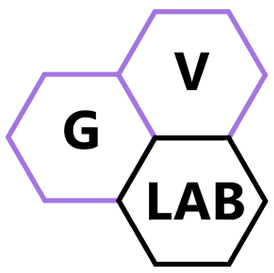
Convolution

Padding and strides

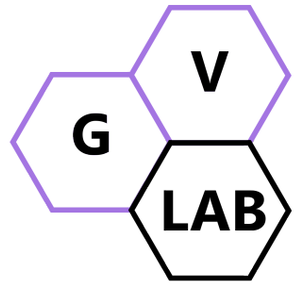
Dr. Thanh-Sach LE
LTSACH@hcmut.edu.vn



GVLab:
Graphics and Vision Laboratory

Faculty of Computer Science and Engineering,
HCMUT

2 Contents

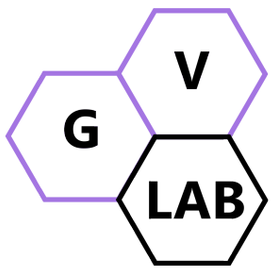


- Zero Padding
- Non-unit strides

Convolution: zero padding

Half padding

Dr. Thanh-Sach LE
LTSACH@hcmut.edu.vn



GVLab:
Graphics and Vision Laboratory

Faculty of Computer Science and Engineering,
HCMUT

Convolution algorithm

Start

- ✳ Rotate the kernel 180^0
- ✳ Flatten it to a vector

1

- ✳ Padding zero to input
- ✳ Allocate output buffer

2

- ✳ Align the kernel window with the top-left of the input

3

- ✳ Extract the sub-image overlapped with the kernel
- ✳ Flatten the sub-image to a vector
- ✳ Take dot-product with the flattened kernel
- ✳ Fill the result to output

4

all output values assigned?

No

- ✳ Slide the kernel window to next position on the input

5

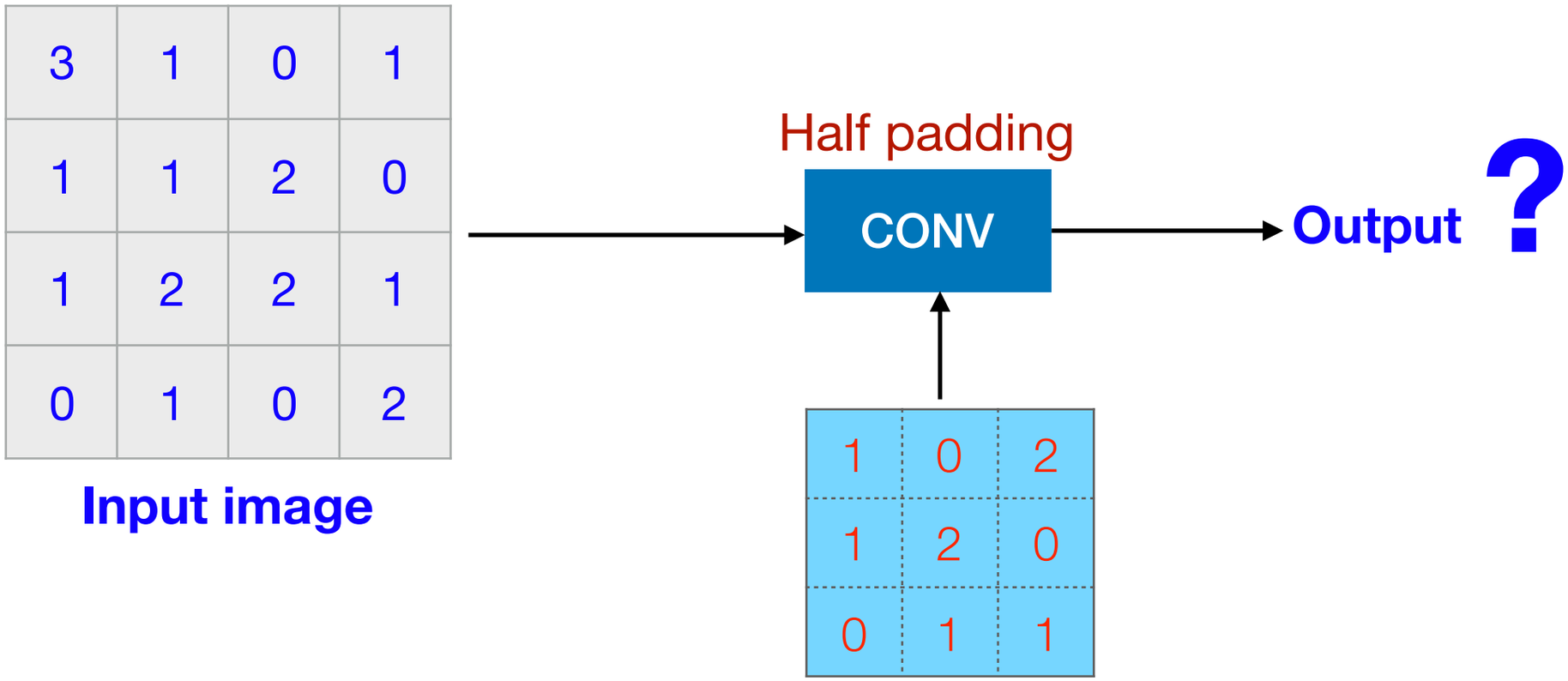
Yes

End

5

Convolution: zero padding

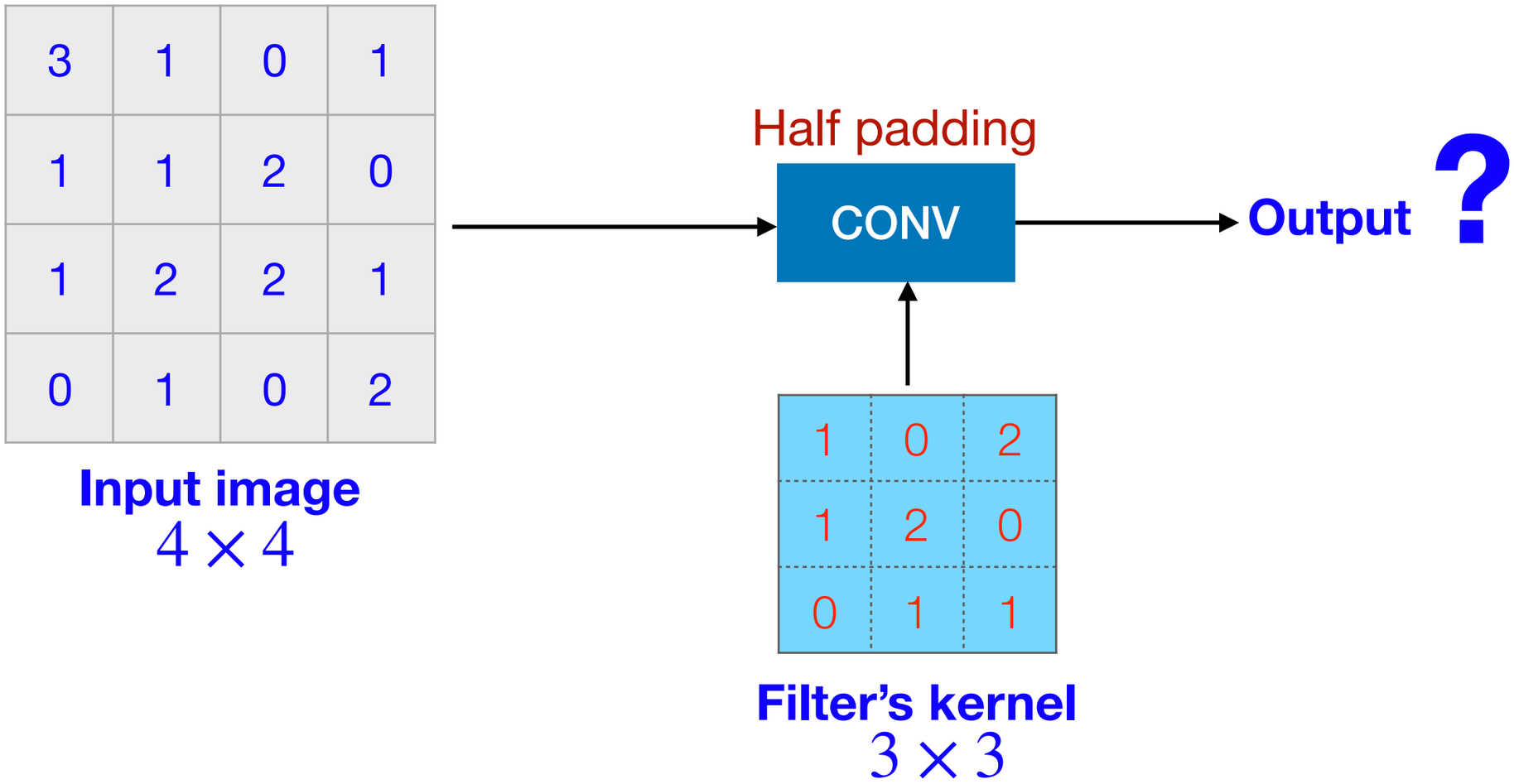
Half padding



6

Convolution: zero padding

Half padding



$$\textbf{padding size} = \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor = 1 \times 1$$

The input is enlarged one pixel on left, right, top, and bottom

7

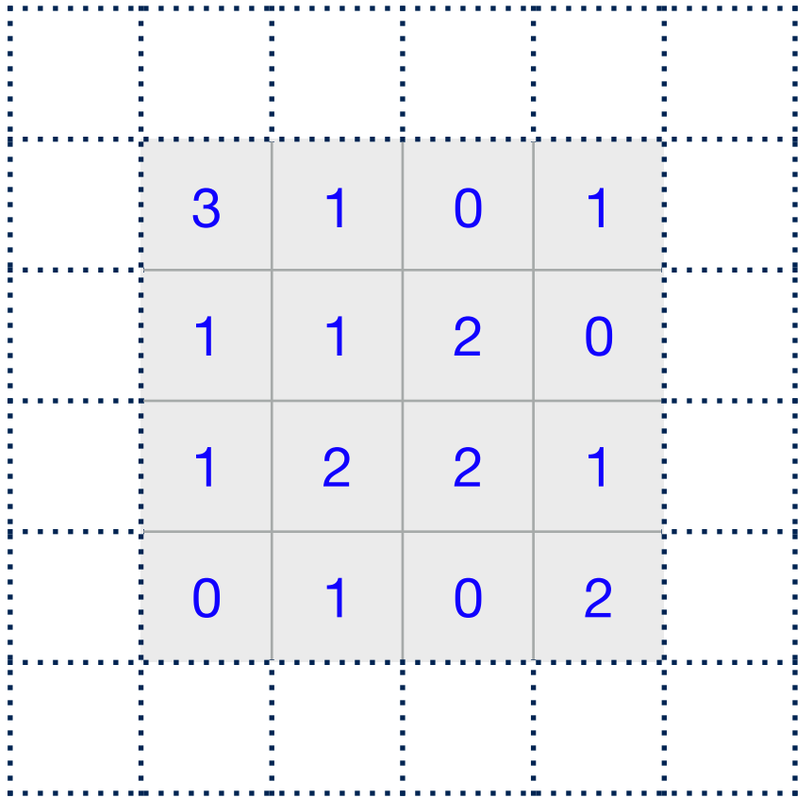
Convolution: zero padding

Half padding

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

Input image

padding size $= \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor = 1 \times 1$

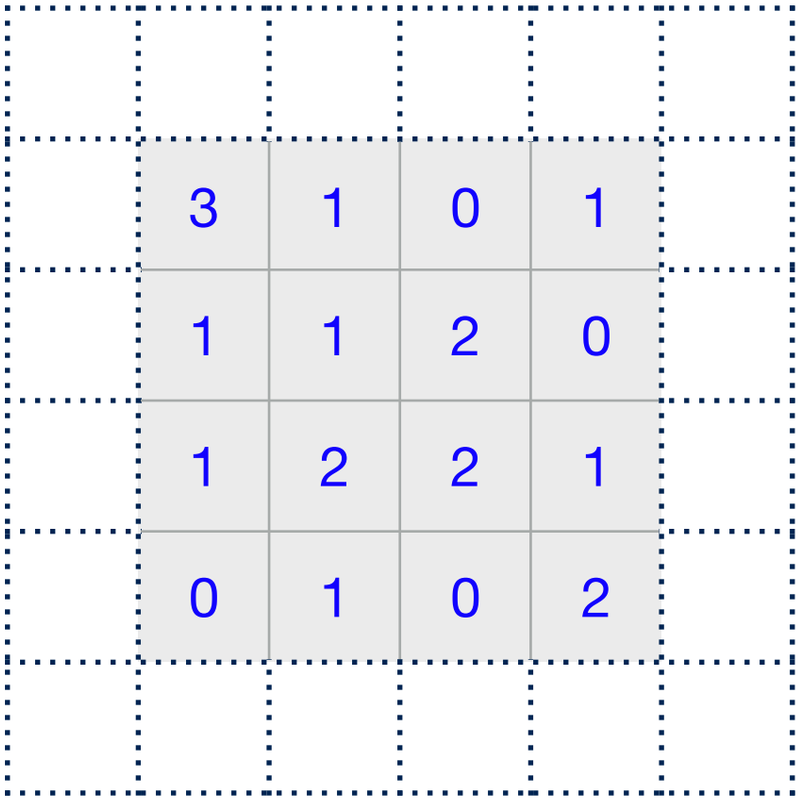


Input image, after padding

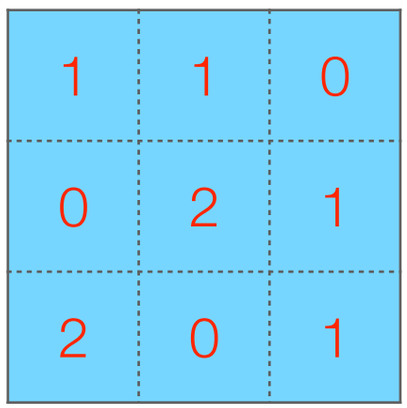
8

Convolution: zero padding

Half padding



Input image, after padding



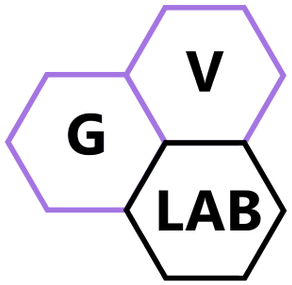
Rotated kernel

Output size = ?

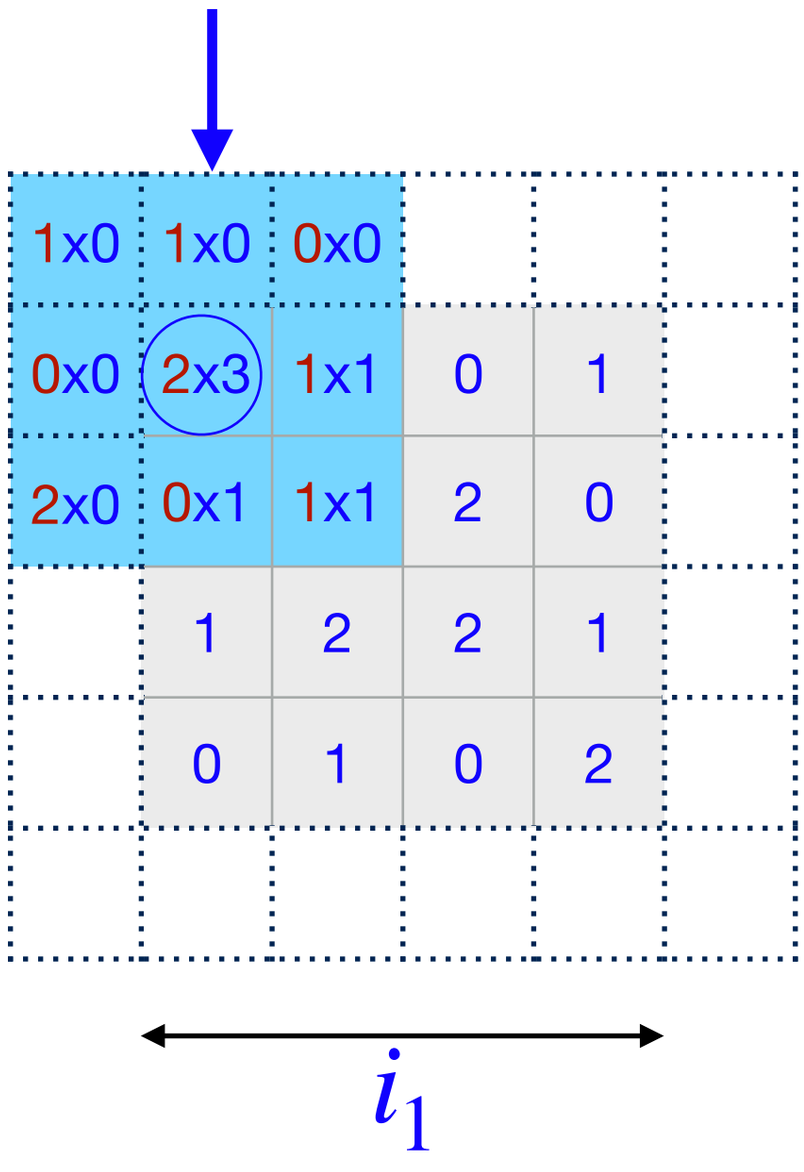
9

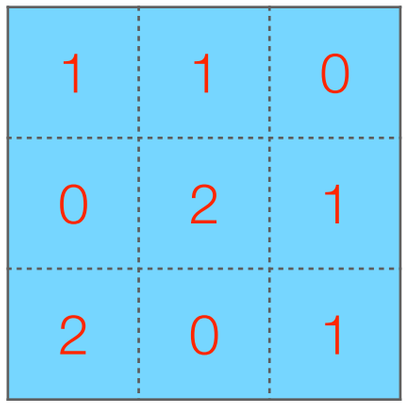
Convolution: zero padding

Half padding



The first valid position



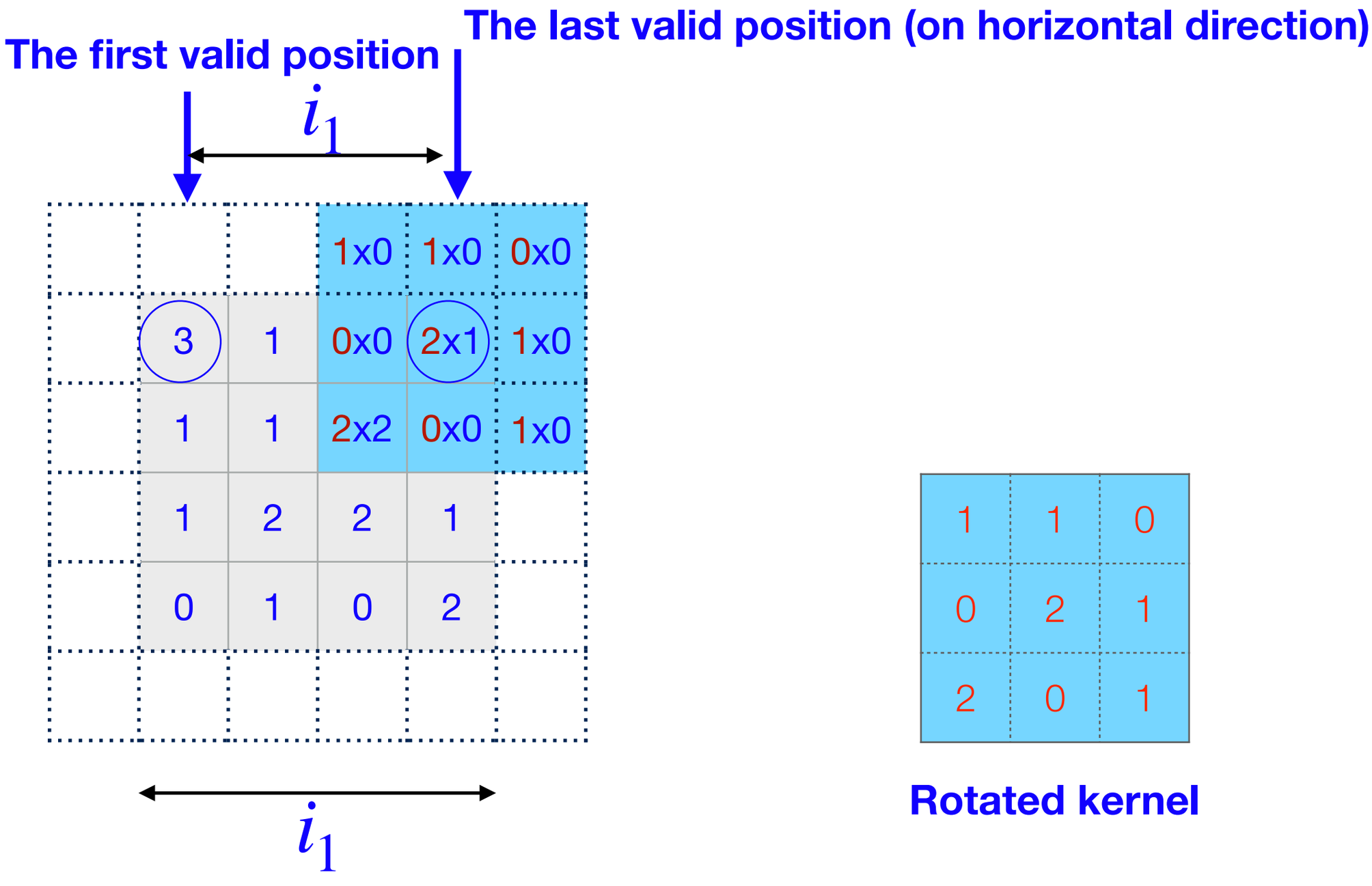


Rotated kernel

10

Convolution: zero padding

Half padding

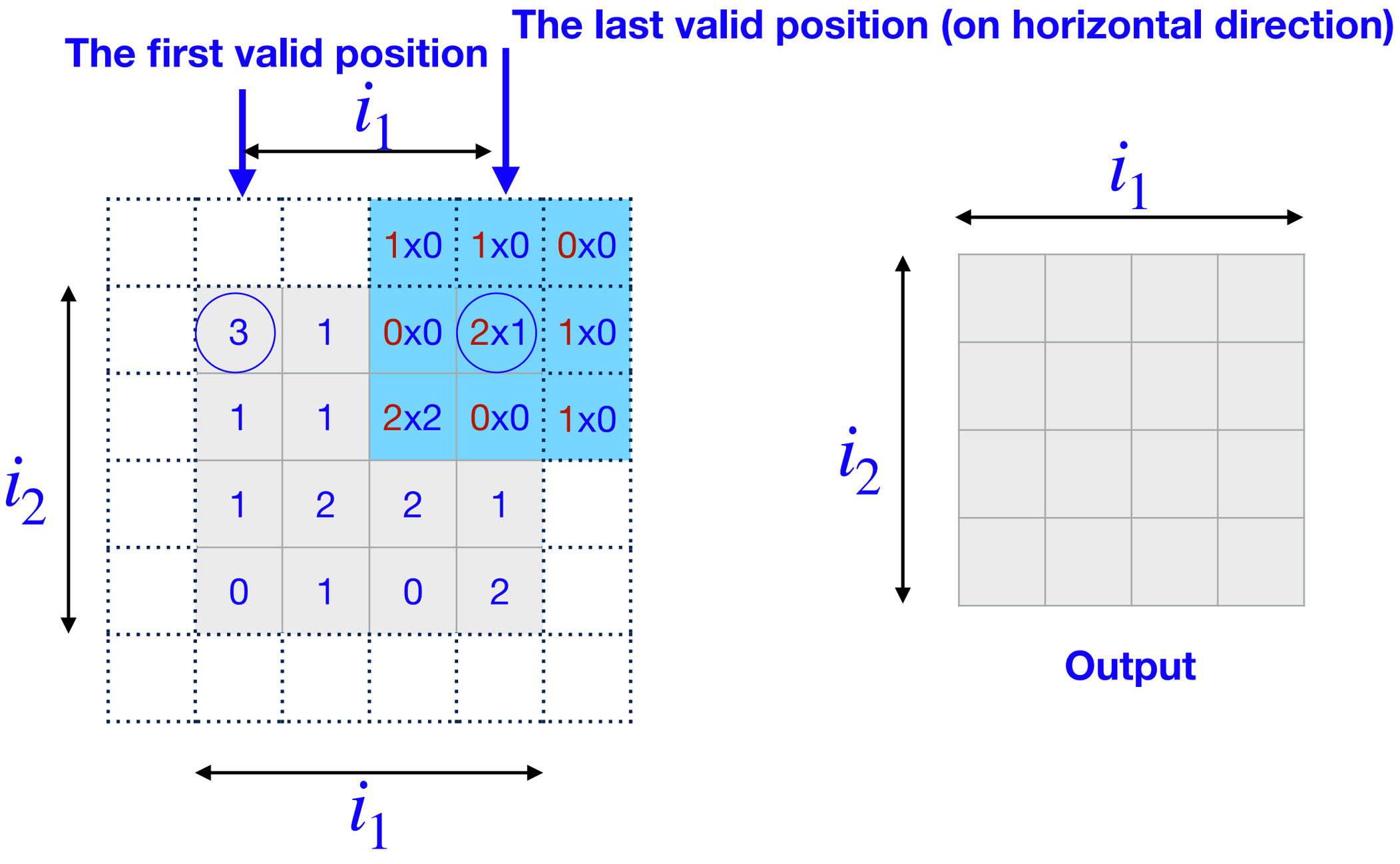


1	1	0
0	2	1
2	0	1

Rotated kernel

Convolution: zero padding

Half padding

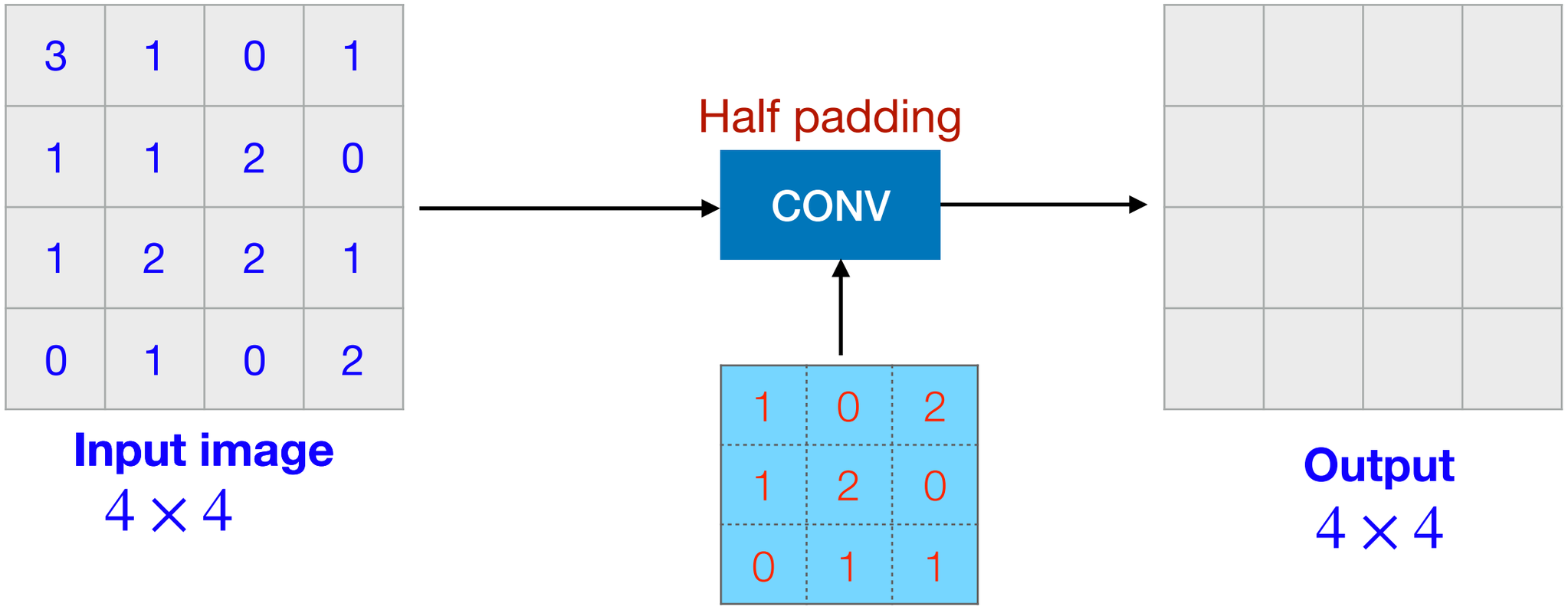


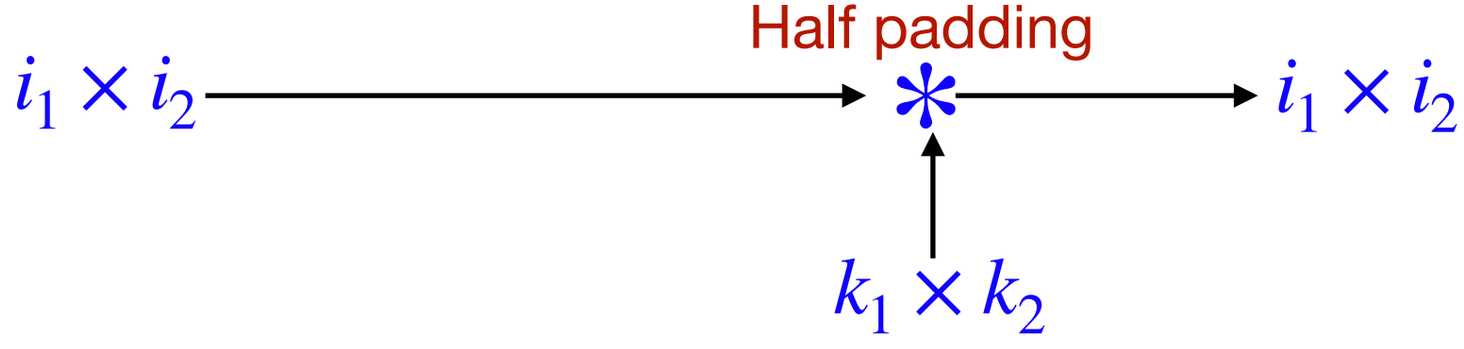
Half padding (with unit stride): input and output have the same size

12

Convolution: zero padding

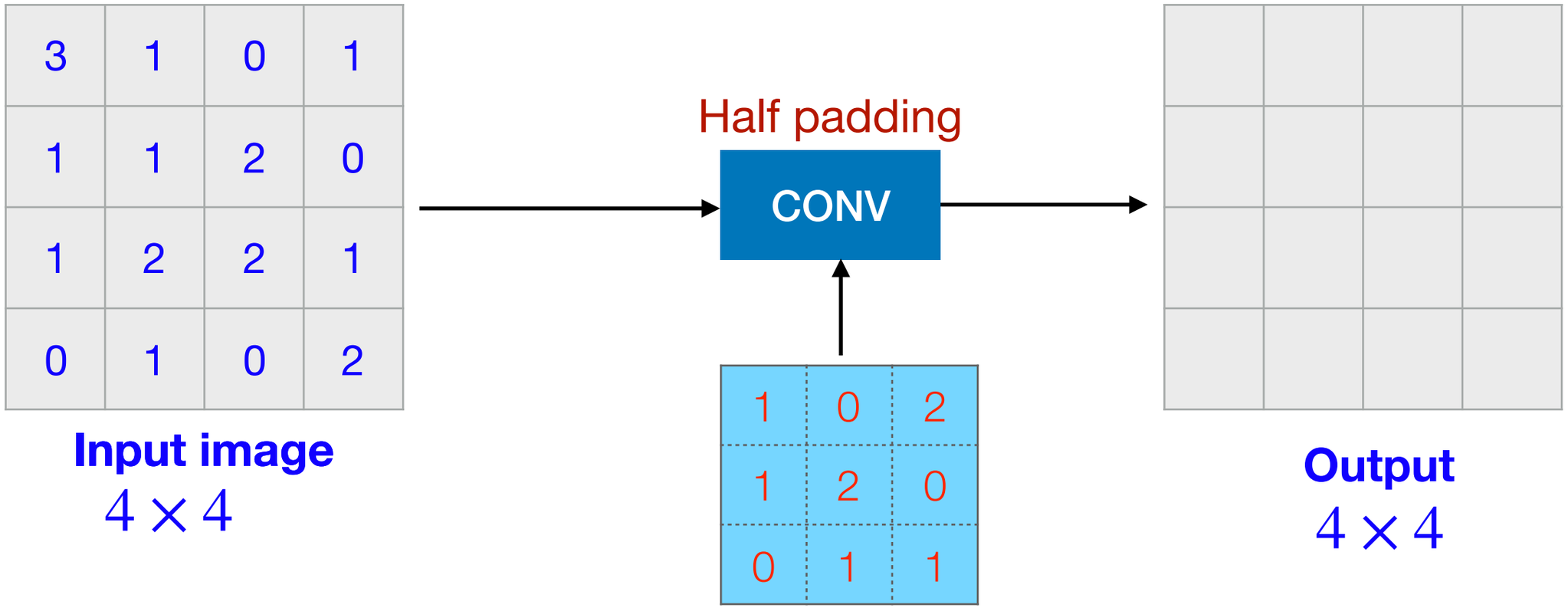
Half padding





Convolution: zero padding

Half padding

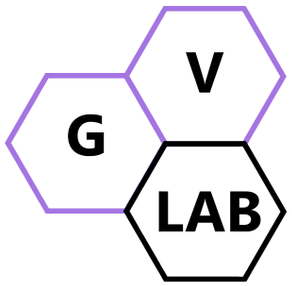


Step-by-step computation
(see next slides)

14

Convolution: zero padding

Half padding



1

1	0	2
1	2	0
0	1	1

Rotation 180º

1	1	0
0	2	1
2	0	1

Flatten to vector

1	1	0	0	2	1	2	0	1

15

Convolution: zero padding

Half padding

2

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

Input image

padding size $= \left\lfloor \frac{3}{2} \right\rfloor \times \left\lfloor \frac{3}{2} \right\rfloor = 1 \times 1$

	3	1	0	1	
	1	1	2	0	
	1	2	2	1	
	0	1	0	2	

Input image, after padding

Output

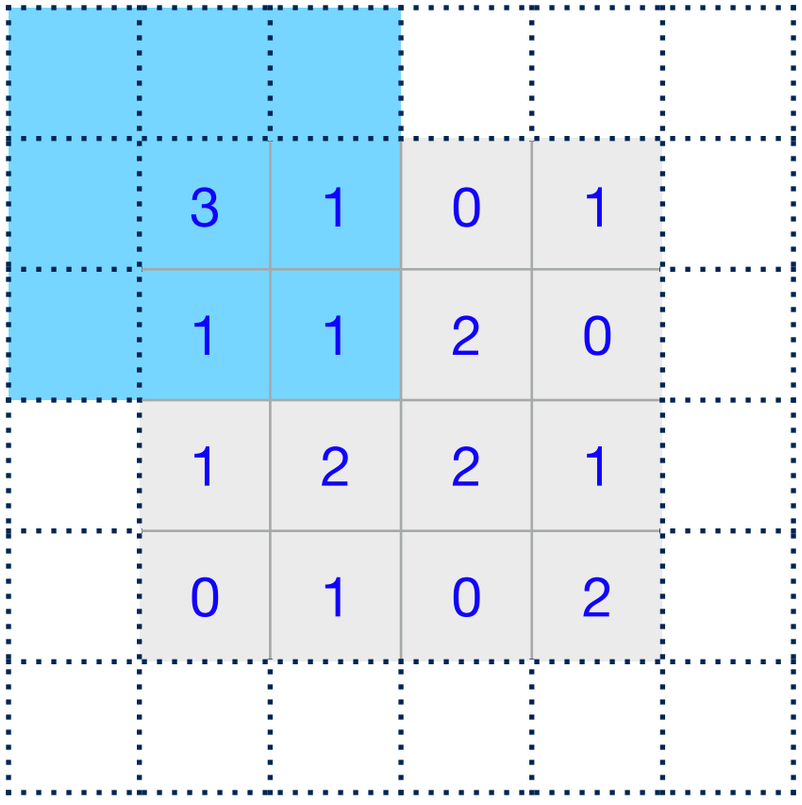
4×4

16

Convolution: zero padding

Half padding

3 starting the cross-correlation process



17

Convolution: zero padding

Half padding

4

collecting sub-image

Flattening

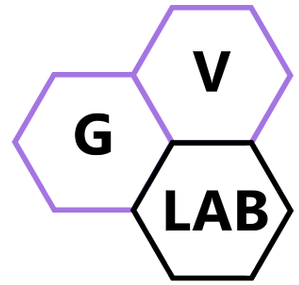
dot-product

Output

18

Convolution: zero padding

Half padding



4

collecting sub-image

Flattening

dot-product

			3	1	0	
	3	1	0	1	1	
	1	1	2	0	2	
	1	2	2	1	0	
	0	1	0	2	1	
					2	

			3	1	0	1	1	2

1	1	0	0	2	1	2	0	1

8	**6**		

Output

19

Convolution: zero padding

Half padding

4

collecting sub-image

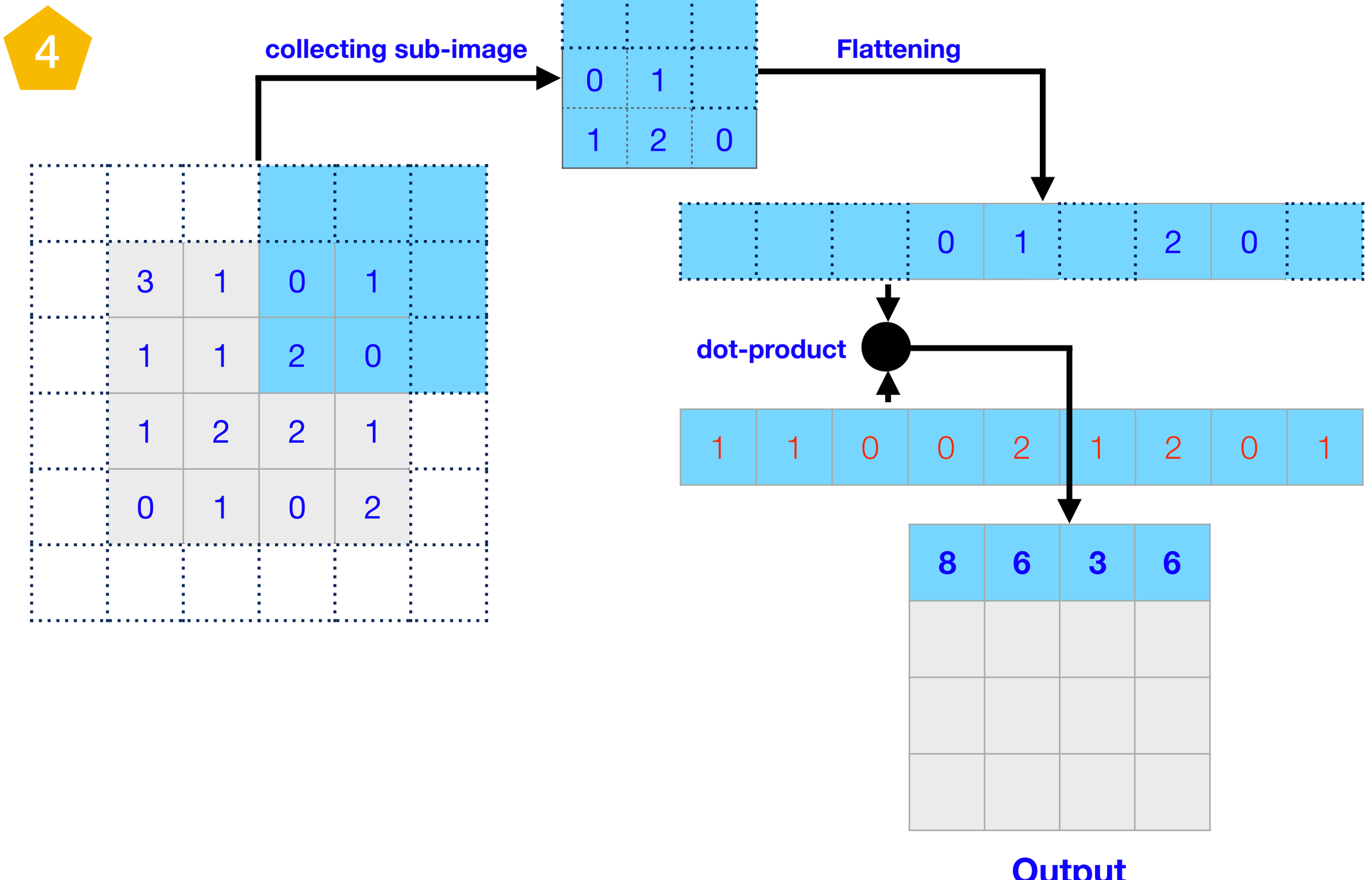
Flattening

dot-product

Output

Convolution: zero padding

Half padding



21

Convolution: zero padding

Half padding

4

collecting sub-image

Flattening

dot-product

	3	1	0	1	
	1	1	2	0	
	1	2	2	1	
	0	1	0	2	

3	1	
1	1	
1	2	

	3	1		1	1		1	2

1	1	0	0	2	1	2	0	1

8	**6**	**3**	**6**
8			

Output

22

Convolution: zero padding

Half padding

4

collecting sub-image

3	1	0
1	1	2
1	2	2

Flattening

3	1	0	1	1	2	1	2	2

dot-product

1	1	0	0	2	1	2	0	1

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

8	6	3	6
8	12		

Output

23

Convolution: zero padding

Half padding

4

collecting sub-image

1	0	1
1	2	0
2	2	1

Flattening

1	0	1	1	2	0	2	2	1

dot-product

1	1	0	0	2	1	2	0	1

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

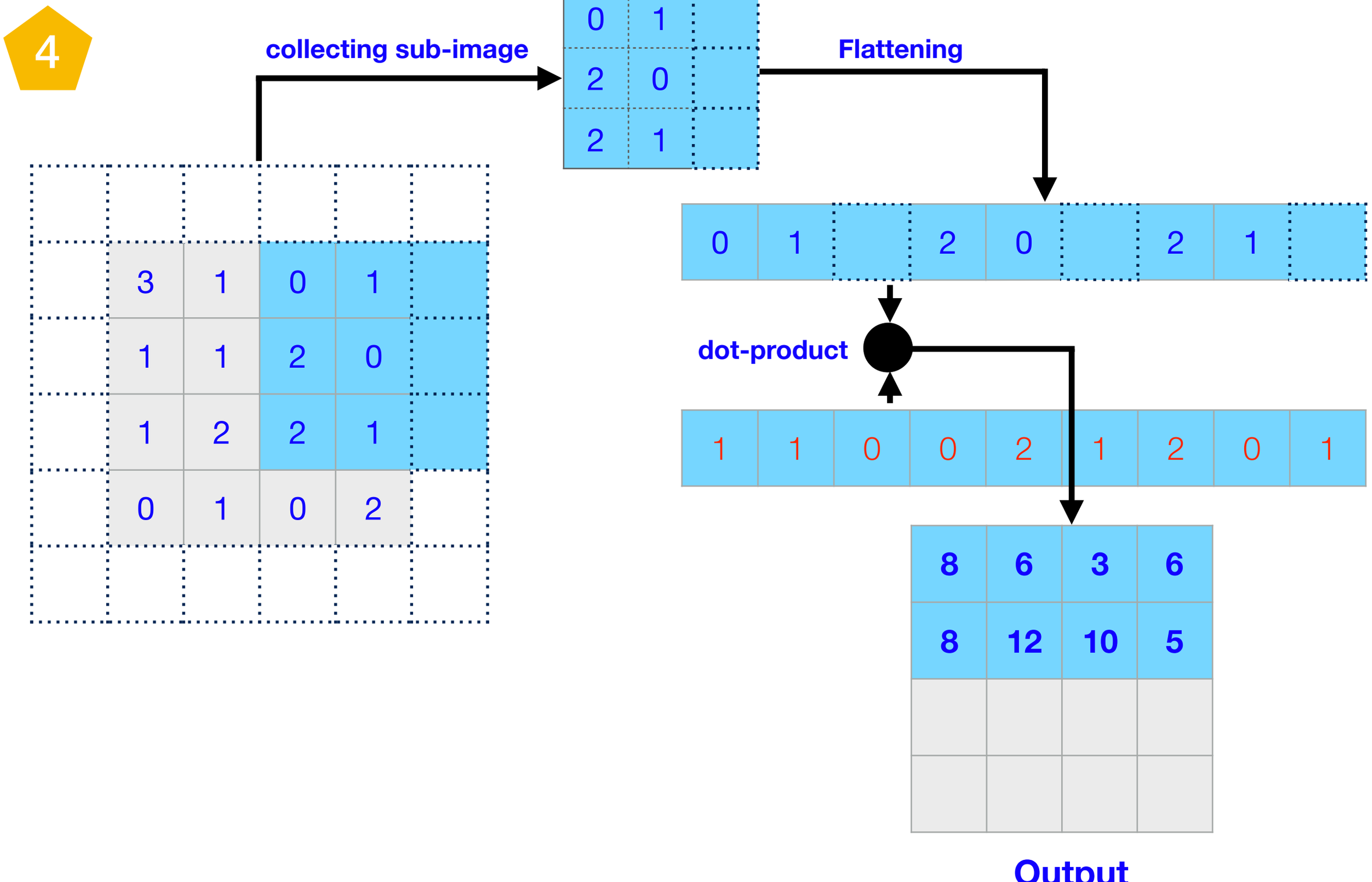
8	**6**	**3**	**6**
8	**12**	**10**	

Output

24

Convolution: zero padding

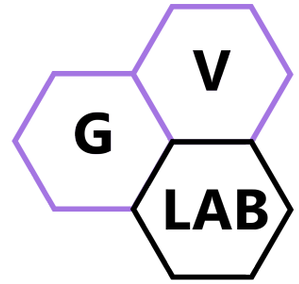
Half padding



25

Convolution: zero padding

Half padding



4

collecting sub-image

Flattening

dot-product

	3	1	0	1	
	1	1	2	0	
	1	2	2	1	
	0	1	0	2	

	1	1
	1	2
	0	1

	1	1		1	2		0	1

1	1	0	0	2	1	2	0	1

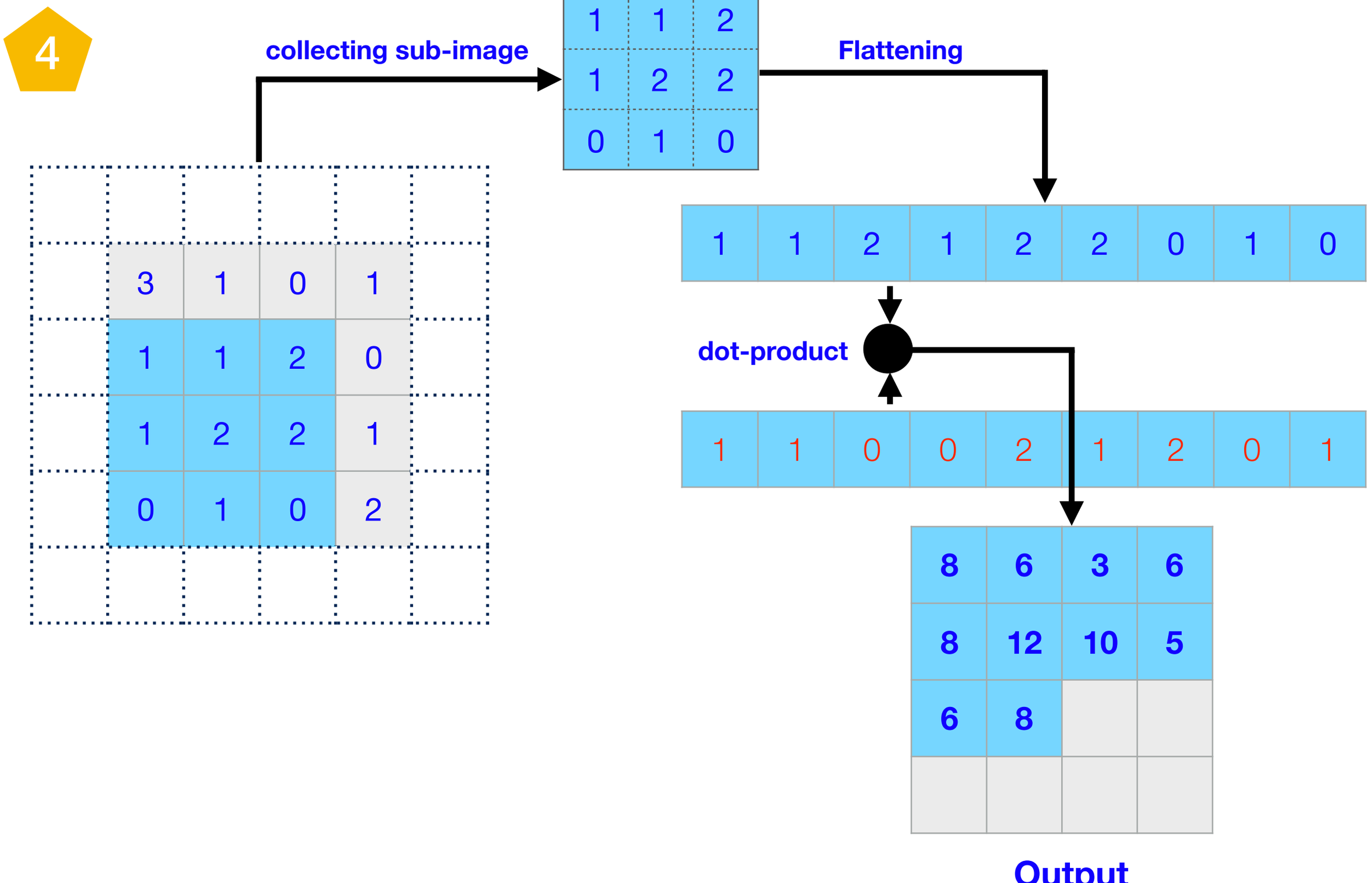
8	6	3	6
8	12	10	5
6			

Output

26

Convolution: zero padding

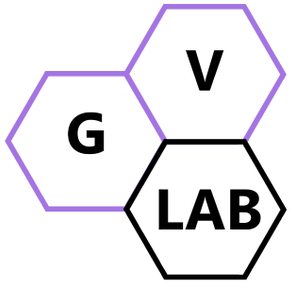
Half padding



27

Convolution: zero padding

Half padding



4

collecting sub-image

Flattening

dot-product

	3	1	0	1	
	1	1	2	0	
	1	2	2	1	
	0	1	0	2	

2	0		2	1		0	2	

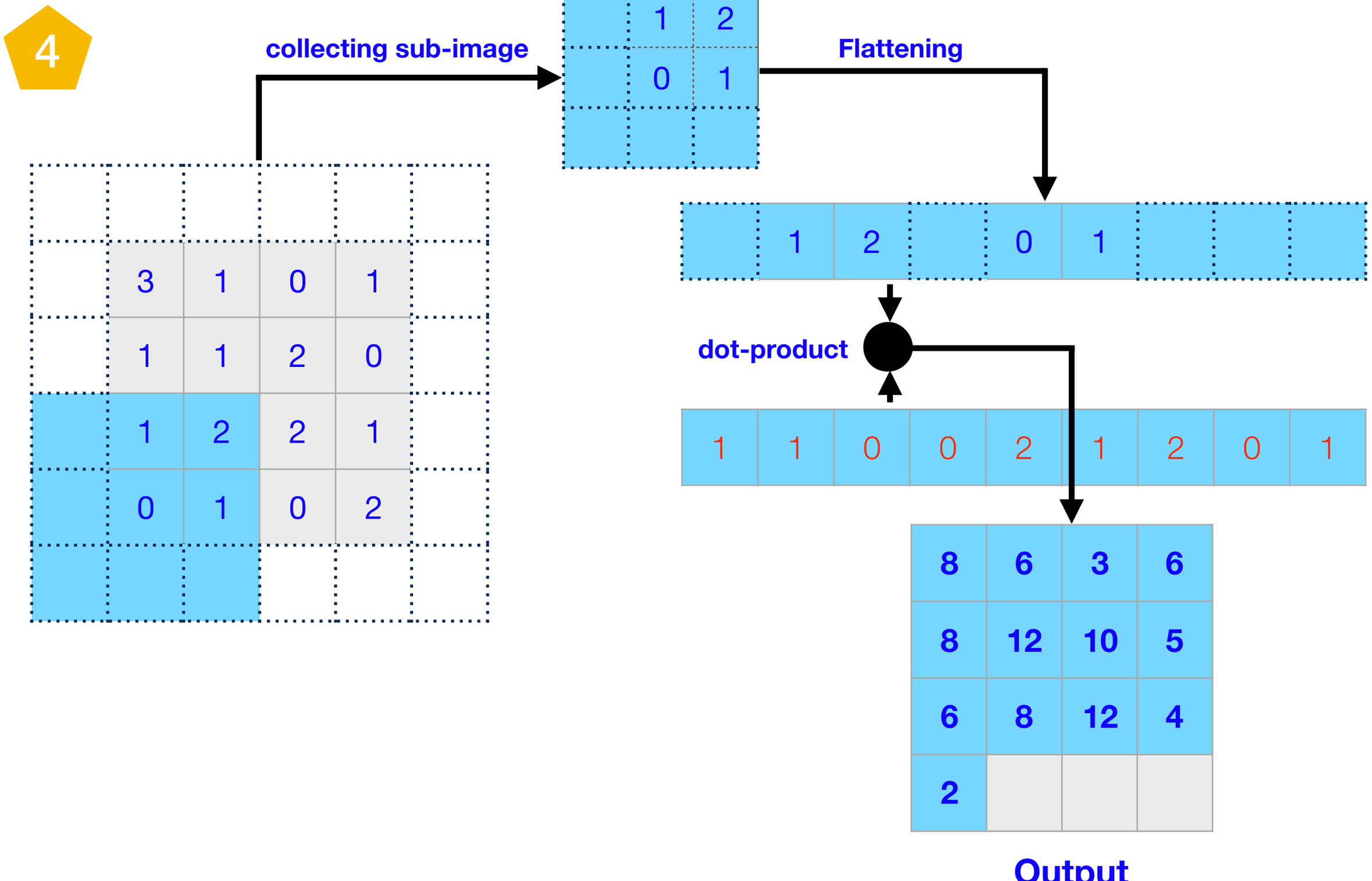
1	1	0	0	2	1	2	0	1

8	6	3	6
8	12	10	5
6	8	12	4

Output

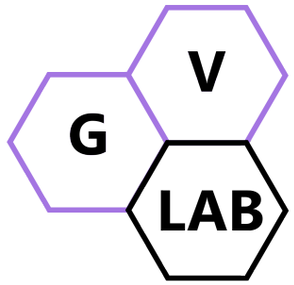
Convolution: zero padding

Half padding



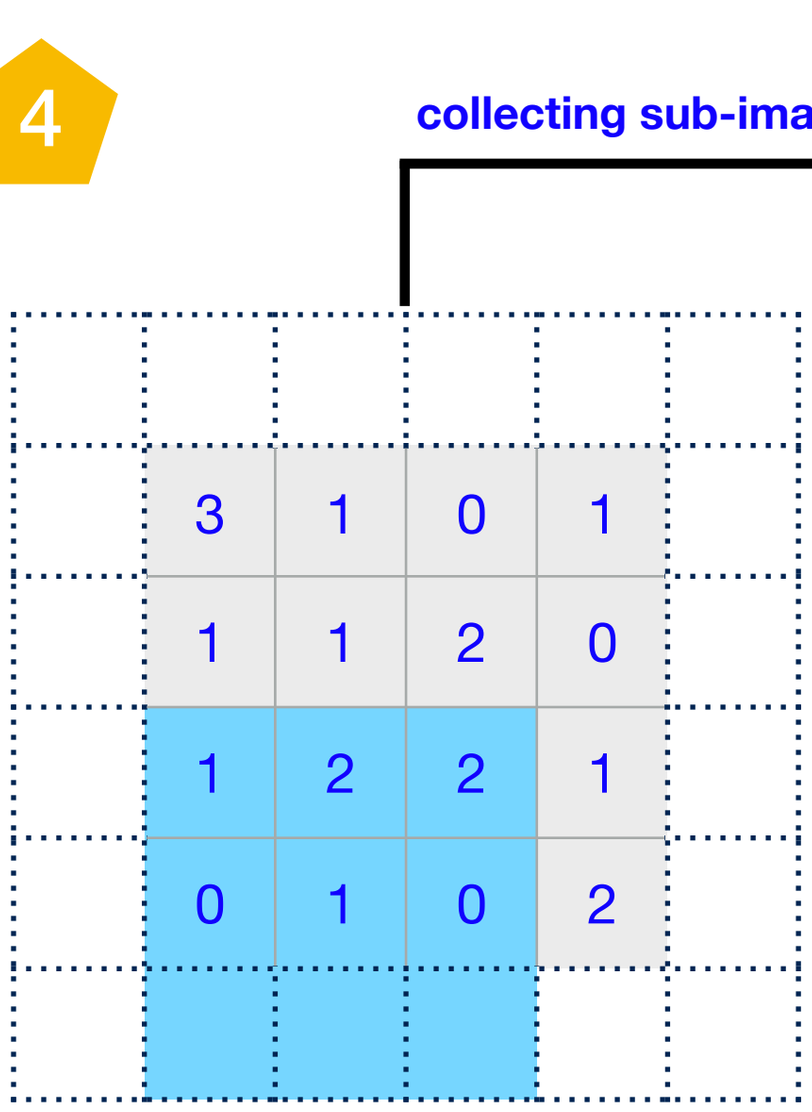
Convolution: zero padding

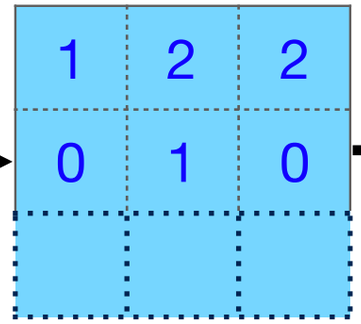
Half padding



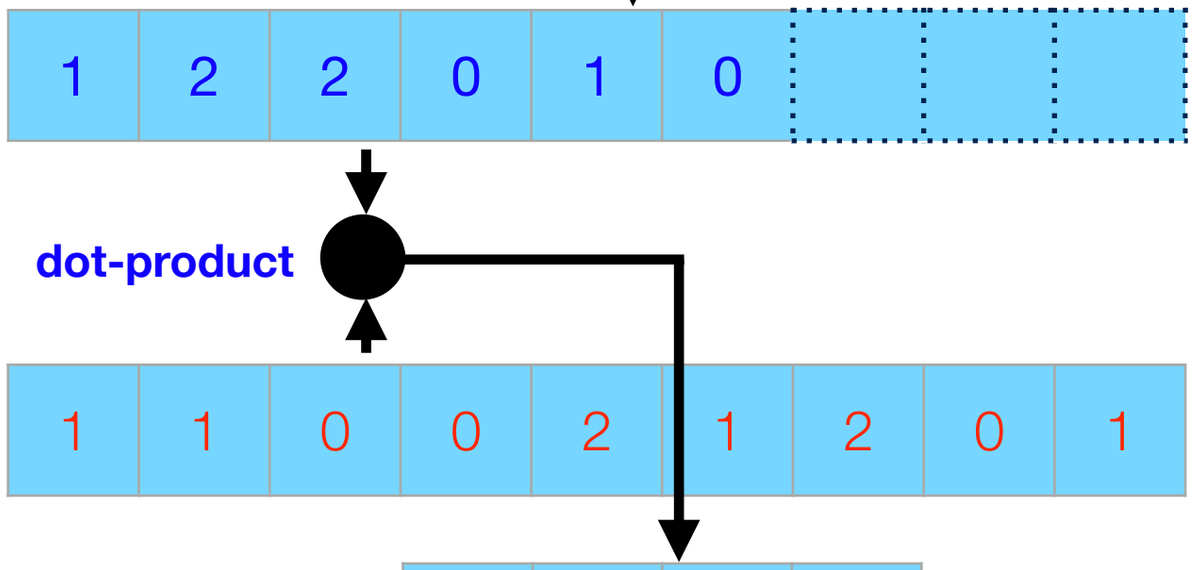
4

collecting sub-image





Flattening



8	6	3	6
8	12	10	5
6	8	12	4
2	5		

Output

30

Convolution: zero padding

Half padding

4

collecting sub-image

2	2	1
0	1	0

Flattening

2	2	1	1	0	2			

dot-product

1	1	0	0	2	1	2	0	1

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

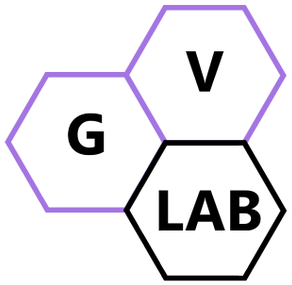
8	6	3	6
8	12	10	5
6	8	12	4
2	5	6	

Output

31

Convolution: zero padding

Half padding



4

collecting sub-image

Flattening

dot-product

1	1	0	0	2	1	2	0	1

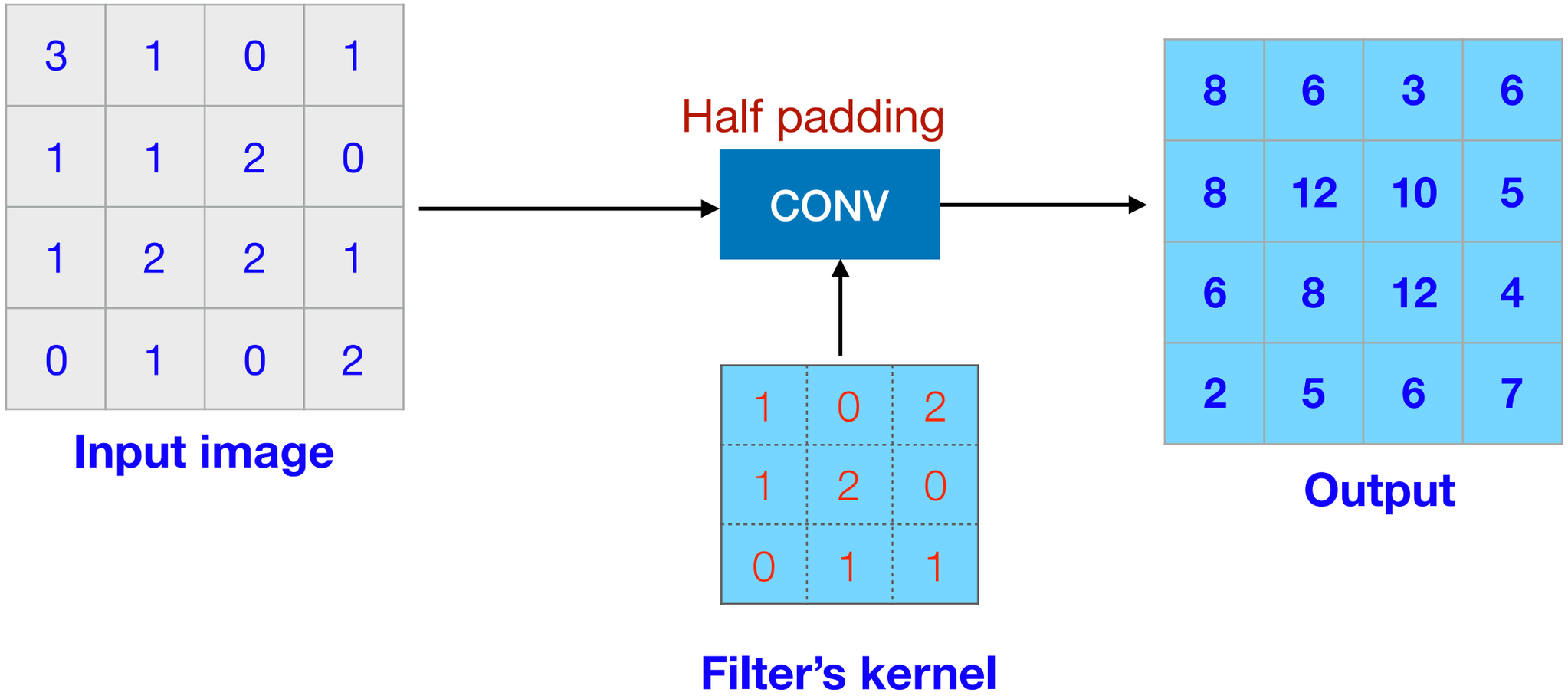
8	6	3	6
8	12	10	5
6	8	12	4
2	5	6	7

Output

Convolution: zero padding

Half padding

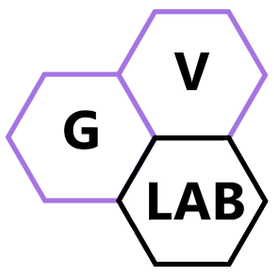
Final result



Convolution: zero padding

Full padding

Dr. Thanh-Sach LE
LTSACH@hcmut.edu.vn

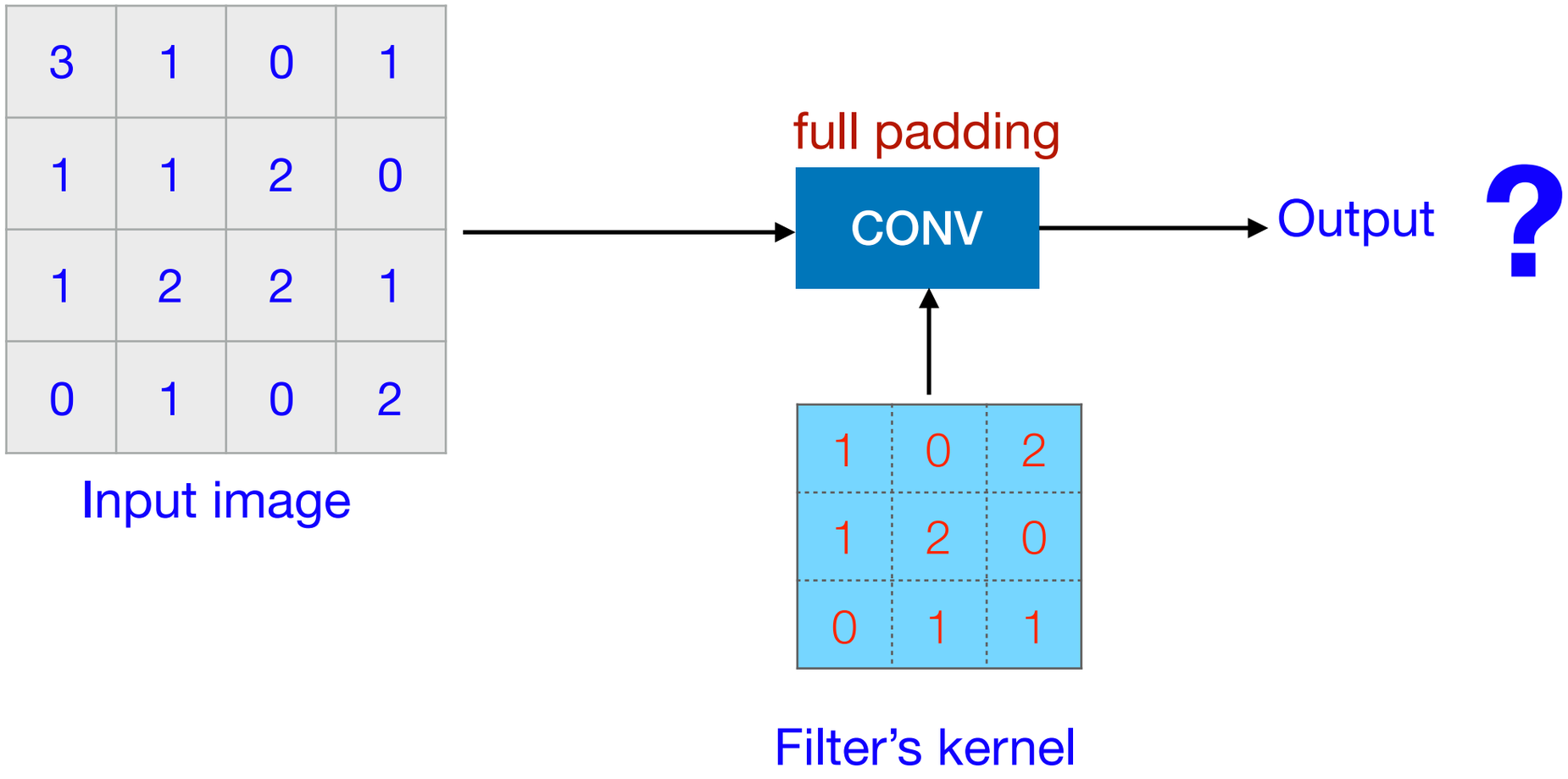


GVLab:
Graphics and Vision Laboratory

Faculty of Computer Science and Engineering,
HCMUT

Convolution: zero padding

Full padding



Convolution: zero padding

Full padding

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

Input image

4×4

full padding

CONV

Output ?

1	0	2
1	2	0
0	1	1

Filter's kernel

3×3

padding size $= (3-1) \times (3-1) = 2 \times 2$

The input is enlarged two pixel on left, right, top, and bottom

Convolution: zero padding

Full padding

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

Input image

4×4

after padding with 2×2

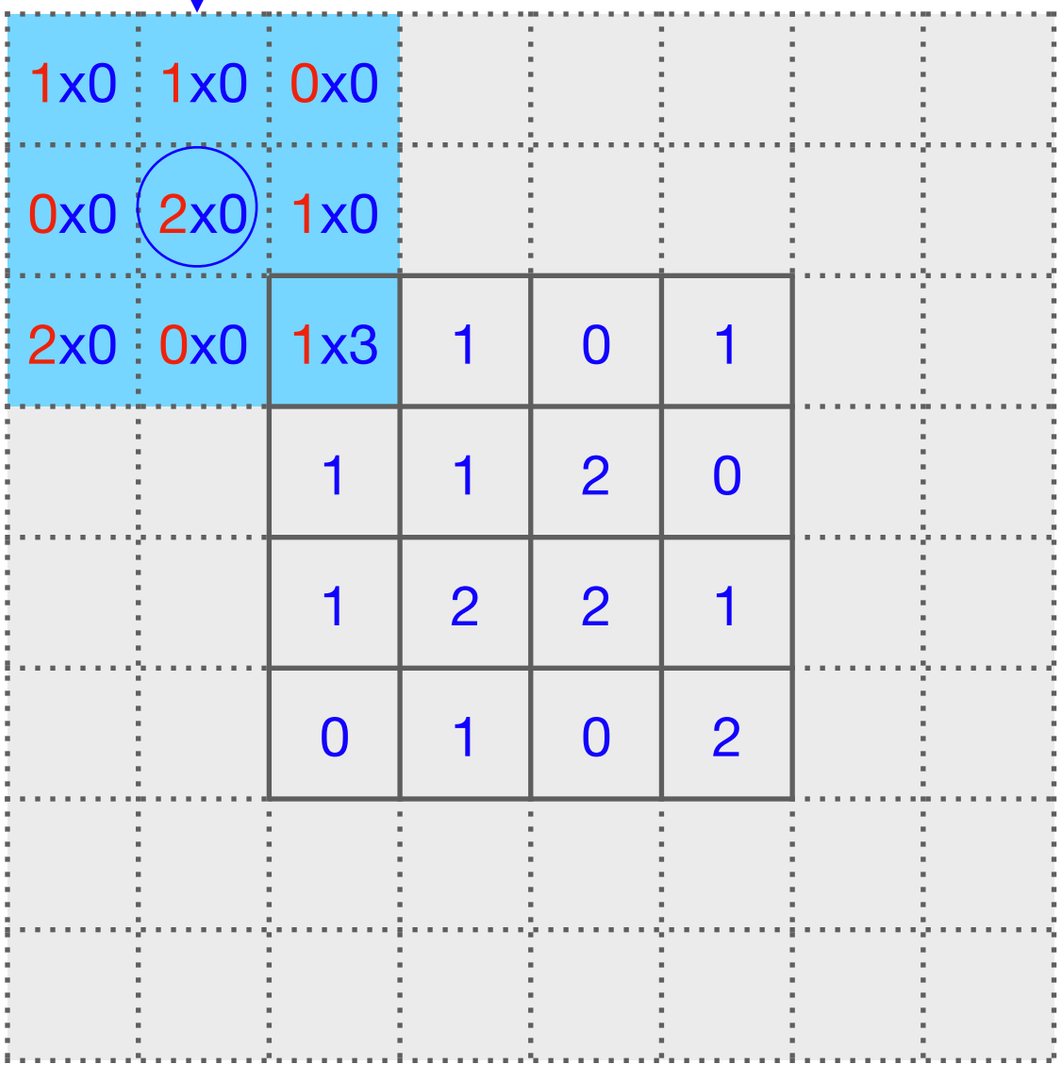
		3	1	0	1		
		1	1	2	0		
		1	2	2	1		
		0	1	0	2		

Output size = ?

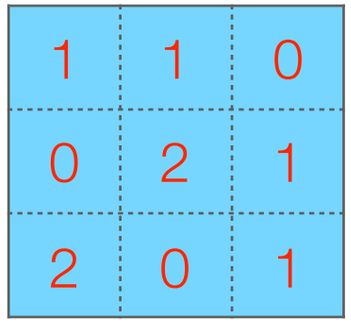
Convolution: zero padding

Full padding

The first valid position



Padded image



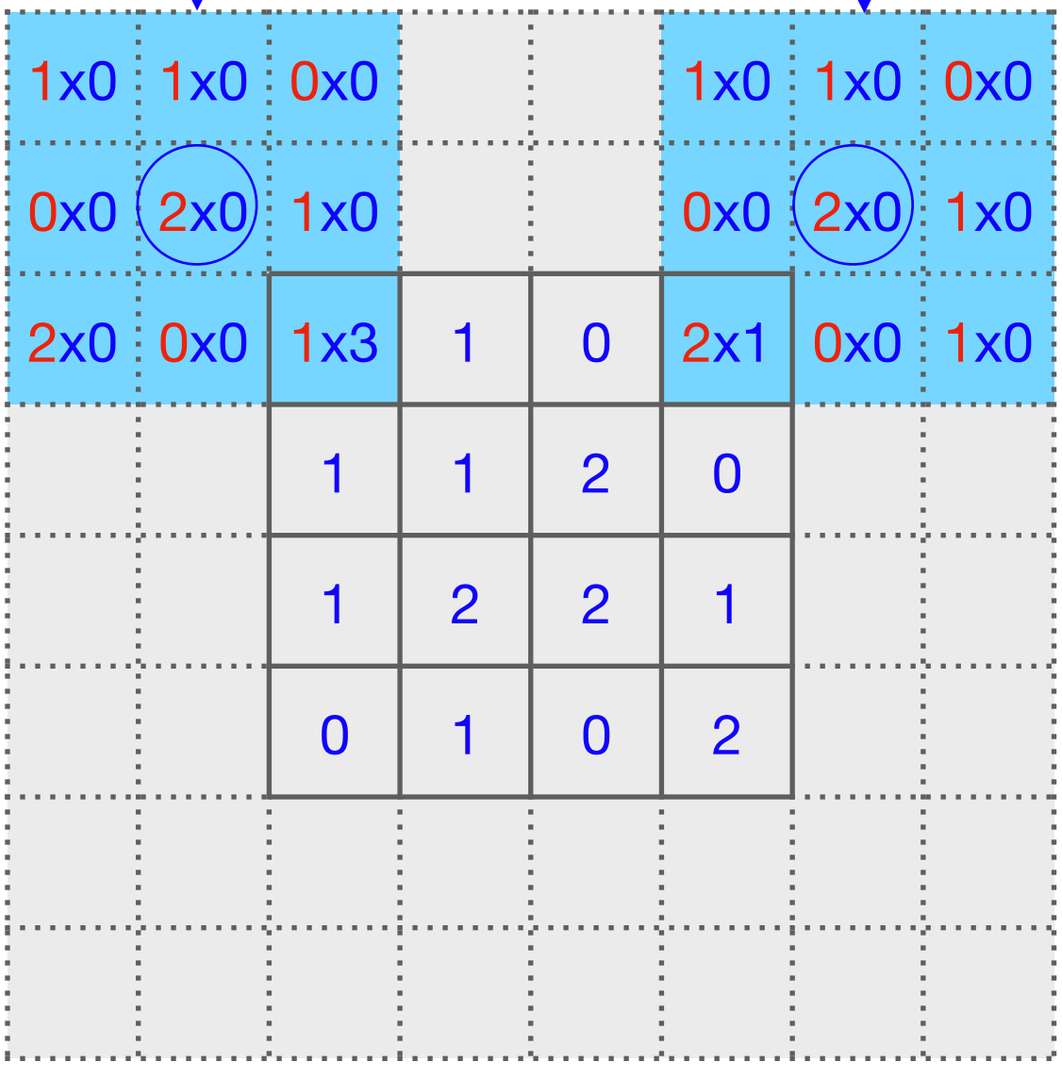
Rotated kernel

Convolution: zero padding

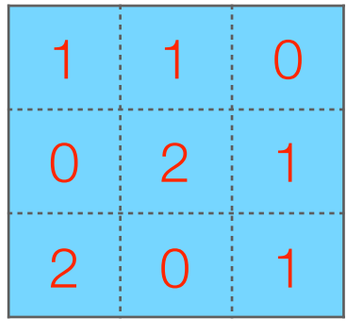
Full padding

The first valid position

The last valid position (on horizontal direction)



Padded image



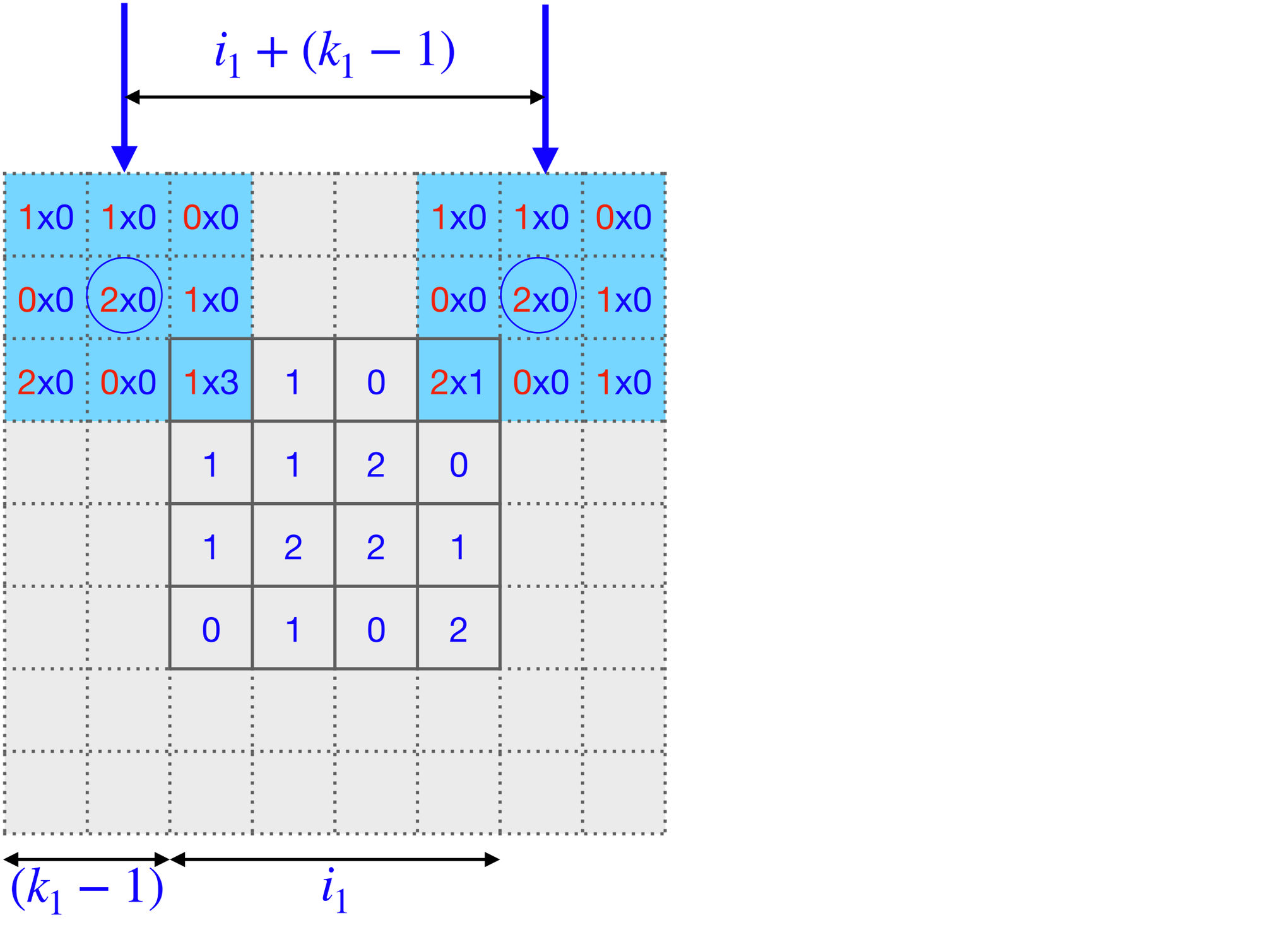
Rotated kernel

Convolution: zero padding

Full padding

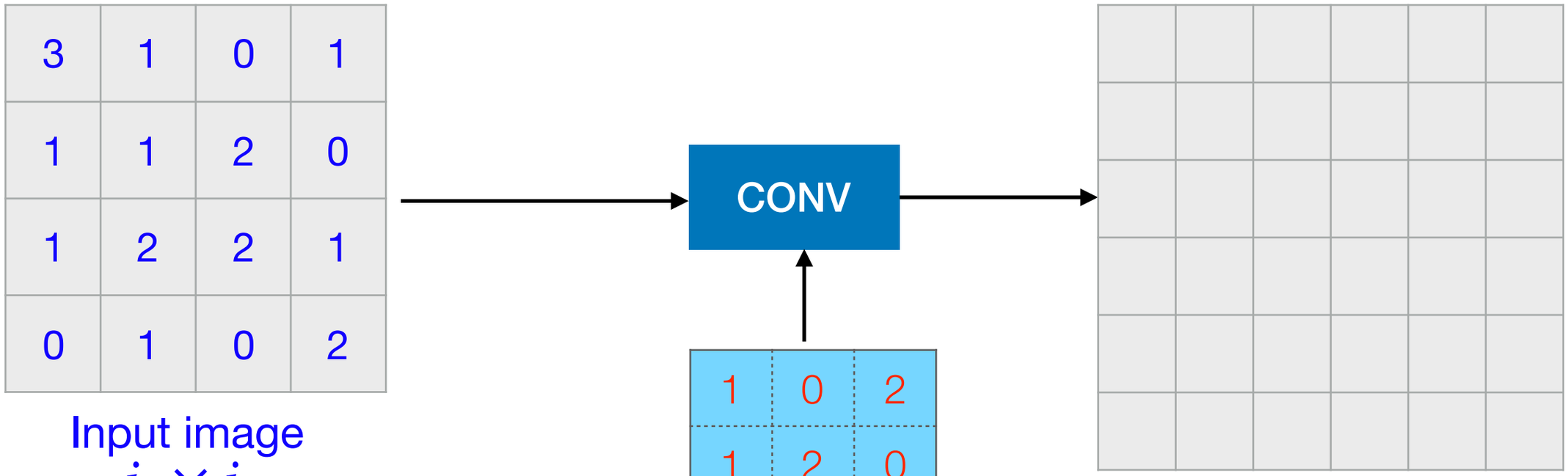
The first valid position

The last valid position (on horizontal direction)



Convolution: zero padding

Full padding



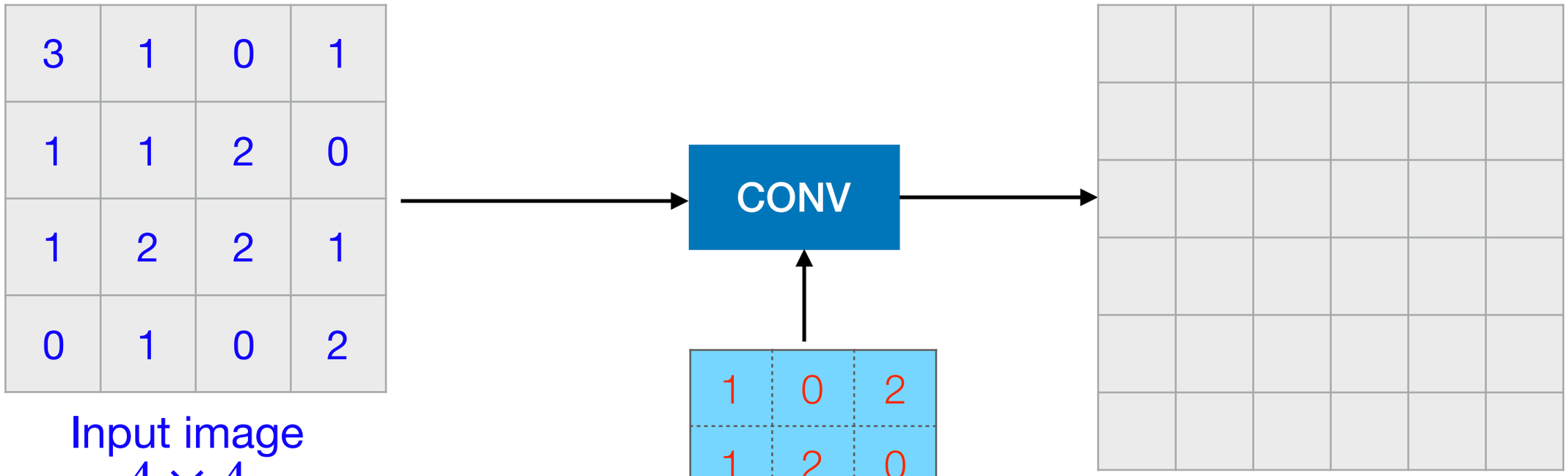
Input image
$i_1 \times i_2$

Filter's kernel
$k_1 \times k_2$

Output
$i_1 + (k_1 - 1) \times i_2 + (k_2 - 1)$

Convolution: zero padding

Full padding



Input image
4×4

Filter's kernel
3×3

Output
6×6

Convolution: zero padding

Full padding

3	1	0	1
1	1	2	0
1	2	2	1
0	1	0	2

Input image
4×4

full-padding

CONV

1	0	2
1	2	0
0	1	1

Filter's kernel
3×3

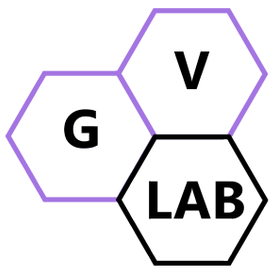
3	1	6	3	0	2
4	8	6	3	6	0
2	8	12	10	5	3
1	6	8	12	4	4
0	2	5	6	7	1
0	0	1	1	2	2

Output
6×6

The computation is the same as the example in half padding

Convolution: non-unit strides

Dr. Thanh-Sach LE
LTSACH@hcmut.edu.vn



GVLab:
Graphics and Vision Laboratory

Faculty of Computer Science and Engineering,
HCMUT

Convolution algorithm

Start

- ✳ Rotate the kernel 180^0
- ✳ Flatten it to a vector

1

- ✳ Padding zero to input
- ✳ Allocate output buffer

2

- ✳ Align the kernel window with the top-left of the input

3

- ✳ Extract the sub-image overlapped with the kernel
- ✳ Flatten the sub-image to a vector
- ✳ Take dot-product with the flattened kernel
- ✳ Fill the result to output

4

all output values assigned?

No

- ✳ Slide the kernel window to next position on the input

5

Yes

End

Default, the kernel moves to right and down with stride of 1-unit.

✳ Stride > 1 can be used to reduce output feature map => reduce the computation in next layers.

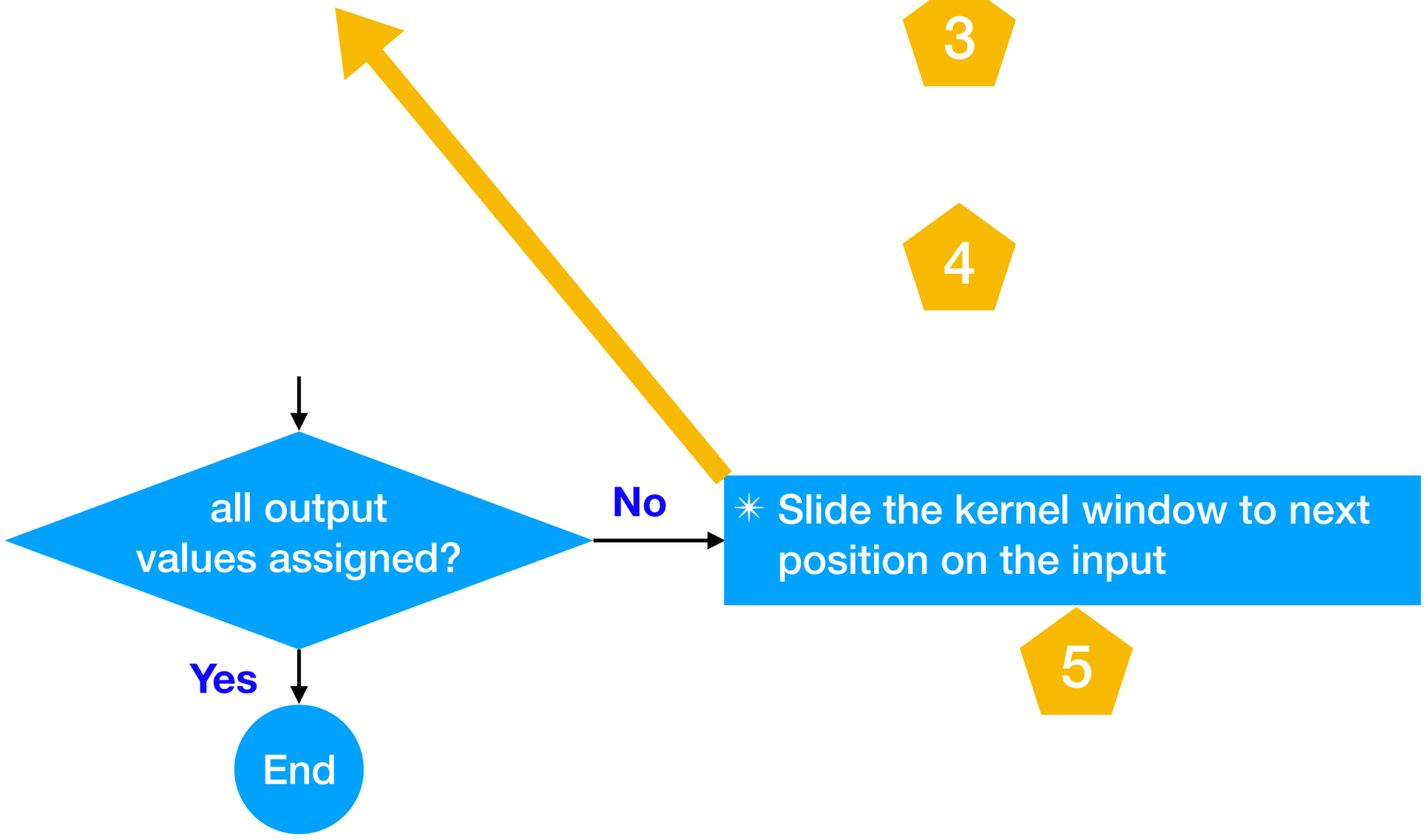
1

2

3

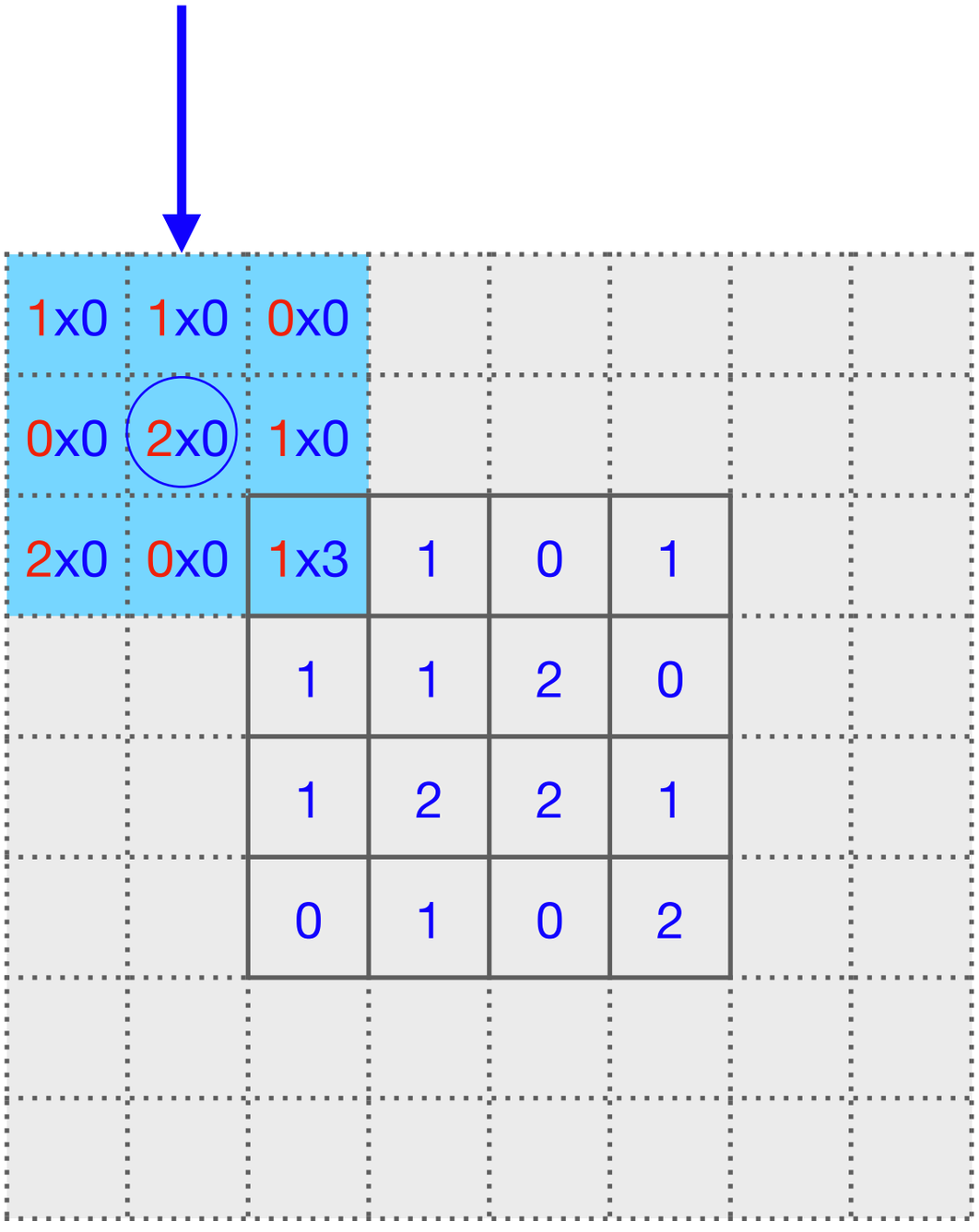
4

5



46 Convolution: non-unit strides

1st valid position



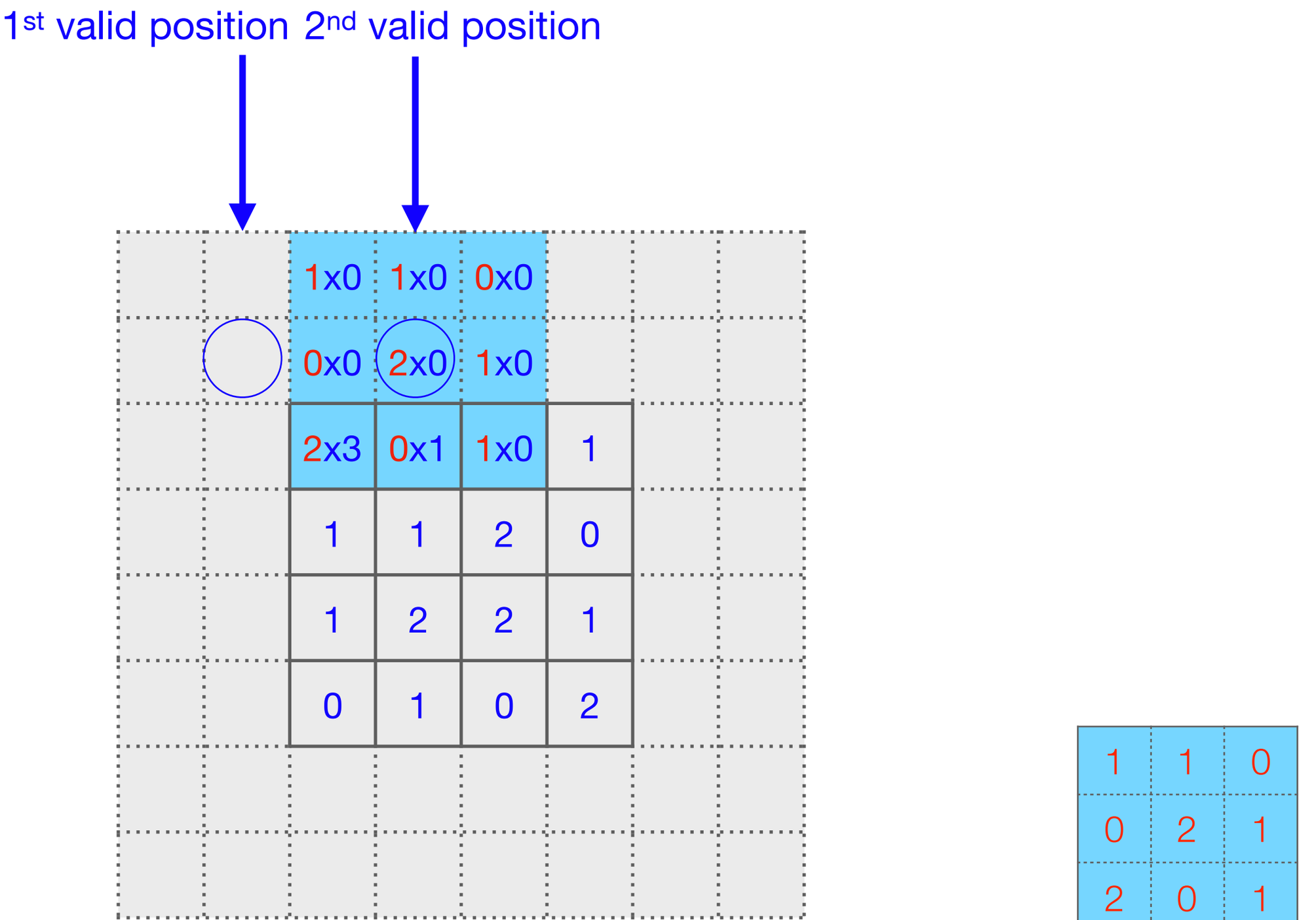
Padded image

stride = 2

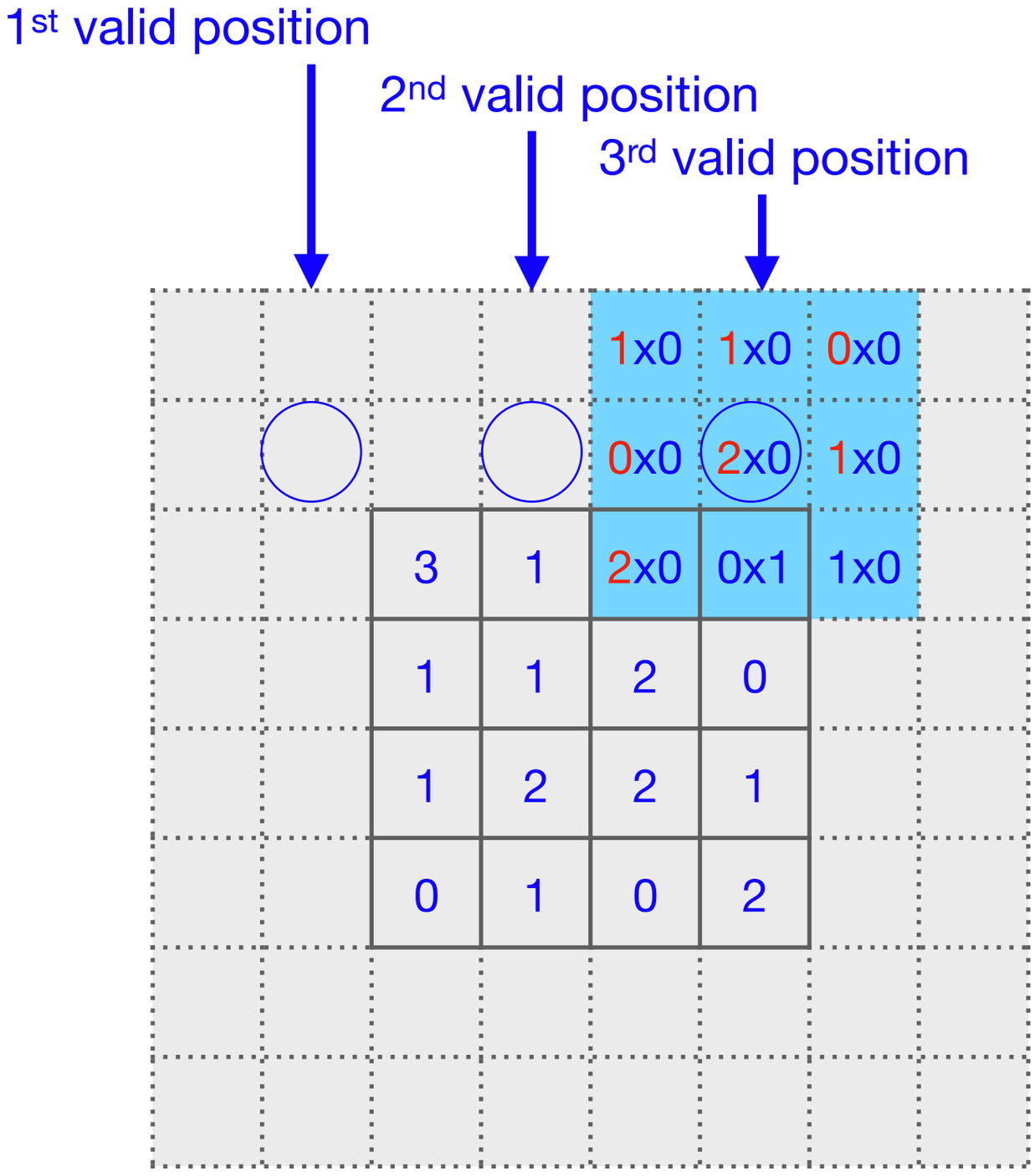
1	1	0
0	2	1
2	0	1

Rotated kernel

47 Convolution: non-unit strides

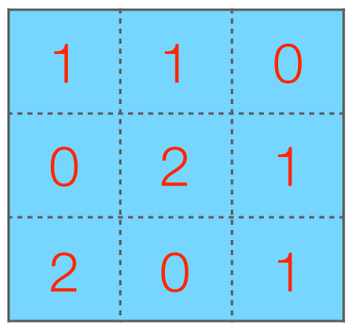


48 Convolution: non-unit strides



stride = 2

CAN NOT move the kernel to right more



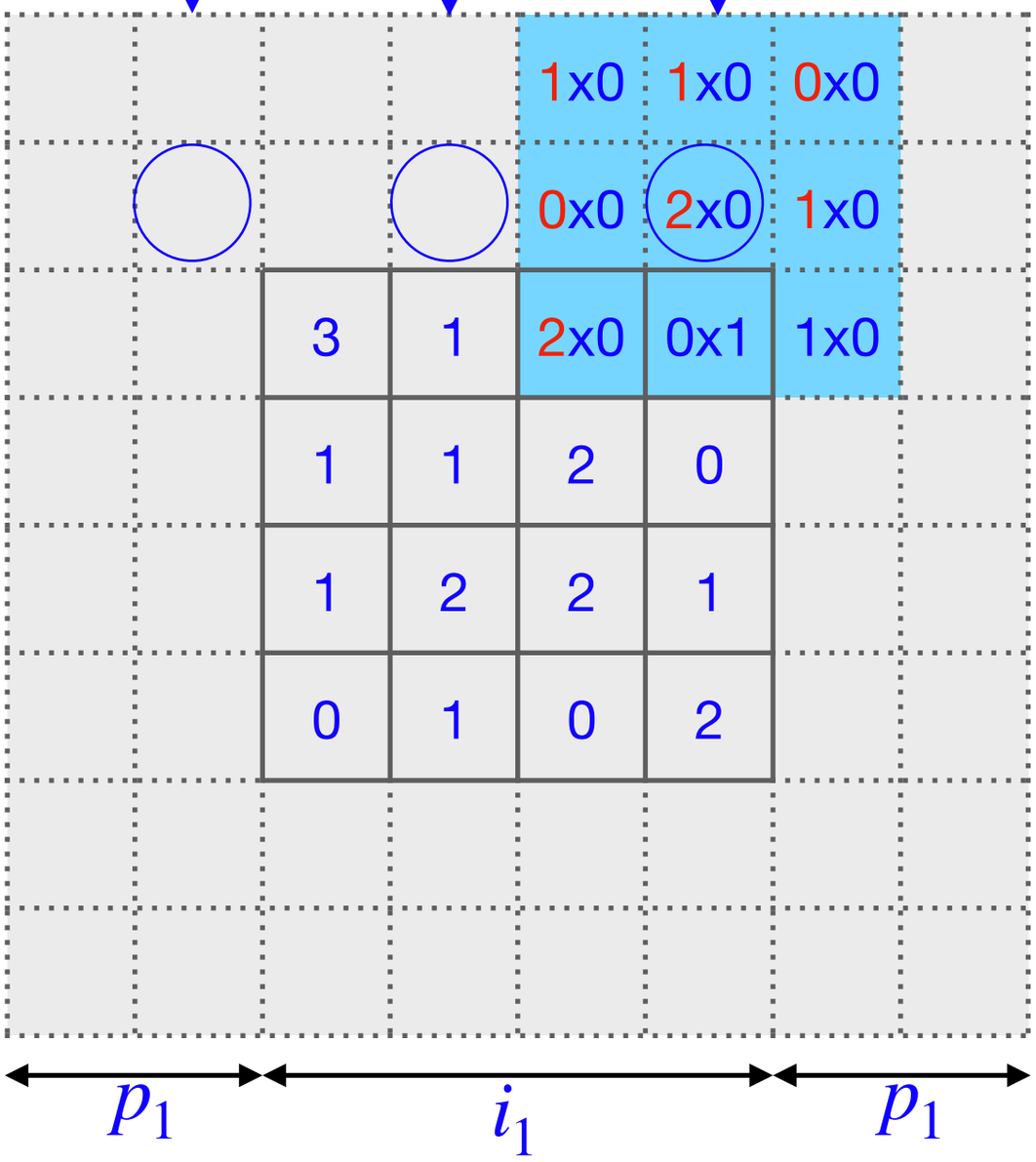
49 Convolution: non-unit strides

stride on x $= s_1$

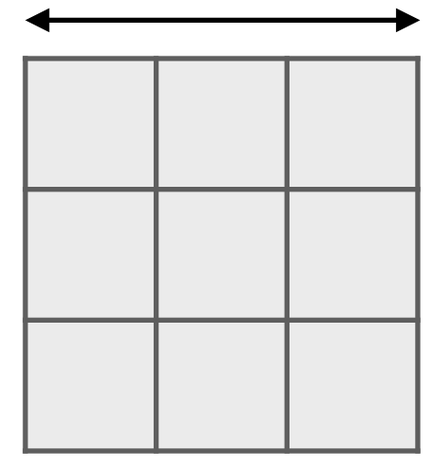
1st valid position

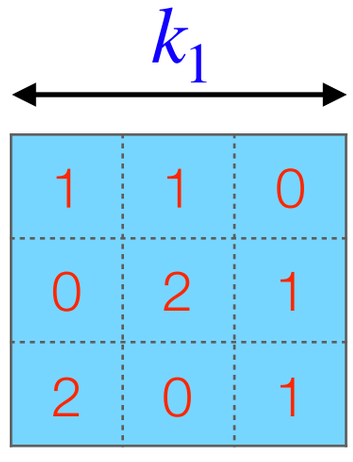
2nd valid position

3rd valid position



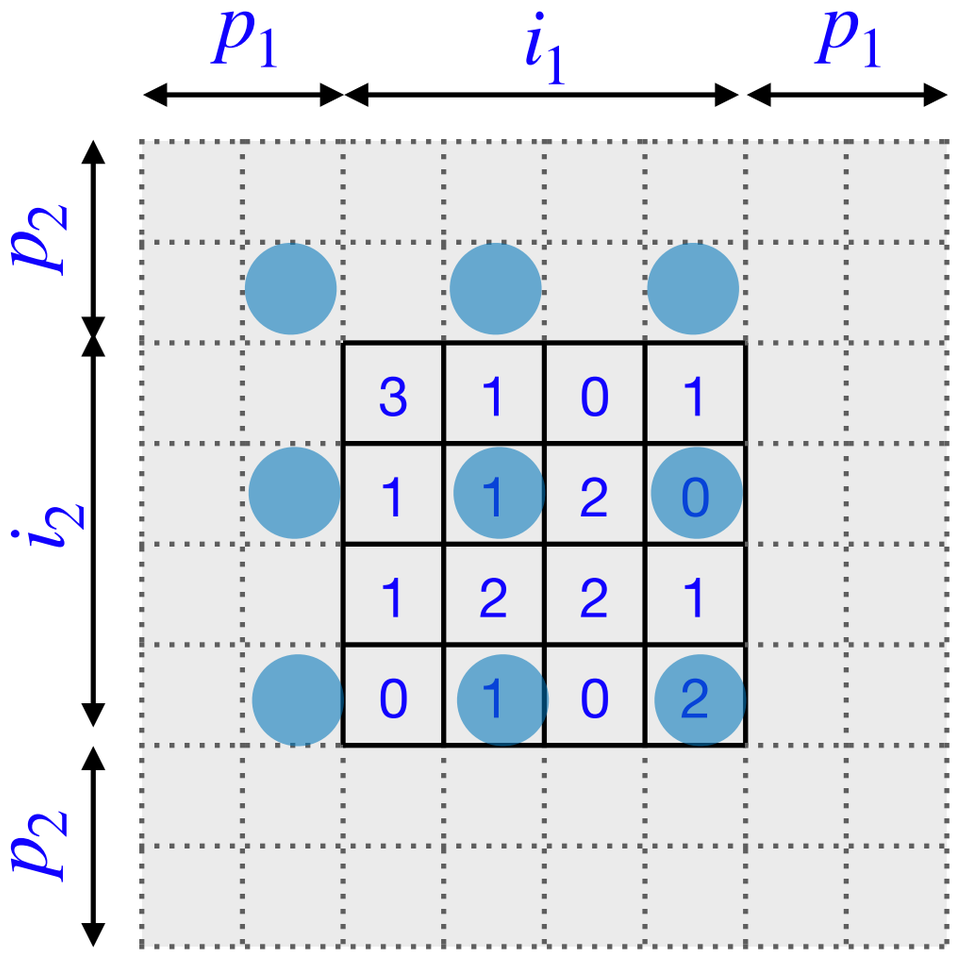
$$\left\lfloor \frac{i_1 + 2p_1 - k_1}{s_1} \right\rfloor + 1$$





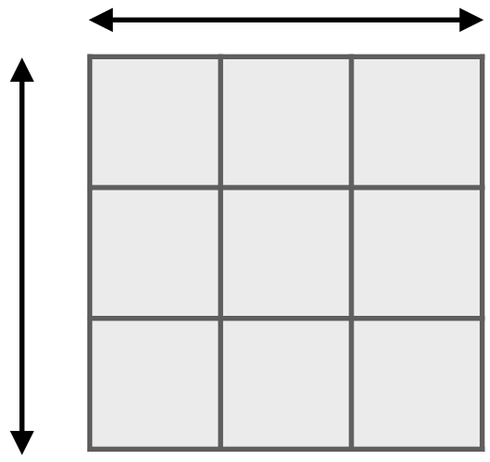
Rotated kernel

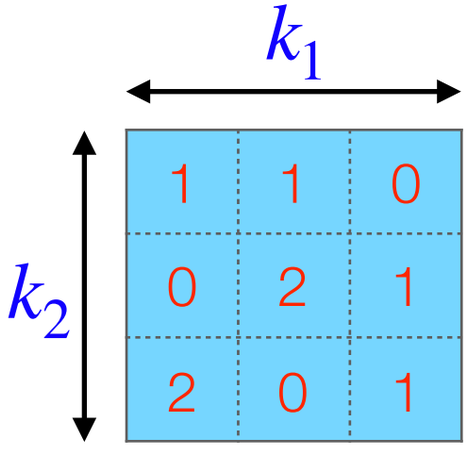
50 Convolution: non-unit strides



$$\left\lfloor \frac{i_1 + 2p_1 - k_1}{s_1} \right\rfloor + 1$$

$$\left\lfloor \frac{i_2 + 2p_2 - k_2}{s_2} \right\rfloor + 1$$



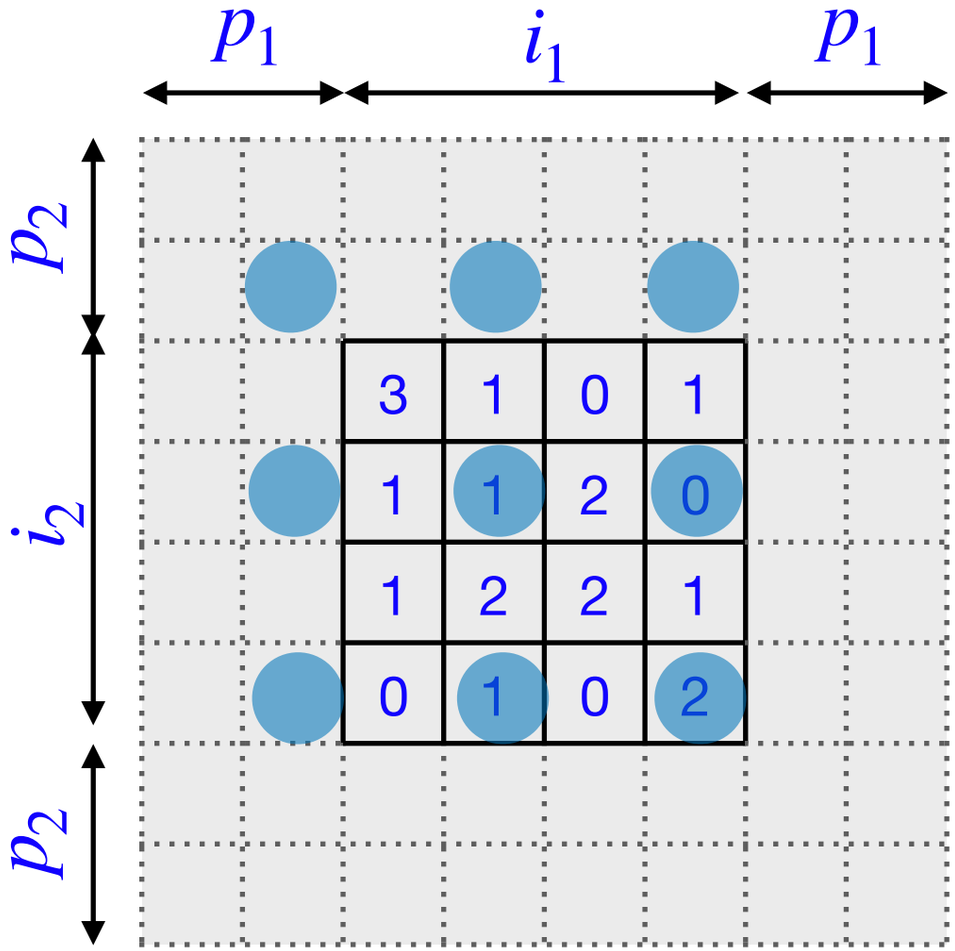


Rotated kernel

stride on x $= s_1$

stride on y $= s_2$

51 Convolution: non-unit strides

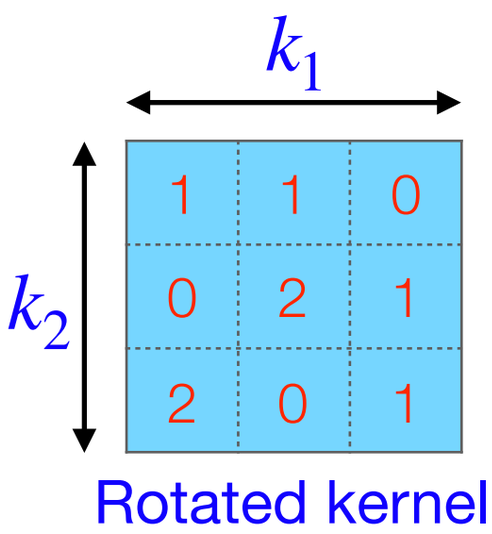


$$i_1 = i_2 = 4$$

$$k_1 = k_2 = 3$$

full padding: $p_1 = p_2 = 2$

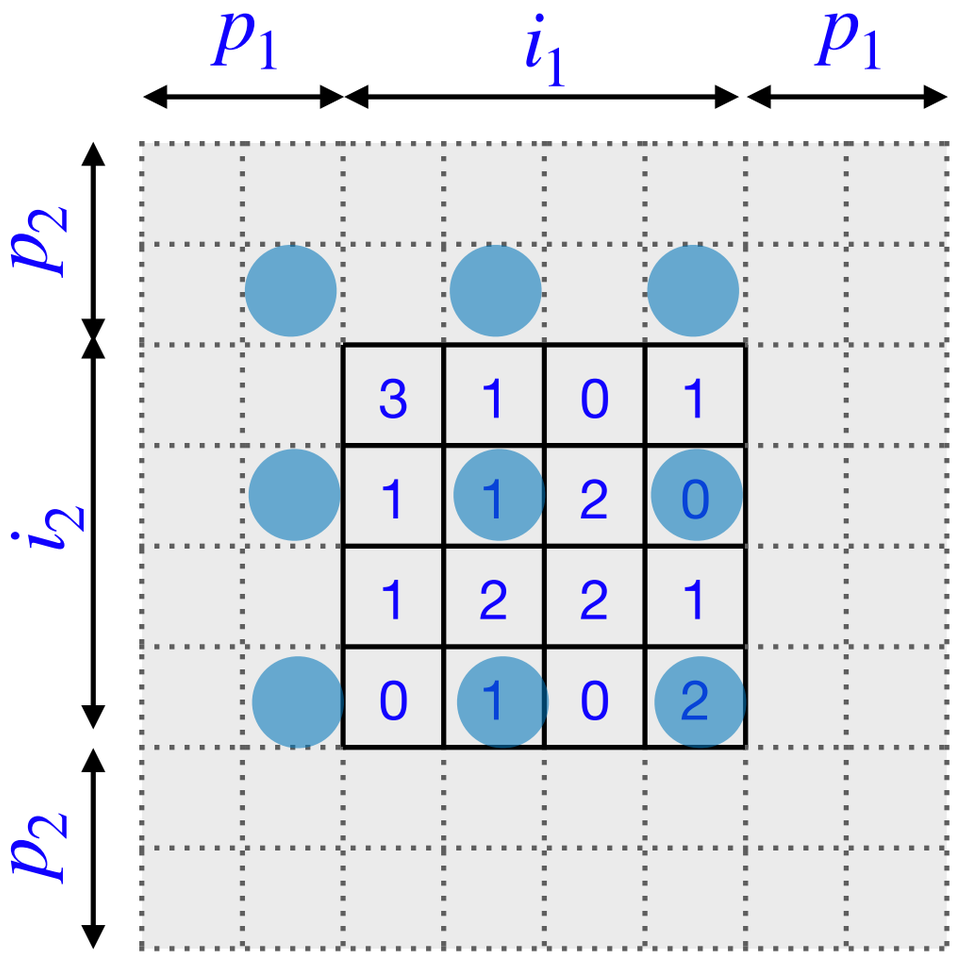
non-unit strides: $s_1 = s_2 = 2$



stride on x $= s_1$

stride on y $= s_2$

52 Convolution: non-unit strides



$$i_1 = i_2 = 4$$

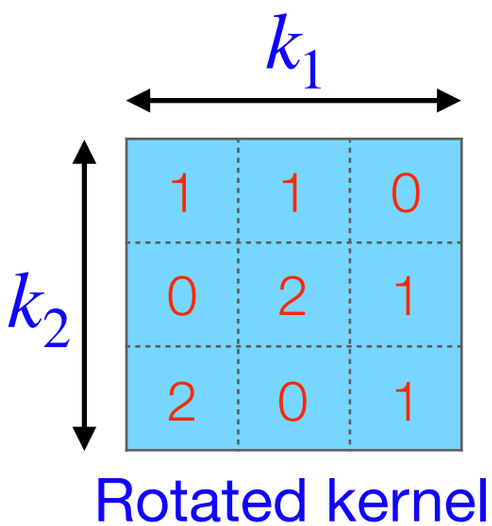
$$k_1 = k_2 = 3$$

full padding: $p_1 = p_2 = 2$

non-unit strides: $s_1 = s_2 = 2$

$$\left\lfloor \frac{i_1 + 2p_1 - k_1}{s_1} \right\rfloor + 1 = 3$$

$$\left\lfloor \frac{i_2 + 2p_2 - k_2}{s_2} \right\rfloor + 1 = 3$$



stride on x $= s_1$

stride on y $= s_2$