EQ: How do we perform 2D convolution?

Mr. Mina
12th Grade Computer Vision
9-12.CT.2, 9-12.CT.5

Lesson 01 - 2D Convolution

Do Now

Answer two of the four questions below with your partner:

1. Give one of the two names for the second operand in convolution.
2. How do we determine the size of our second operand for convolution?
3. What is the very first step of convolution?
4. Can we ever skip flipping the second operand for convolution? If so, describe the scenario.

Lesson 01 - 2D Convolution

Do Now - Answers

Answer two of the four questions below with your partner:

1. Kernel or filter
2. The size of the kernel is determined by the number of elements we want to look at
3. Flipping the kernel is always the first step
4. Yes, we **can** skip flipping the kernel when it's symmetric (i.e. [½, ½] or [⅓, ⅓, ⅓])

2D Convolution

- Same idea as 1D convolution
- Apply a kernel (1D or 2D) onto a 2D array
- **2D Kernels** will be squares with odd size (1x1, 3x3, 5x5, etc.)
  - There will always be a *center* square
  - Place the kernel on top of the pixel you want to calculate
- Make sure to flip the second operand across both axes! For example:

1	2	3
4	5	6
7	8	9

flip across x axis →

3	2	1
6	5	4
9	8	7

flip across y axis →

9	8	7
6	5	4
3	2	1

Computing 2D Convolution

- For simplicity, assume the kernel has already been flipped
- Let's convolve Pixel (1, 1)

Input data

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

$*$

Kernel

a	b	c
d	e	f
g	h	i

$=$

Output

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

Computing 2D Convolution - Pixel (1, 1)

1. Overlay the kernel on top of the desired pixel. In this case, Pixel (1, 1).

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Computing 2D Convolution - Pixel (1, 1)

2. Multiply each value in the input data by the corresponding value in the kernel. Add all of the products together.
   a. a*(0, 0) + b*(0, 1) + c*(0,2) + d*(1, 0) + e*(1, 1) + f*(1, 2) + g*(2, 0) + h*(2, 1) + i*(2, 2)

      = **RESULT #1**

a*(0, 0)	b*(0, 1)	c*(0, 2)	(0, 3)
d*(1, 0)	e*(1, 1)	f*(1, 2)	(1, 3)
g*(2, 0)	h*(2, 1)	i*(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Computing 2D Convolution - Pixel (1, 1)

3. Use the result from Step 2 to set the new value at (1, 1) in the output array

a*(0, 0)	b*(0, 1)	c*(0, 2)	(0, 3)
d*(1, 0)	e*(1, 1)	f*(1, 2)	(1, 3)
g*(2, 0)	h*(2, 1)	i*(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Step 2

Output

?	?	?	?
?	**RESULT #1**	?	?
?	?	?	?
?	?	?	?

Computing 2D Convolution

- Rinse and repeat for the remaining pixels

Input data

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

*

Kernel

a	b	c
d	e	f
g	h	i

=

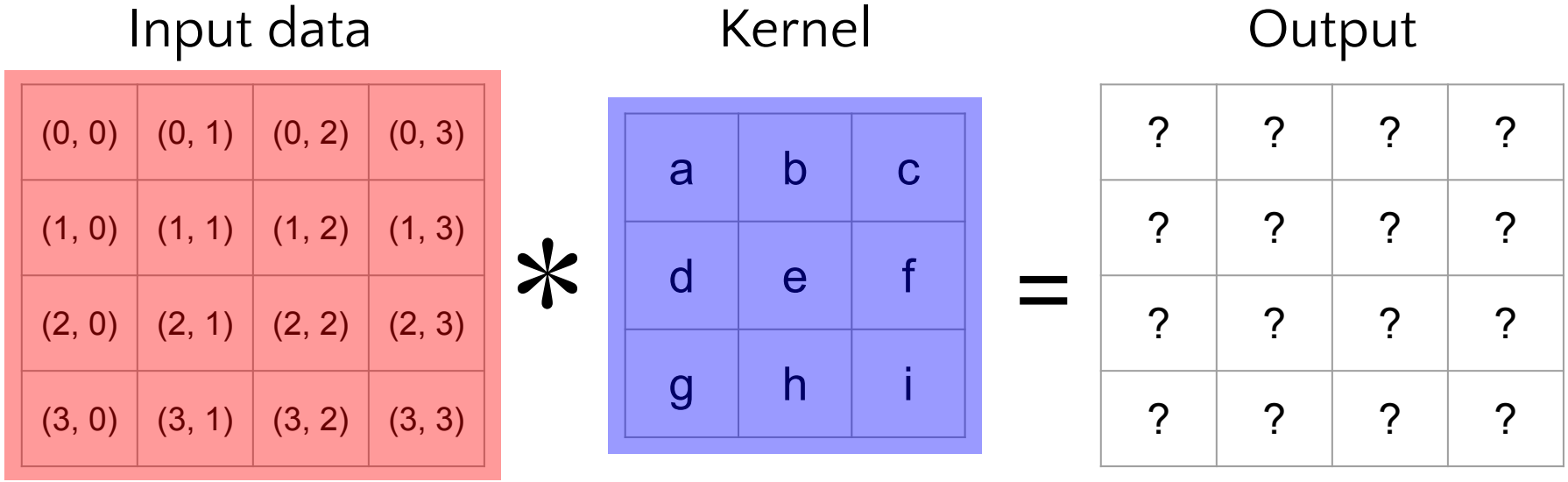
Output

?	?	?	?
?	**RESULT #1**	?	?
?	?	?	?
?	?	?	?

Roll for confidence!

Computing 2D Convolution

- Let's convolve Pixel (1, 2)



Computing 2D Convolution - Pixel (1, 2)

1. Overlay the kernel on top of the desired pixel. In this case, Pixel (1, 2).

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Computing 2D Convolution - Pixel (1, 2)

2. Multiply each value in the input data by the corresponding value in the kernel. Add all of the products together.
   a. a*(0, 1) + b*(0, 2) + c*(0, 3) + d*(1, 1) + e*(1, 2) + f*(1, 3) + g*(2, 1) + h*(2, 2) + i*(2, 3)

      = **RESULT #2**

(0, 0)	a*(0, 1)	b*(0, 2)	c*(0, 3)
(1, 0)	d*(1, 1)	e*(1, 2)	f*(1, 3)
(2, 0)	g*(2, 1)	h*(2, 2)	i*(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Computing 2D Convolution - Pixel (1, 2)

3. Use the result from Step 2 to set the new value at (1, 2) in the output array

(0, 0)	a*(0, 1)	b*(0, 2)	c*(0, 3)
(1, 0)	d*(1, 1)	e*(1, 2)	f*(1, 3)
(2, 0)	g*(2, 1)	h*(2, 2)	i*(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Step 2

Output

?	?	?	?
?	**RESULT #1**	**RESULT #2**	?
?	?	?	?
?	?	?	?

Practice: Computing 2D Convolution

- With a partner, use the numbers below to find what RESULT #3 and RESULT #4
  - Do we *need* to flip the kernel here? 🤔

Input data

4	6	3	7
4	8	2	6
8	3	6	10
1	6	7	6

*

Kernel

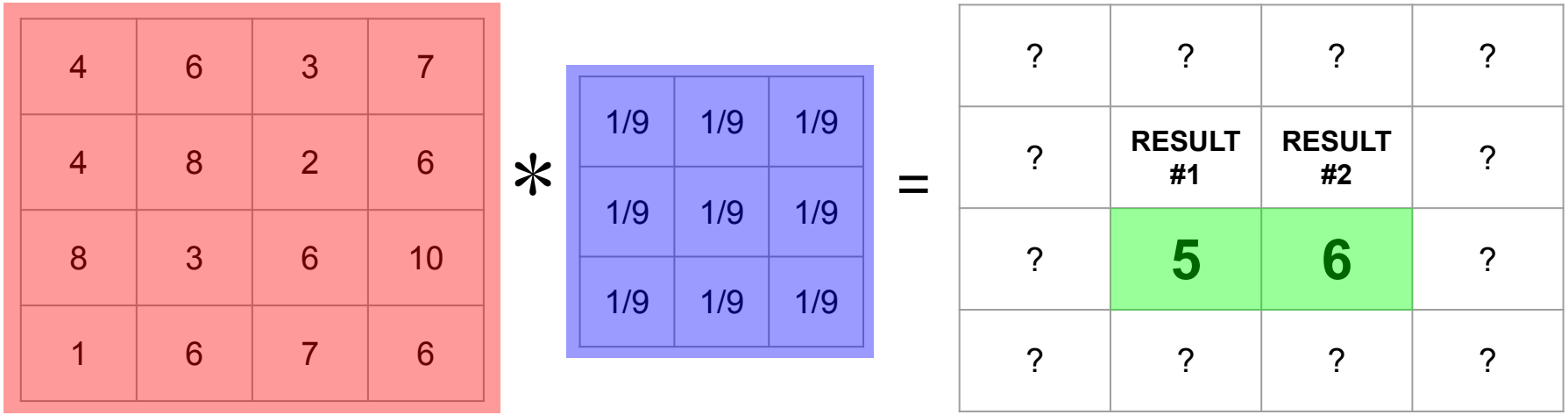
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

=

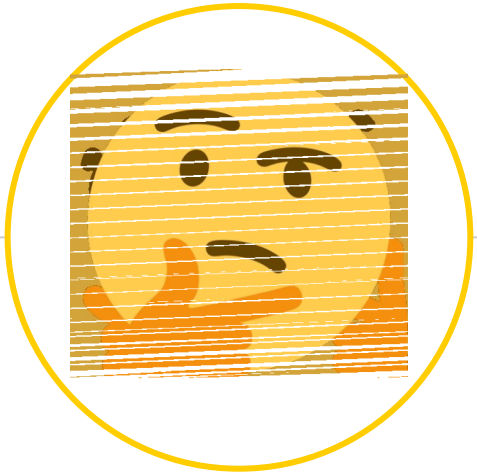
Output

?	?	?	?
?	**RESULT #1**	**RESULT #2**	?
?	**RESULT #3**	**RESULT #4**	?
?	?	?	?

Answer: Computing 2D Convolution



Roll for confidence!



How do we convolve near borders?

Convolution near borders

(Hi, Grace Hopper!)

There are four different approaches when it comes to convolution near borders:

1. **Zero pad** - values are set to 0
2. **Circular** - values are copied from the opposite end of the list
3. **Replicate** - values are duplicates of the last row/column
4. **Symmetric** - the image is reflected and values are taken as their mirror image