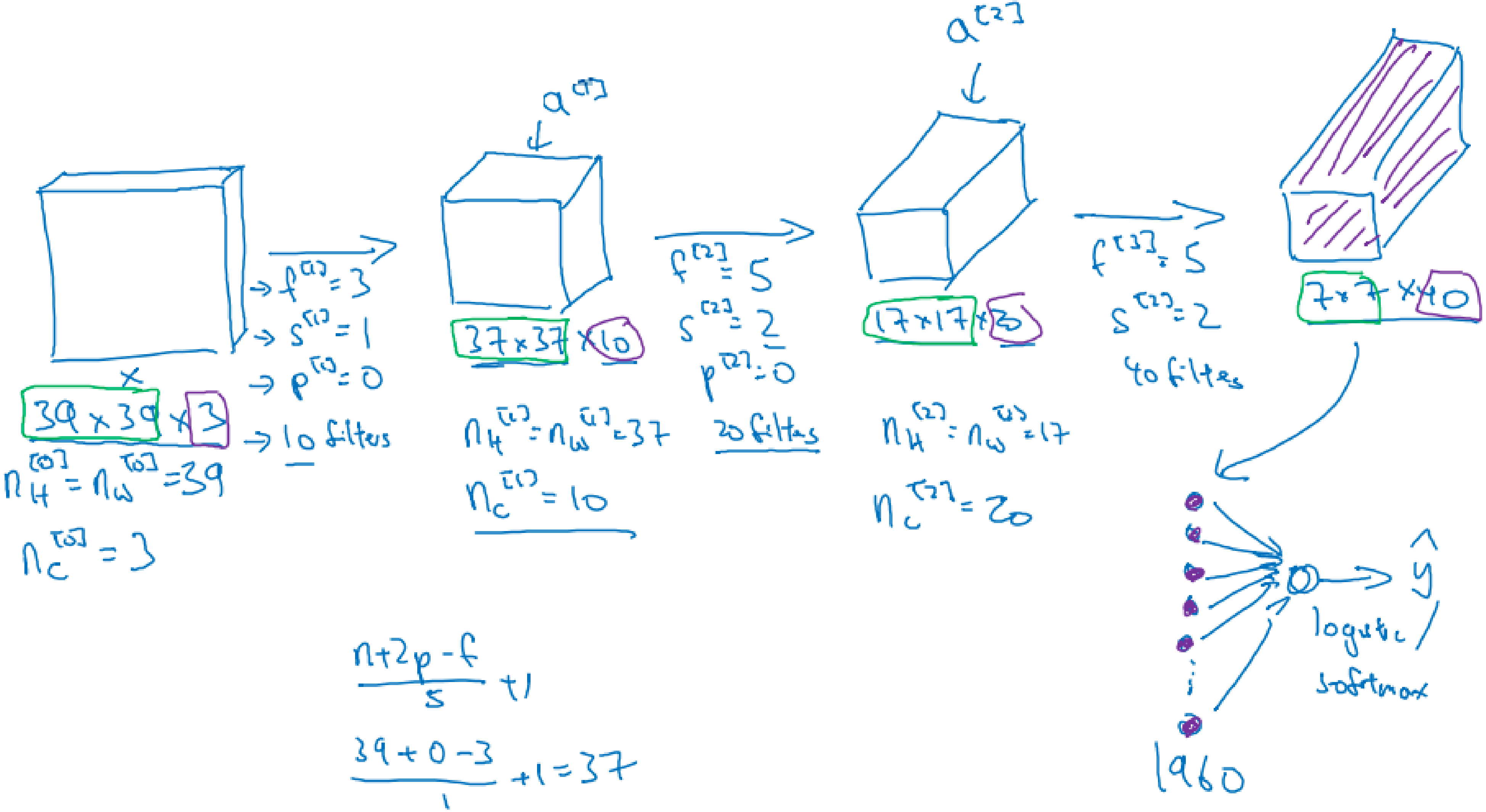
$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$n_c \times n_H \times n_W$

Convolutional Neural Networks

A simple convolution network example

Example ConvNet



Types of layer in a convolutional network:

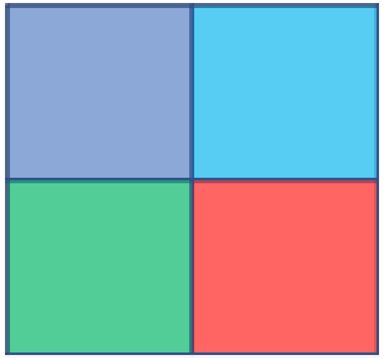
- Convolution
- Pooling
- Fully connected

Convolutional Neural Networks

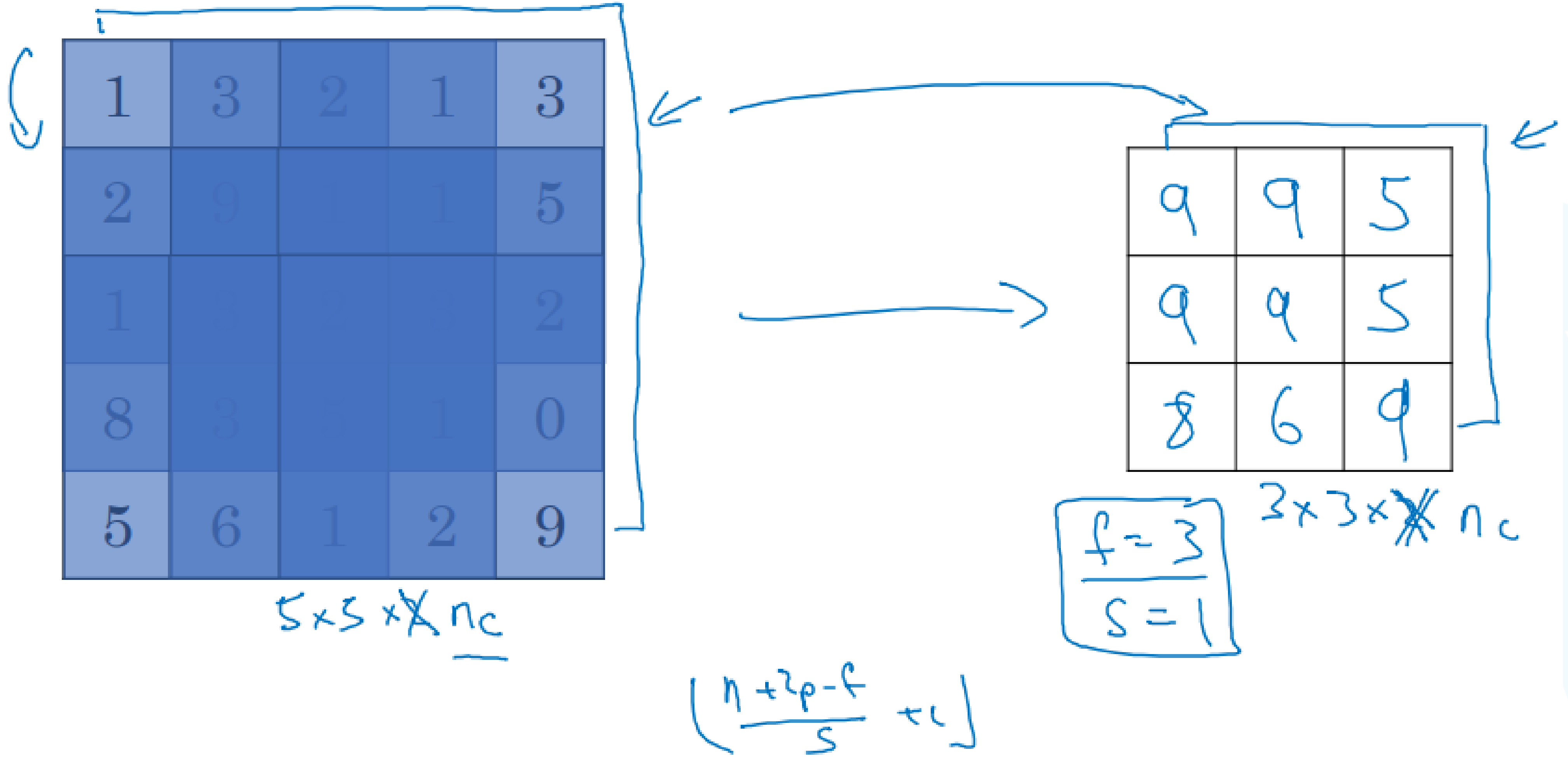
Pooling layers

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

$f = 2$

$s = 2$

$7 \times 7 \times 1000 \rightarrow 1 \times 1 \times 1000$

Summary of pooling

Hyperparameters:

f : filter size

s : stride

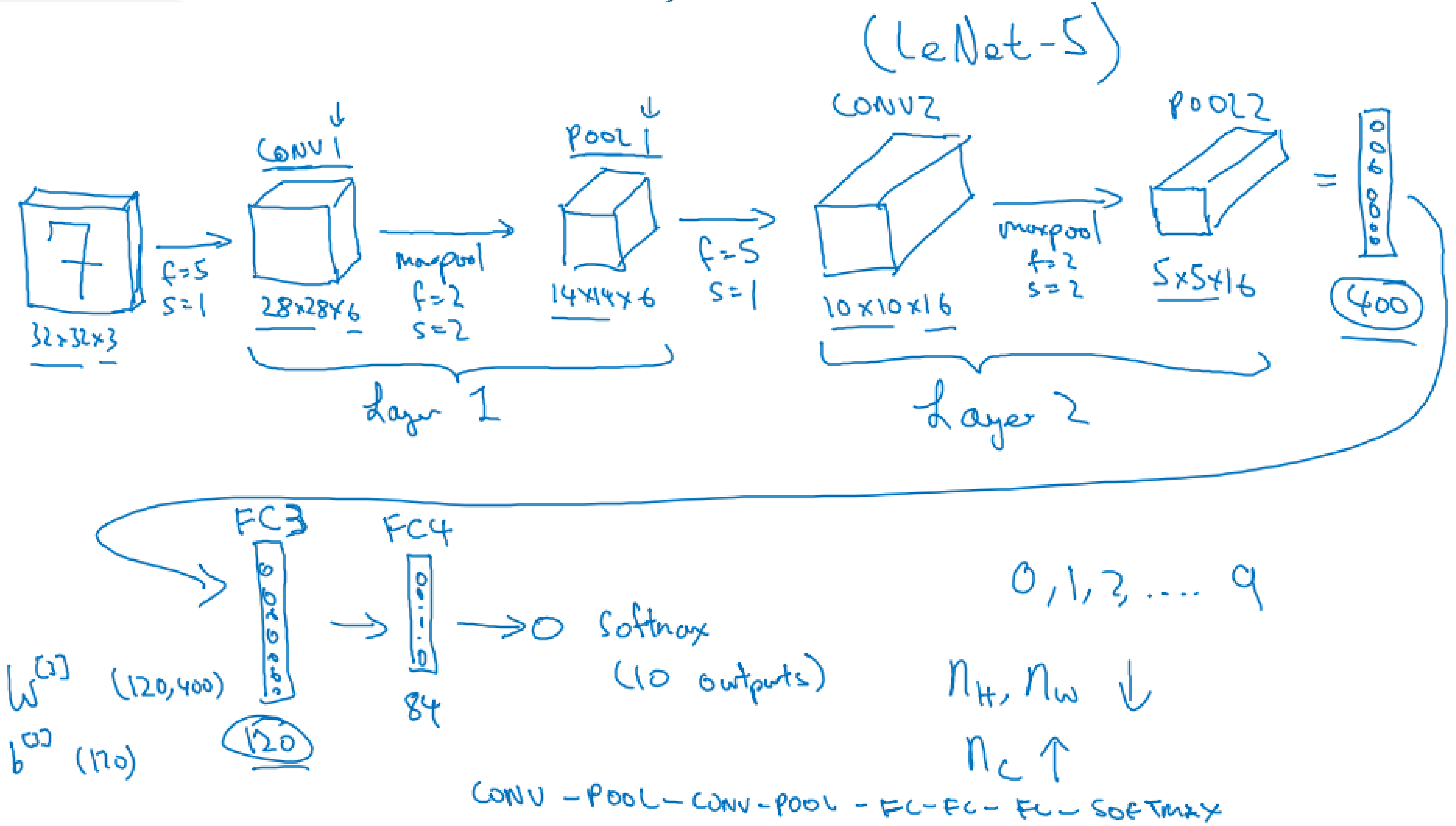
Max or average pooling

جامعة المنارة
MANARA UNIVERSITY

Convolutional Neural Networks

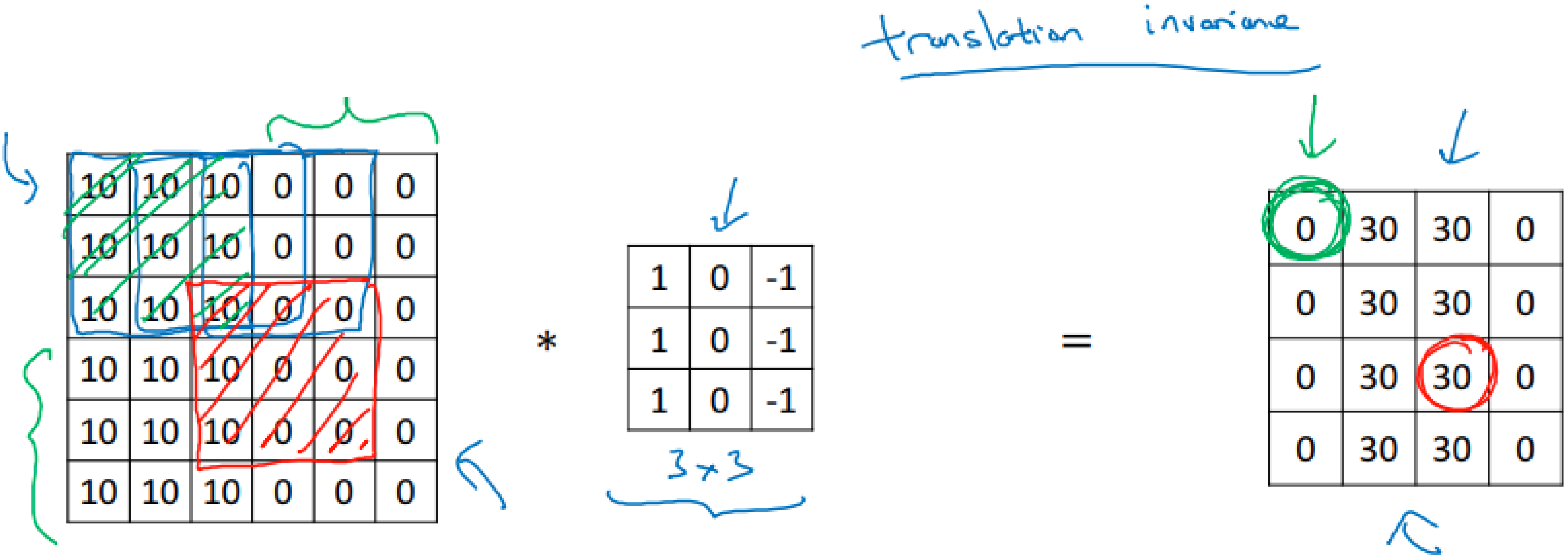
Convolutional neural network example

Neural network example



Why convolutions

جامعة المنارة

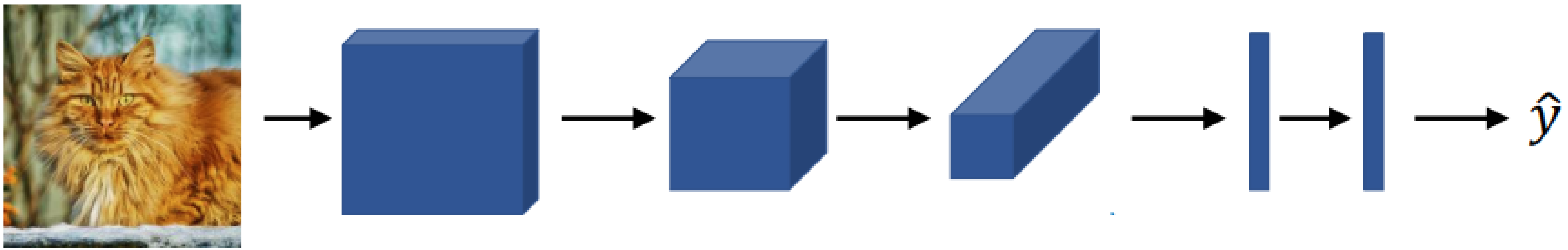


Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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

Putting it together

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



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