Neural network

Convolution neural network

Khiem Nguyen

Email	khiem.nguyen@glasgow.ac.uk
MS Teams	khiem.nguyen@glasgow.ac.uk
Whatsapp	+44 7729 532071 (Emergency only)

May 18, 2025

Table of Contents

Computer vision problems

Figure: Image classification: Is it a cat? (0/1)n

Computer vision problems

Figure: Neural style transfer

Computer vision problems

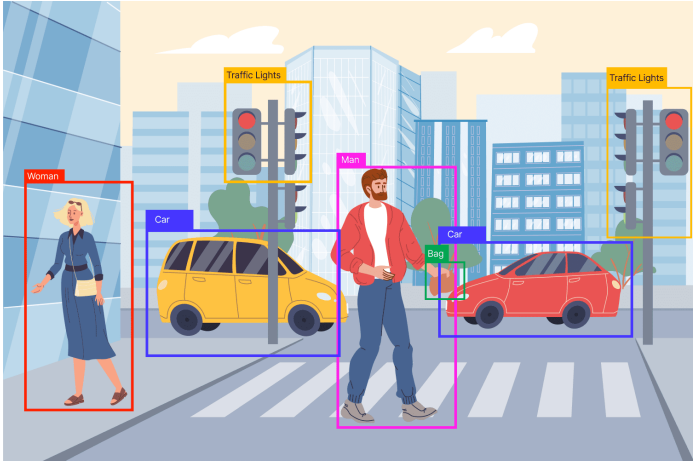


Figure: Object detection

Deep learning on large images

Figure: Sensai Saitama – One Punch Man. Image size: $(1632, 2912, 3)$

- Flattening the image into a 1-dimensional vector, we obtain a vector of approximately 14.2 million features: $1632 \times 2912 \times 3 = 14.257.152$
- Even for a small image of size $(64, 64, 3)$, we have $64 \times 64 \times 3 = 12288$ features.

Deep learning on large images

- Assume that we have a neural network with only one 1 hidden layer of 1000 neurons, then the number of learnable parameters would be

$$(\underbrace{64 \times 64 \times 3}_{12288} + 1) \times 1000 + (1000 + 1) \times 1 = 12290001 \quad (\approx 12.3 \text{ millions})$$

- For an image of quite large size $(h, w, 3)$, by using a neural network with 1 hidden layer of 1000 neurons, the number of learnable parameters is

$$(h \times w \times 3 + 1) \times 1000 + (1000 + 1) \times 1 \approx (h \times w \times 3 + 1) \times 1000 \quad \text{(very large number)}$$

➠ Clearly, multi-layer perceptrons are not suitable for such type of problem.

Computer vision problem

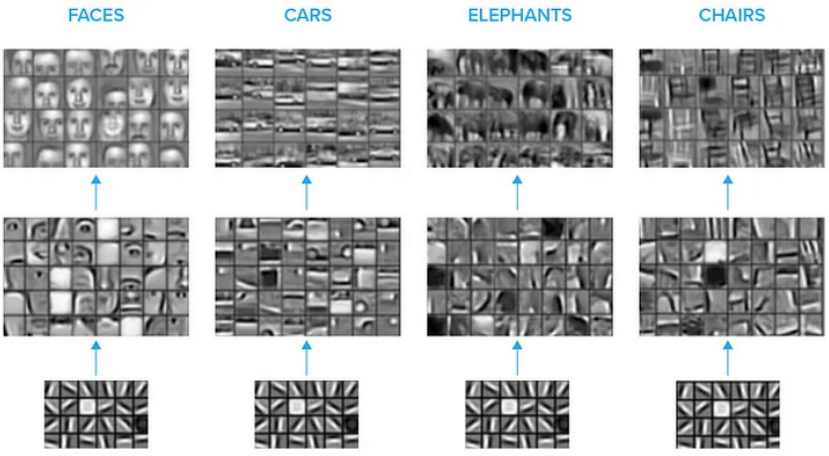


Figure: Feature detections in different layers

Edge detection in computer vision

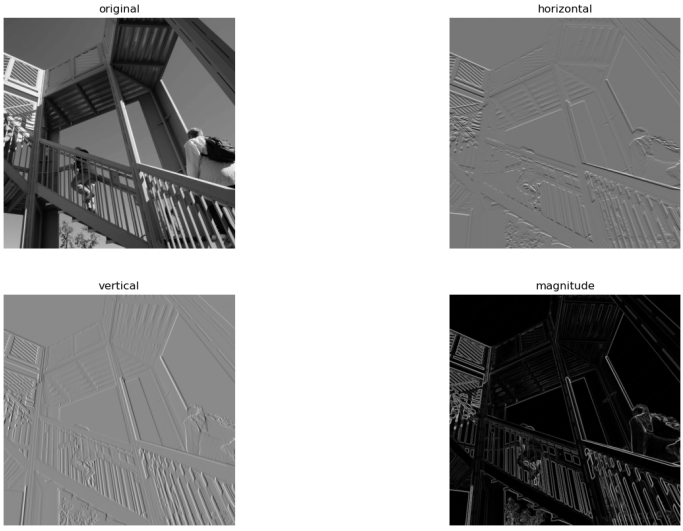


Figure: Edge detection using Sobel filter

Vertical edge detection

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			

Computation:

$$3 \cdot 1 + 0 \cdot 1 + 1 \cdot (-1) + 1 \cdot 1 + 5 \cdot 0 + 8 \cdot (-1) + 2 \cdot 1 + 7 \cdot 0 + 2 \cdot (-1) = -5$$

Vertical edge detection

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		

Computation:

$$0 \cdot 1 + 1 \cdot 1 + 2 \cdot (-1) + 5 \cdot 1 + 8 \cdot 0 + 9 \cdot (-1) + 7 \cdot 1 + 2 \cdot 0 + 5 \cdot (-1) = -4$$

Vertical edge detection

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

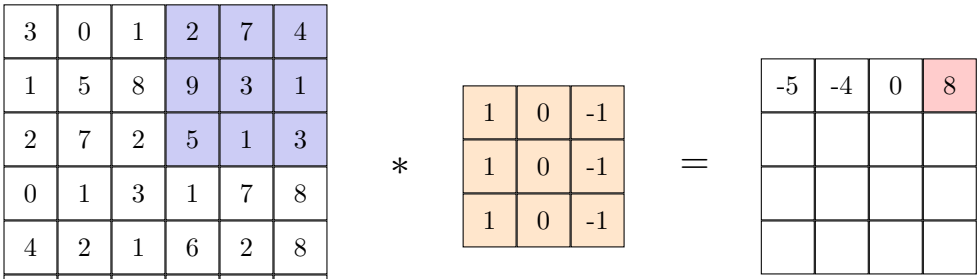
$=$

-5	-4	0	

Computation:

$$1 \cdot 1 + 2 \cdot 1 + 7 \cdot (-1) + 8 \cdot 1 + 9 \cdot 0 + 3 \cdot (-1) + 2 \cdot 1 + 5 \cdot 0 + 1 \cdot (-1) = 0$$

Vertical edge detection



Computation:

$$2 \cdot 1 + 7 \cdot 1 + 4 \cdot (-1) + 9 \cdot 1 + 3 \cdot 0 + 1 \cdot (-1) + 5 \cdot 1 + 1 \cdot 0 + 3 \cdot (-1) = 8$$

Vertical edge detection

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			

Computation:

$$1 \cdot 1 + 5 \cdot 1 + 8 \cdot (-1) + 2 \cdot 1 + 7 \cdot 0 + 2 \cdot (-1) + 0 \cdot 1 + 1 \cdot 0 + 3 \cdot (-1) = -10$$

Vertical edge detection

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		

Computation:

$$5 \cdot 1 + 8 \cdot 1 + 9 \cdot (-1) + 7 \cdot 1 + 2 \cdot 0 + 5 \cdot (-1) + 1 \cdot 1 + 3 \cdot 0 + 1 \cdot (-1) = -2$$

Vertical edge detection

Vertical edge detection

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

Computation:

$$1 \cdot 1 + 7 \cdot 1 + 8 \cdot (-1) + 6 \cdot 1 + 1 \cdot 0 + 8 \cdot (-1) + 2 \cdot 1 + 3 \cdot 0 + 9 \cdot (-1) = -16$$

Vertical edge detection

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

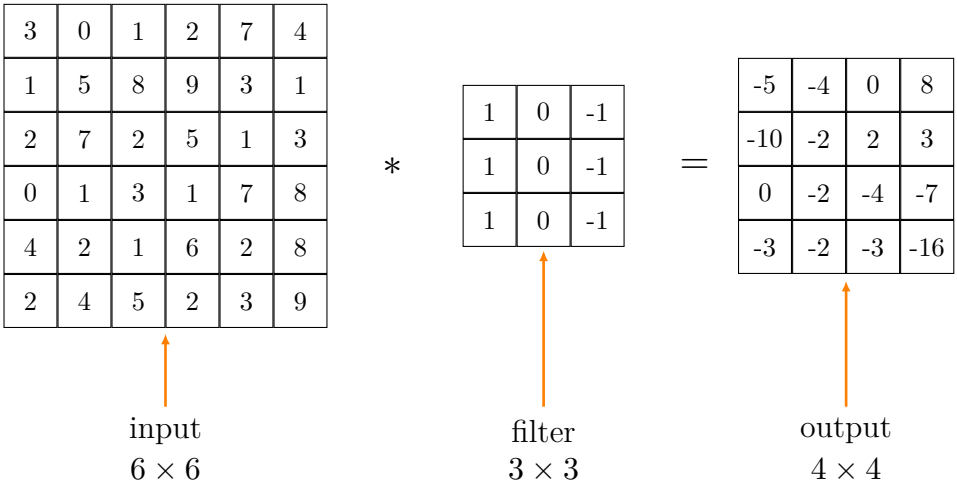
$*$

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

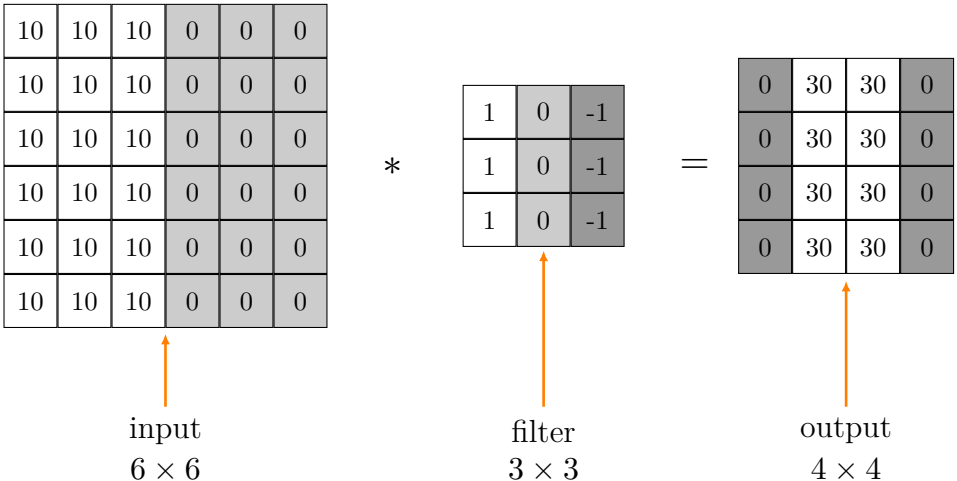
$=$

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

Vertical edge detection



Vertical edge detection



Vertical and horizontal edge detection

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

vertical

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

another vertical filter

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

$*$

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

Vertical

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

Sobel filter

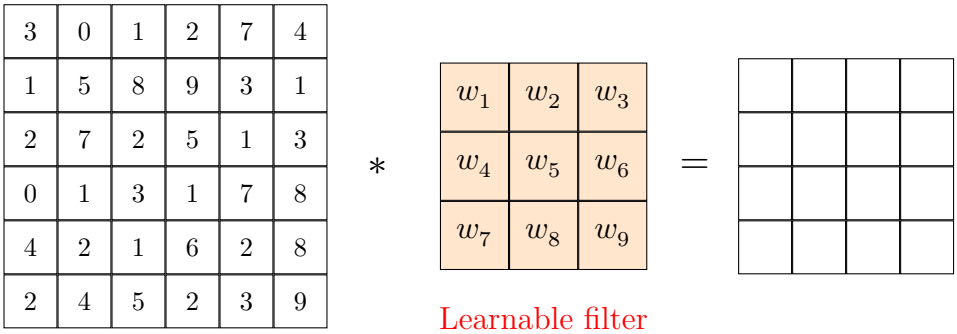
3	0	-3
10	0	-10
3	0	-3

Scharr filter

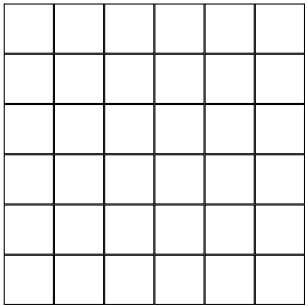
Questions:

- How many filter are there we have to learn?
- Can we learn the filter by a machine learning framework?

Learning to detect edges

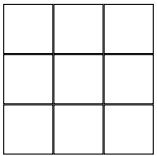


Padding

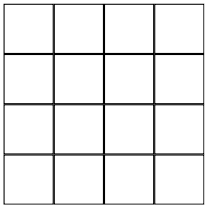


6×6

$*$

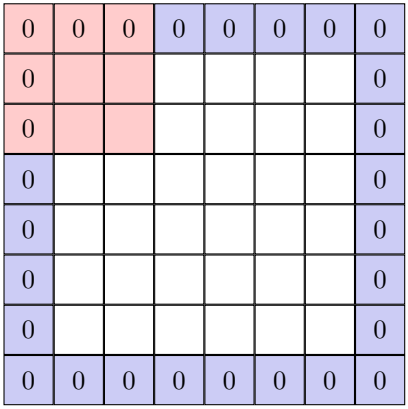


$=$



4×4

Padding

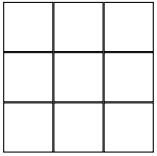


$6 \times 6 \longrightarrow 8 \times 8$

padding $p = 1$

$n \times n \longrightarrow (n+p) \times (n+p)$

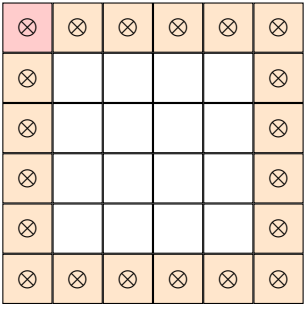
$*$



3×3

$f \times f$

$=$



$\otimes \neq 0$ in general

$4 \times 4 \longrightarrow 6 \times 6$

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

$\rightarrow (n+2p-f+1) \times n+2p-f+1$

Valid and same convolutions

➢ *Valid convolution*: No padding

$$\begin{array}{lllll} n \times n & * & f \times f & \longrightarrow & (n-f+1) \times (n-f+1) \\ 6 \times 6 & * & 3 \times 3 & \longrightarrow & 4 \times 4 \end{array}$$

➢ *Same convolution*: Pad so that output size is the same as the input size

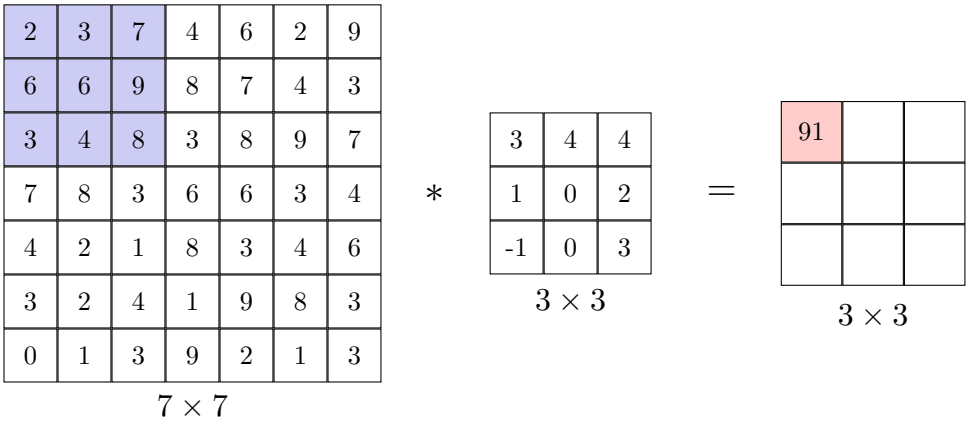
input size	$n \times n$
output size after padding	$(n+2p-f+1) \times (n+2p-f+1)$

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

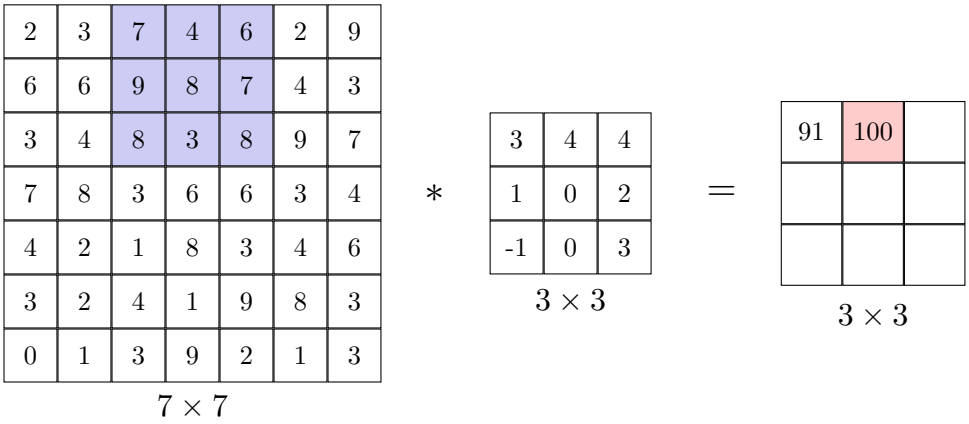
For this reason, f is usually odd. Examples:

- If $f \times f = 3 \times 3$, the padding for same convolution is $p = \frac{3-1}{2} = 1$.
- If $f \times f = 5 \times 5$, the padding for same convolution is $p = \frac{5-1}{2} = 2$

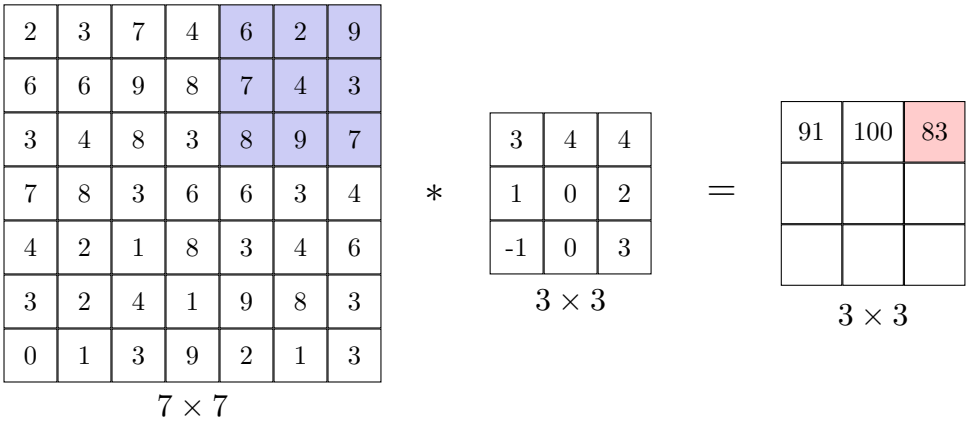
Strided convolution



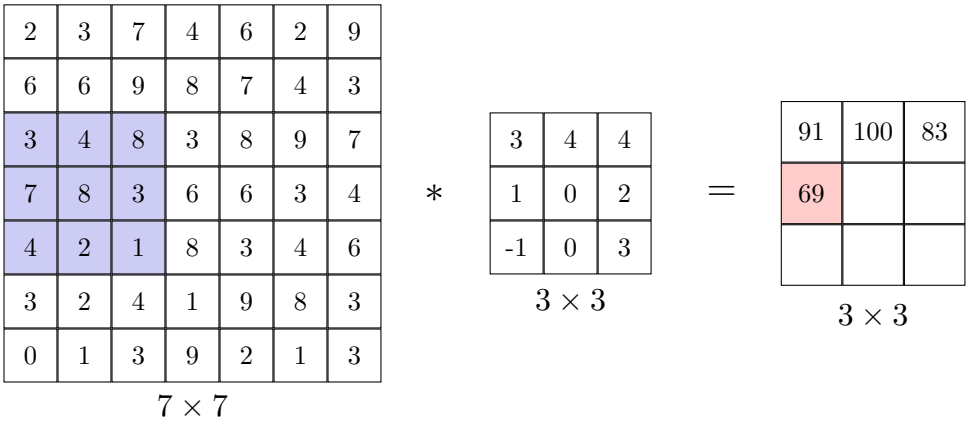
Strided convolution



Strided convolution

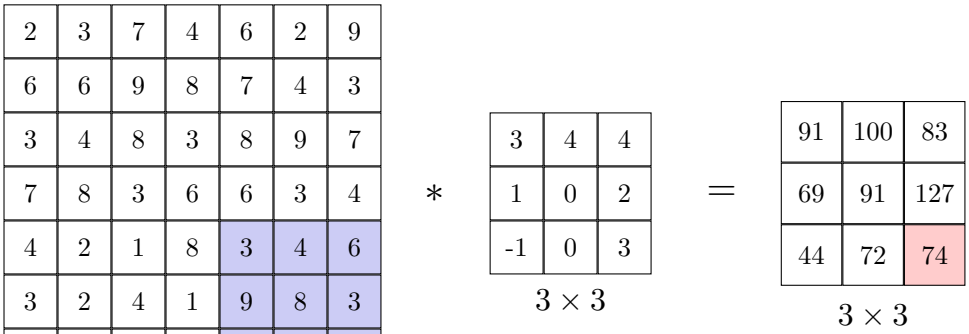


Strided convolution



Strided convolution

Strided convolution



using stride s : $(n \times n) \quad * \quad (f \times f) \quad = \quad \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

Example: With $f = 3, p = 0, s = 2$, the output size is $\left\lfloor \frac{7+0-3}{2} + 1 \right\rfloor \times \left\lfloor \frac{7+0-3}{2} + 1 \right\rfloor = 3 \times 3$

A note on cross-correction versus convolution

- Convolution in mathematics: Assume that we have two sequences $a = \{a[i]\}_{i=-\infty}^{\infty}$ and $b = \{b[i]\}_{i=-\infty}^{\infty}$, the convolution between f and g is given by

$$(a * b)[i] = \sum_{k=-\infty}^{\infty} a[k] \cdot b[k-i]$$

- A nice property of convolution in mathematics:

$$(a * b) * c = a * (b * c)$$