Filter

Name	Dharini Baskaran
Identity Key	dhba5060

	Level	Completed
○	Beginner	11
□	Intermediate	5
◇	Advanced	1
◇◇	Expert	0

Goal	
4722	13
5722	15
Total Completed	
17	

Filter

CSCI 5722/4722: Computer Vision

Spring 2024

Dr. Tom Yeh

Dr. Mehdi Moghari

1D Filters

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

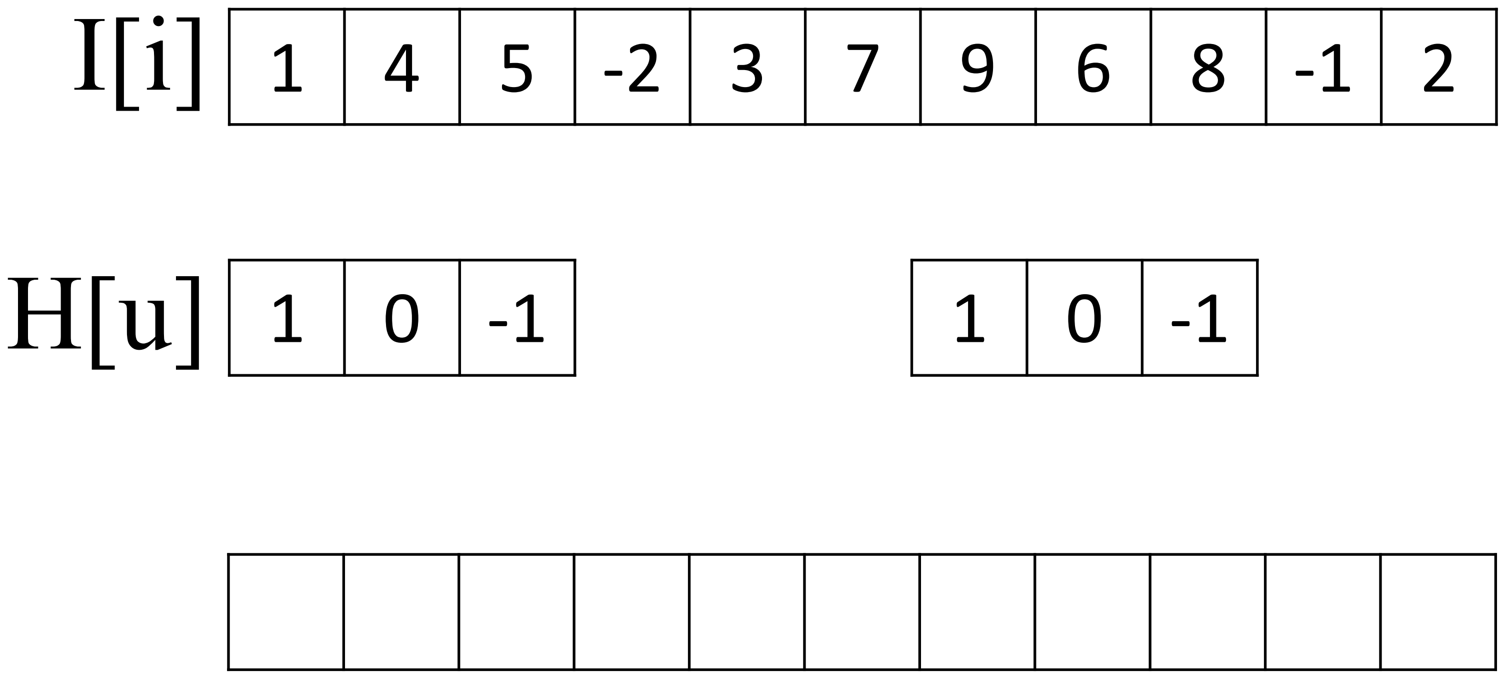
1D Signal

I[i]	1	4	5	-2	3	7	9	6	8	-1	2

Element-wise Operation

I[i]	1	4	5	-2	3	7	9	6	8	-1	2
*	2										
							18	12	16	-2	4

Cross Correlation



I[i]	1	4	5	-2	3	7	9	6	8	-1	2

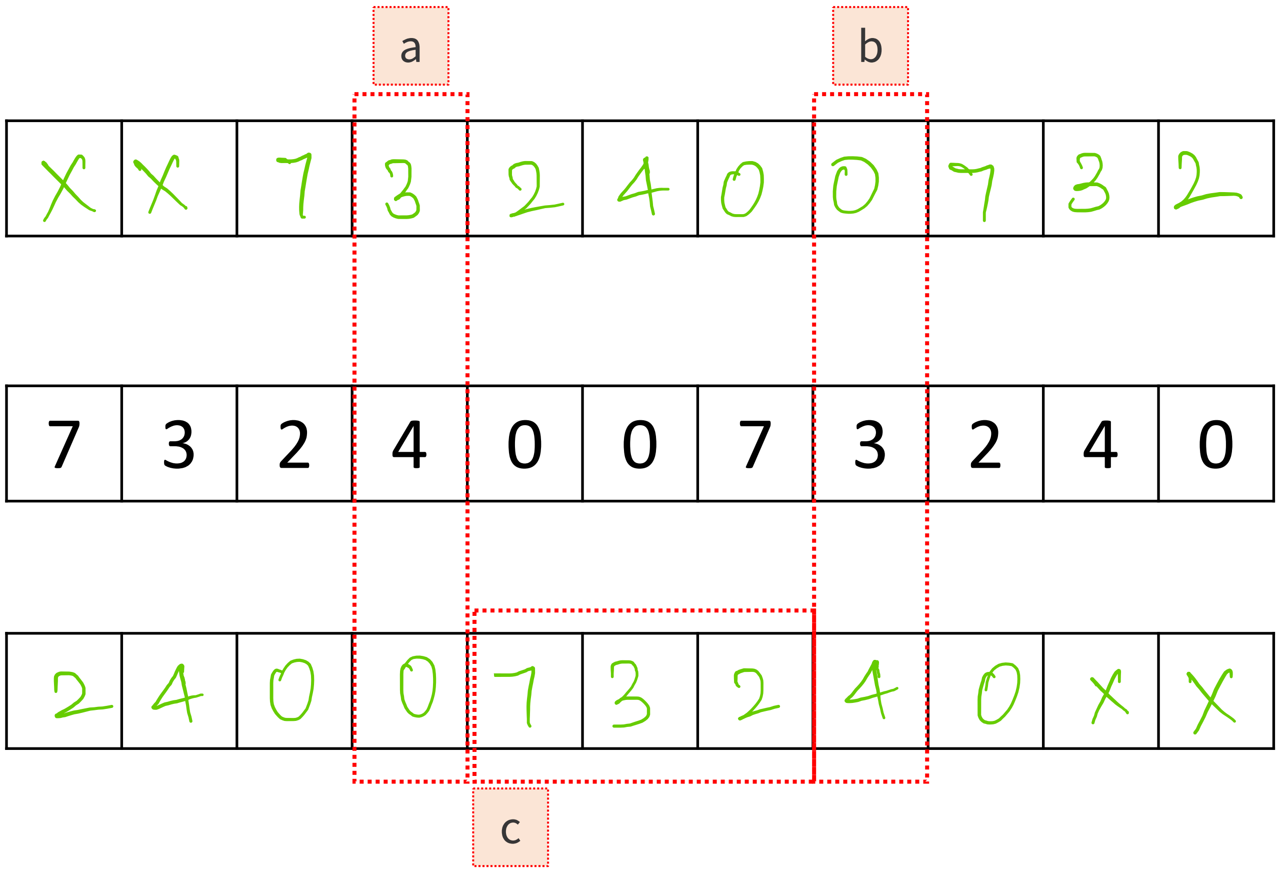
H[u]	1	0	-1

1	0	-1

Neighborhood

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

Neighborhood



Neighbor											
2 to the left	X	X	7	3	2	4	0	0	7	3	2
	7	3	2	4	0	0	7	3	2	4	0
2 to the right	2	4	0	0	7	3	2	4	0	X	X

🔑 Σa = 7; Σb = 7; Σc = 12

Cross Correlation as Matrix Multiplication

Neighbor

L1							
C	1	4	5	2	3	7	9
R1							

Box filter

pad with -1

Neighbor

L1	-1	2	1	0	2	3	1
C	2	1	0	2	3	1	0
R2	0	2	3	1	0	-1	-1
	~~0~~	~~3~~	4	3	5	3	0
	1	5	b		c		

a

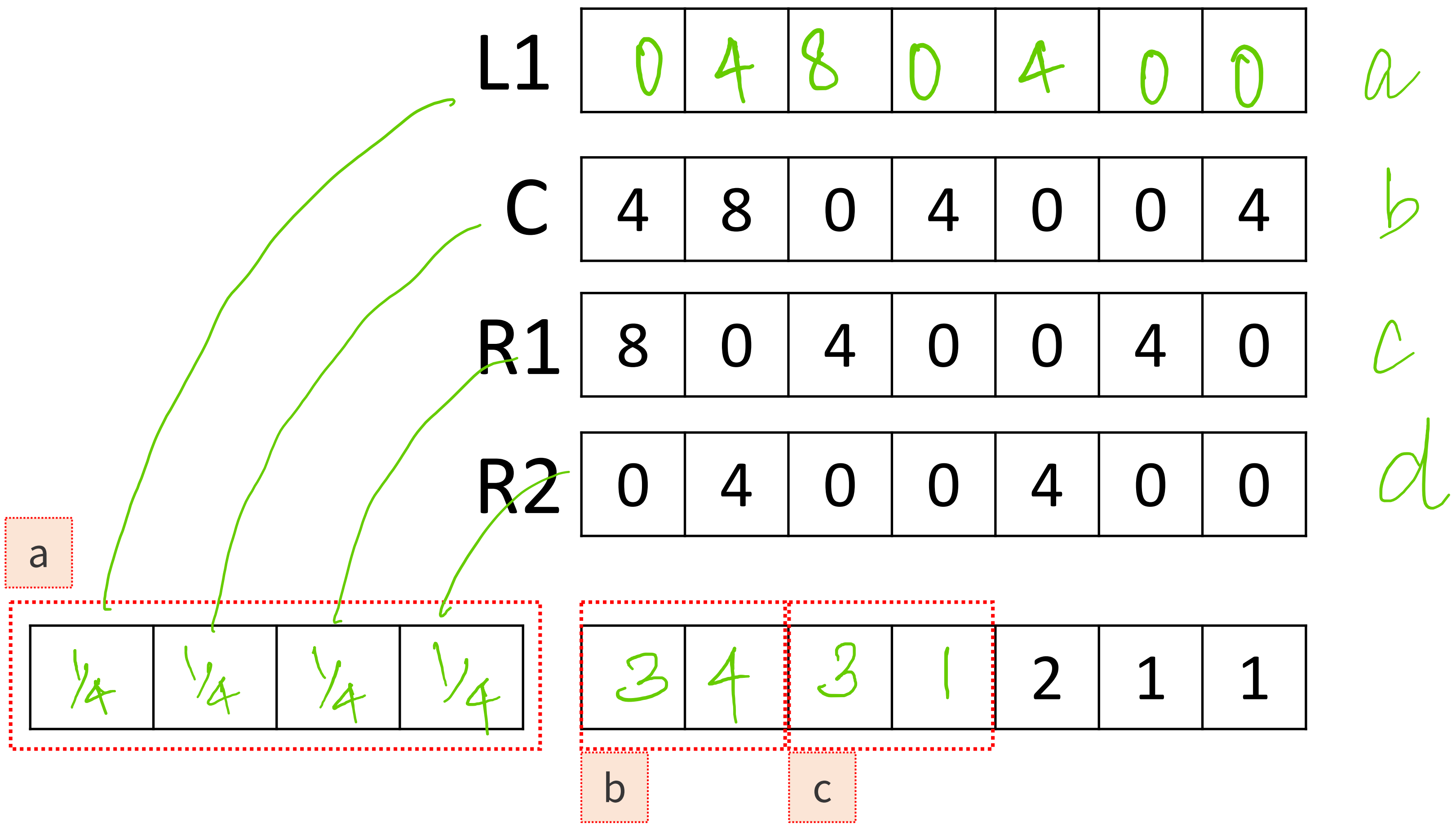
1	1	1

8

🔑 sum(a) = 3; sum(b) = 7; sum(c) = ~~7~~

Mean filter

pad with 0



L1	0	4	8	0	4	0	0	a
C	4	8	0	4	0	0	4	b
R1	8	0	4	0	0	4	0	c
R2	0	4	0	0	4	0	0	d

a

1/4	1/4	1/4	1/4

3	4	3	1	2	1	1

🔑 sum(a) = 1; sum(b) = 7; sum(c) = 4

Up or Down?

L	X	1	2	6	8	12	7	5	2	-2
C	1	4	6	8	12	7	5	2	-2	-5
R	4	6	8	12	7	5	2	-2	-5	X

Math: Cross-Correlation $H\otimes F$

	0	1	4	5	2	3	7	L
I[i]	1	4	5	2	3	7	9	C
	4	5	2	3	7	9	0	R

H[u]	1	-1	0

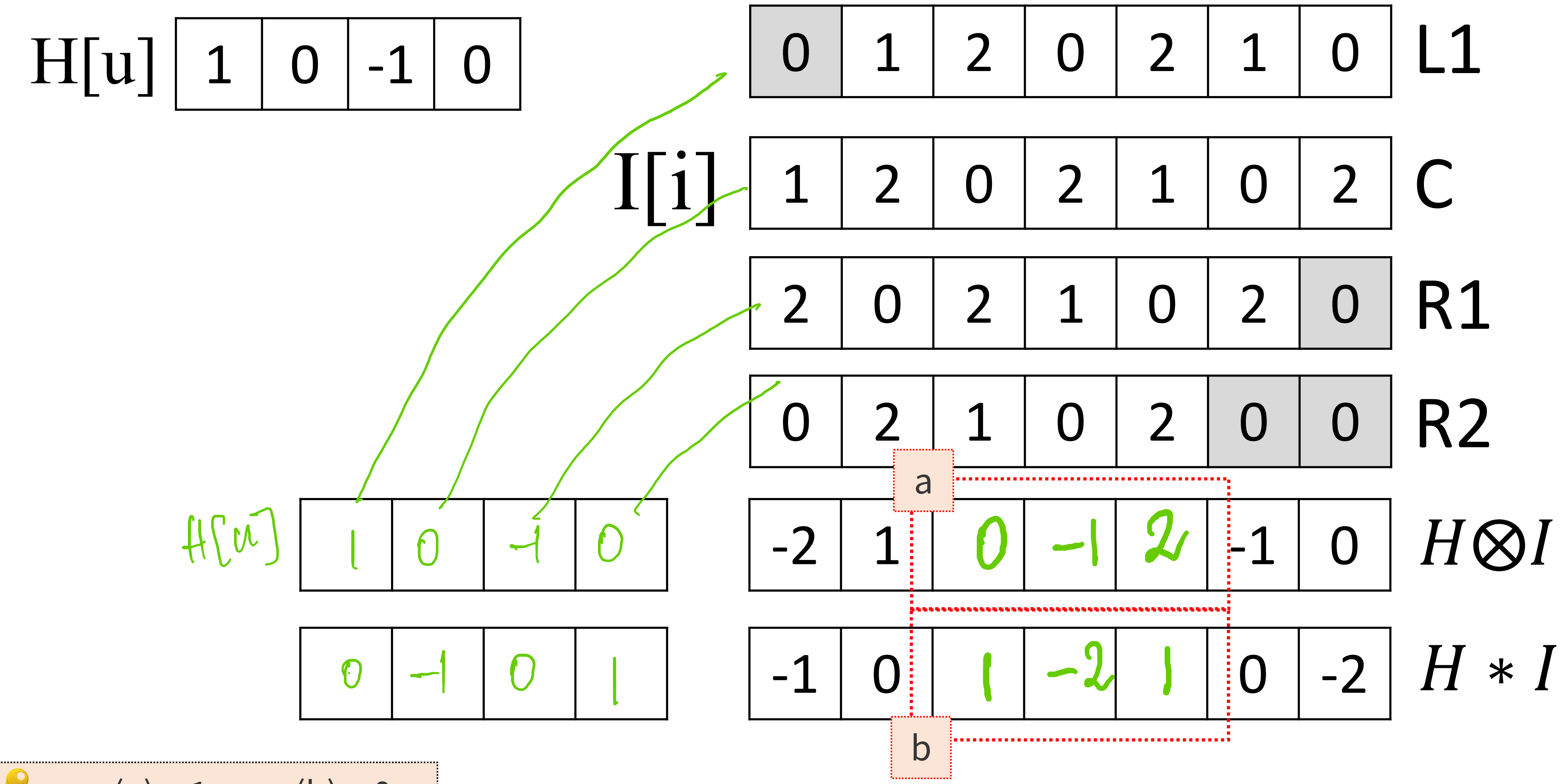
Math: Convolution $H * F$

	0	1	4	5	2	3	7	L
I[i]	1	4	5	2	3	7	9	C
	4	5	2	3	7	9	0	R

H[u]	1	-1	0

0	-1	1							

Cross Correlation ⊗ vs. Convolution ∗



H[u]

1	0	-1	0

I[i]

0	1	2	0	2	1	0	L1
1	2	0	2	1	0	2	C
2	0	2	1	0	2	0	R1
0	2	1	0	2	0	0	R2
-2	1	0	-1	2	-1	0	$H \otimes I$
-1	0	1	-2	1	0	-2	$H * I$

H[u]

1	0	-1	0
0	-1	0	1

🔑 sum(a) = 1; sum(b) = 0

Properties

$$H \otimes F = F \otimes H \qquad H * F = F * H$$

$$F \otimes H_1 \otimes H_2 = \qquad F * H_1 * H_2 =$$

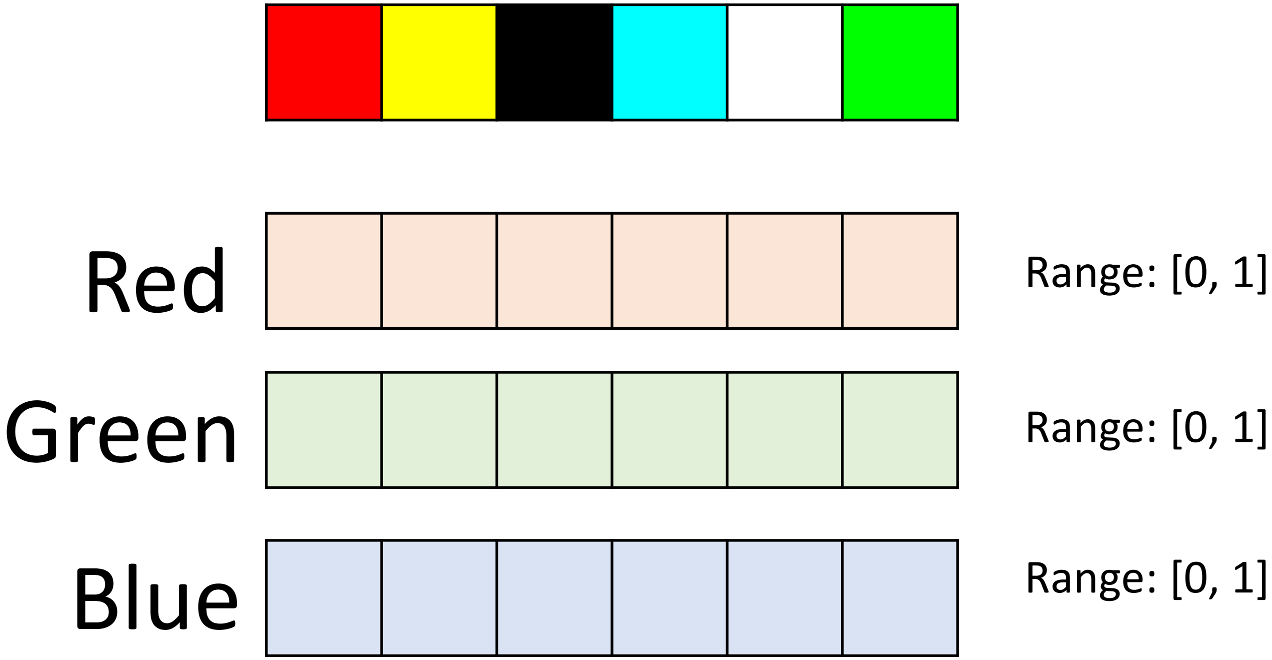
$$F \otimes (H_1 \otimes H_2) \qquad F * (H_1 * H_2)$$

Multiple Channels

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Color: floating value representation



Red

Range: [0, 1]

Green

Range: [0, 1]

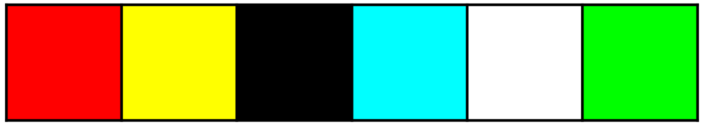
Blue

Range: [0, 1]

RGB → Grayscale

	🟥	🟨	⬛	🟦	⬜	🟩	
Red	1	1	0	0	1	0	Range: [0, 1]
Green	0	1	0	1	1	1	Range: [0, 1]
Blue	0	0	0	1	1	0	Range: [0, 1]

RGB: Float → Integer



Red	1	1	0	0	1	0
Green	0	1	0	1	1	1
Blue	0	0	0	1	1	0
.*						

Range: [0, 1]

Red						
Green						
Blue						

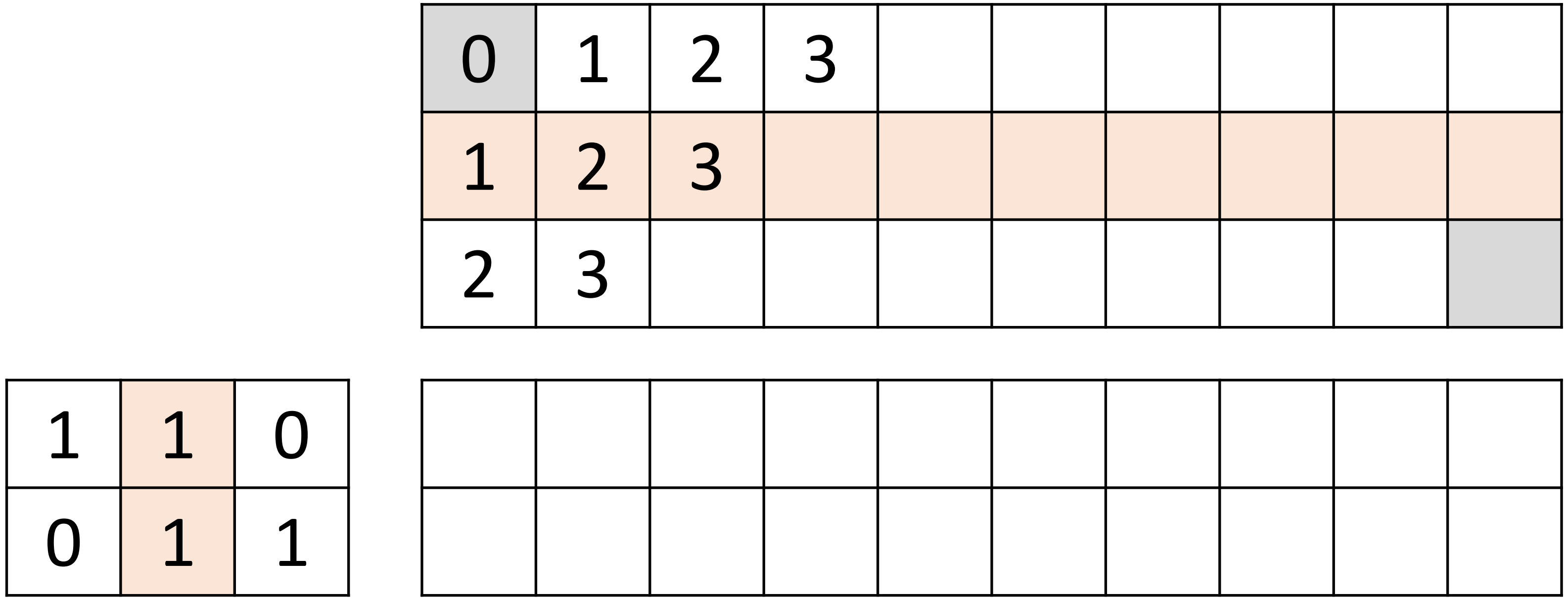
Range: [0, 255]

Scaling 1D Filtering

CSCI 5722/4722 Computer Vision

University of Colorado
Boulder

Single Channel



0	1	2	3						
1	2	3							
2	3								

1	1	0
0	1	1

Two Channels

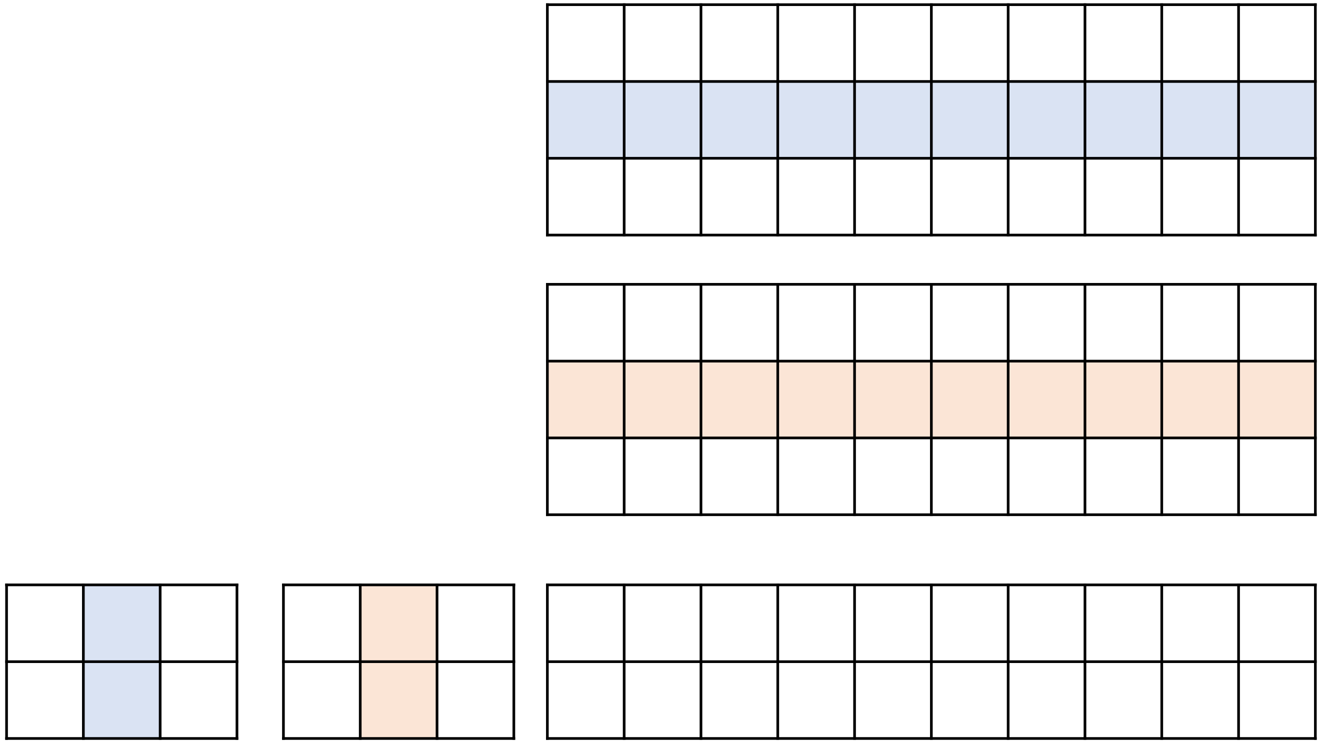
0		-1							
	-1								
-1									

0	1	2	3						
1	2	3							
2	3								

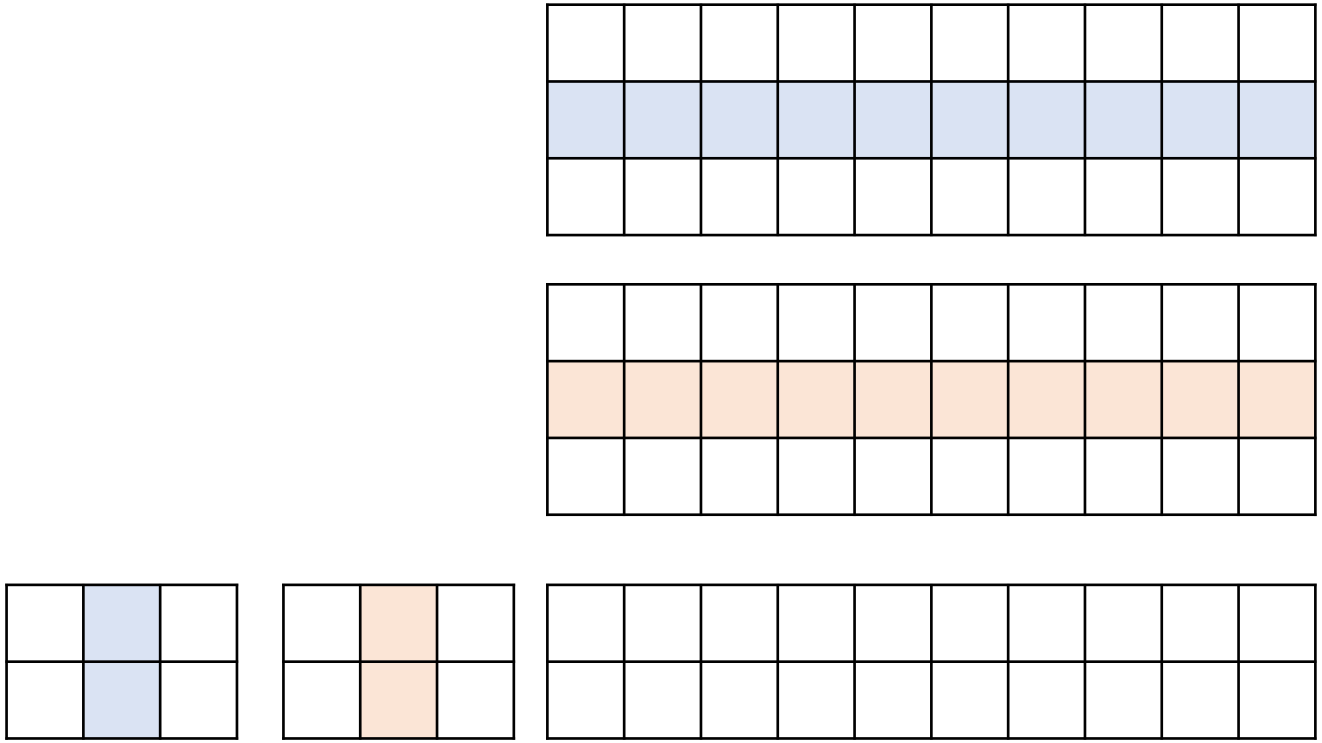
0	0	1
1	0	0

1	1	0
0	0	1

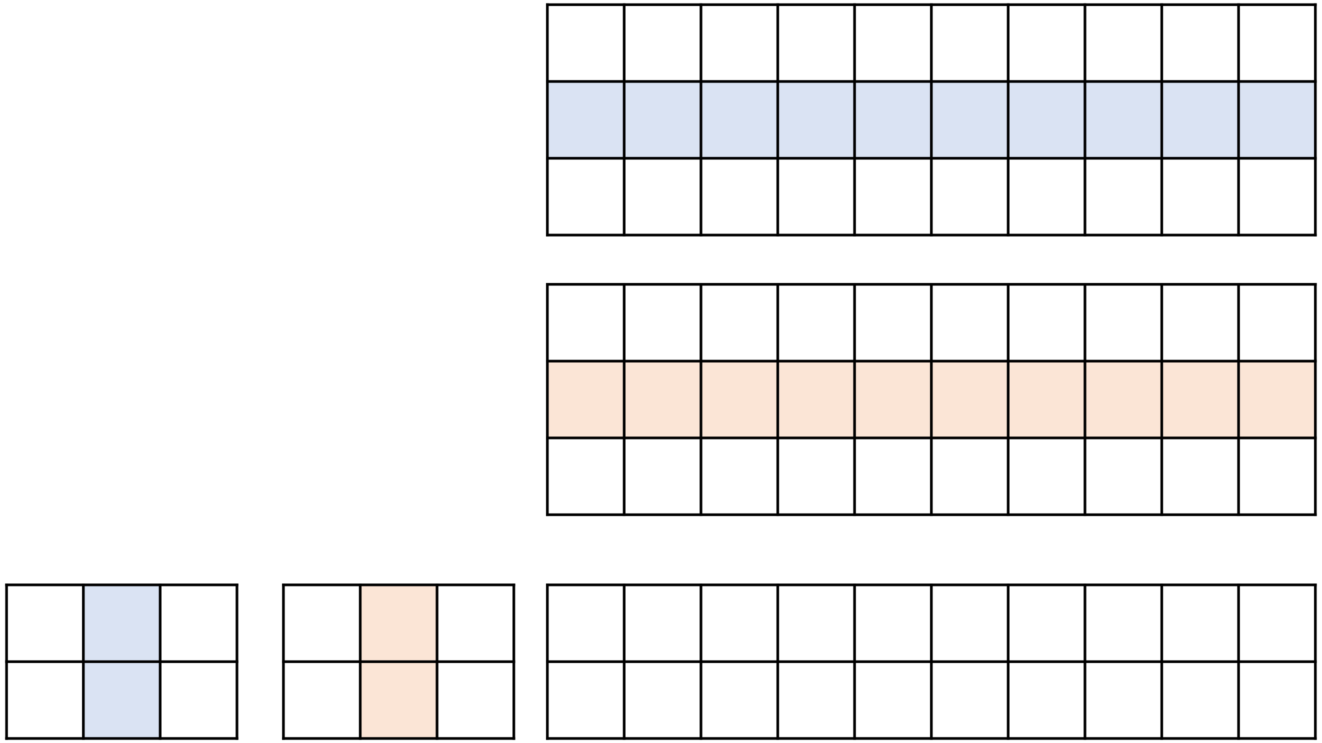
Add One Channel



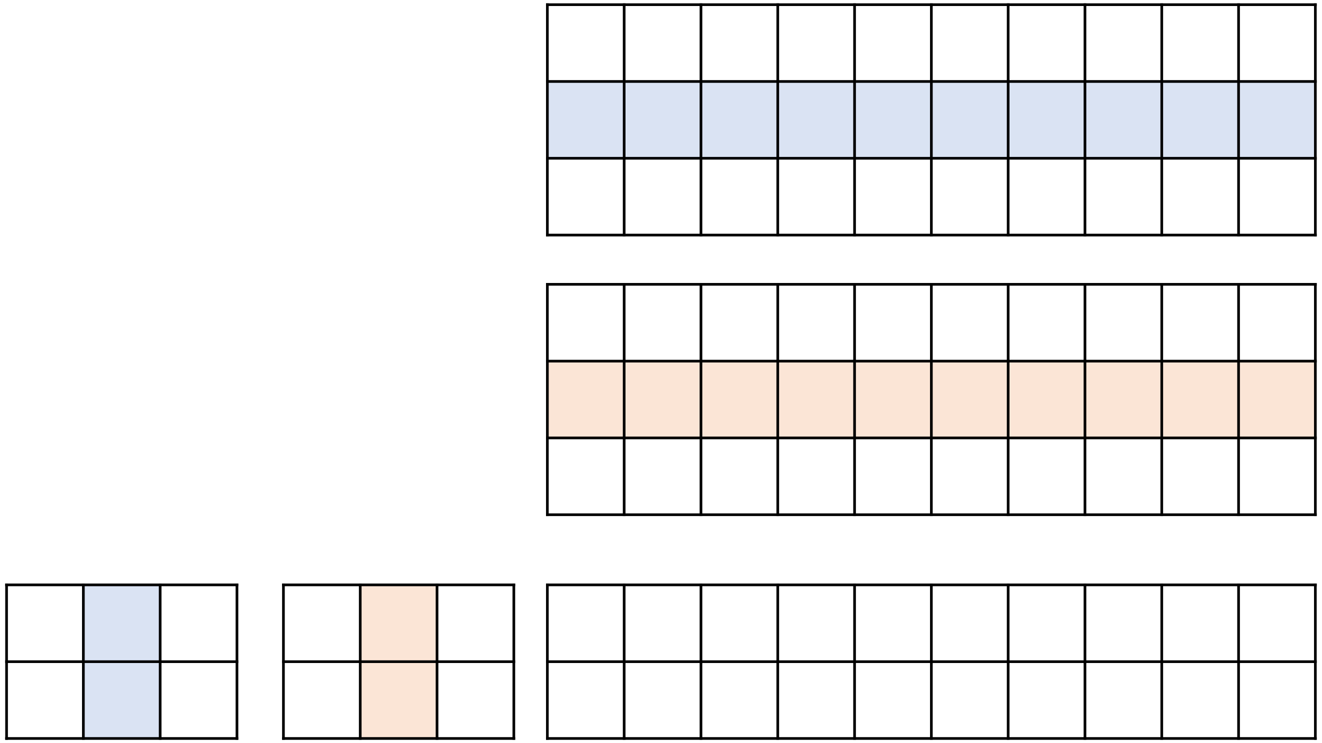
Add One Filter



Add One Neighbor

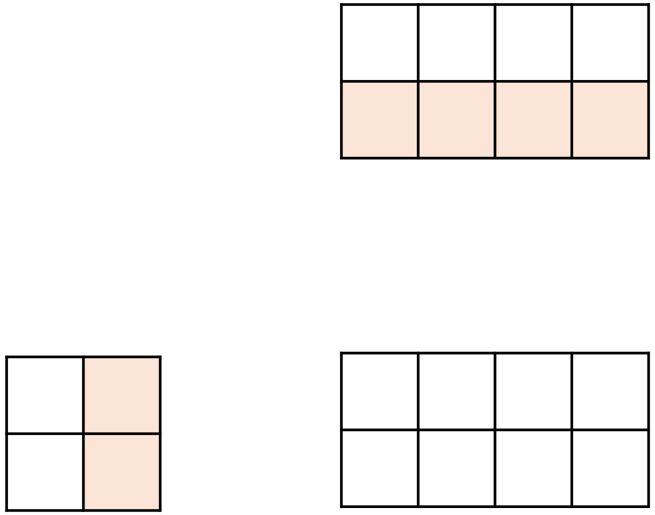


Add One Input



Add 1 Neighbor

Copy, add rows and columns
Show your work

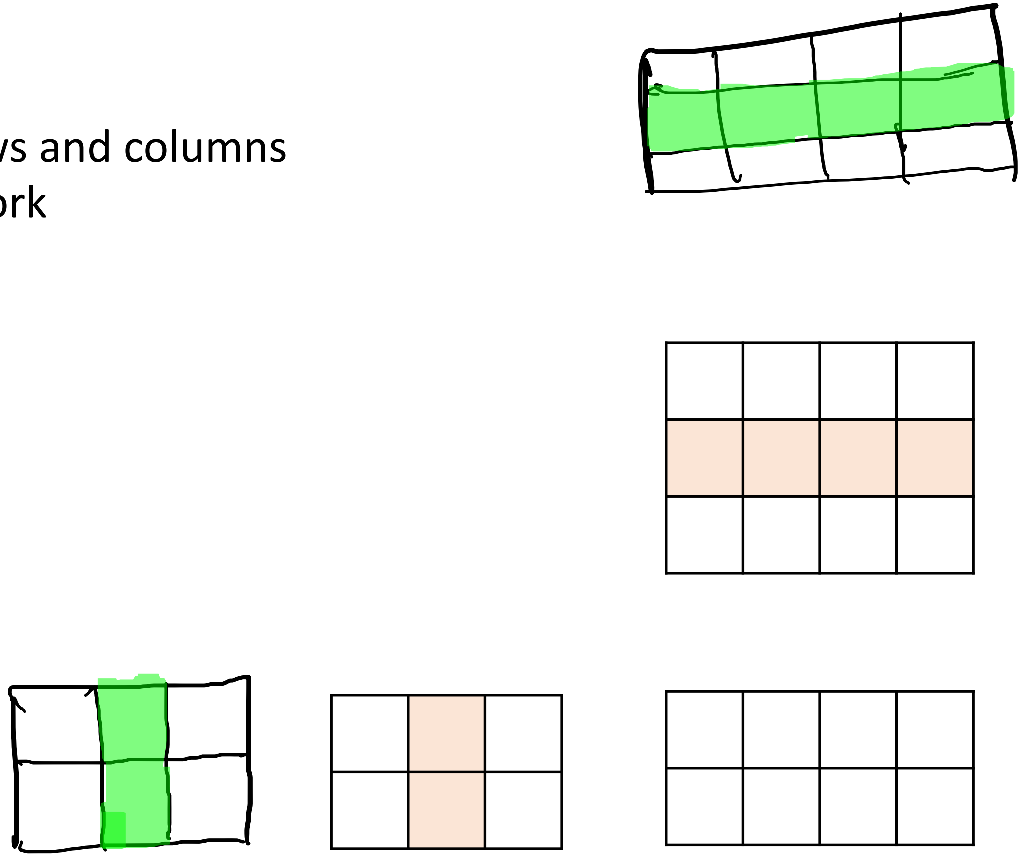


Number of new cells added = ____________

☑ ○

Add 1 Channel

Copy, add rows and columns
Show your work



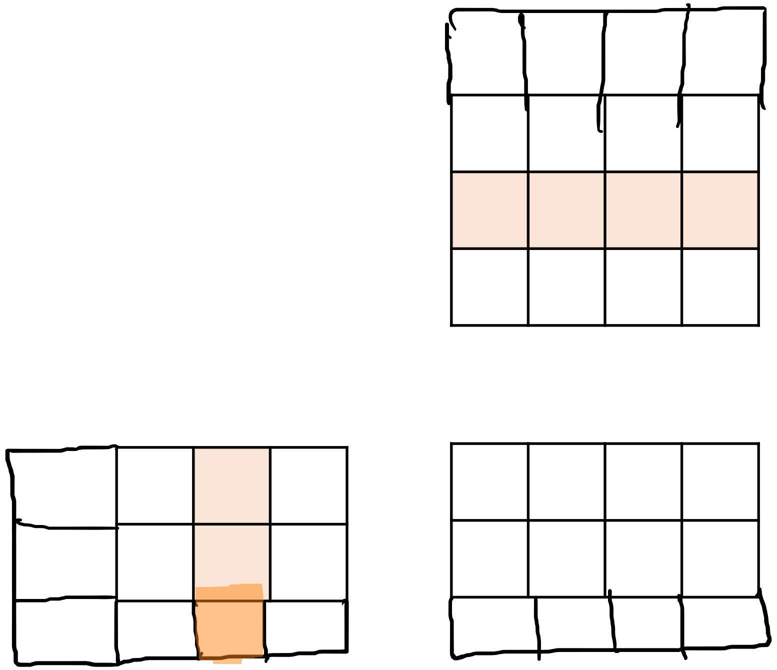
a

Number of new cells added = 18

🔑 a % 4 = 2

Add 1 Neighbor + 1 Filter

Copy, add rows and columns
Show your work



a

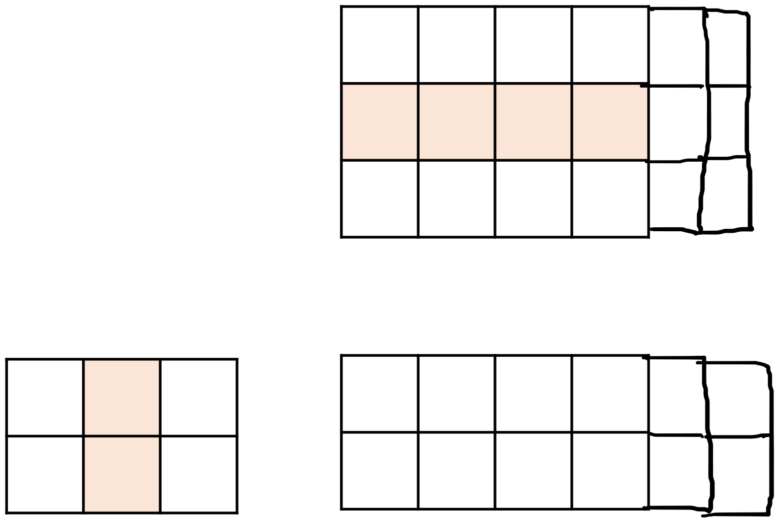
Number of new cells added = 14

a % 6 = 2

☑ ○

Add 2 inputs

Copy, add rows and columns
Show your work



a

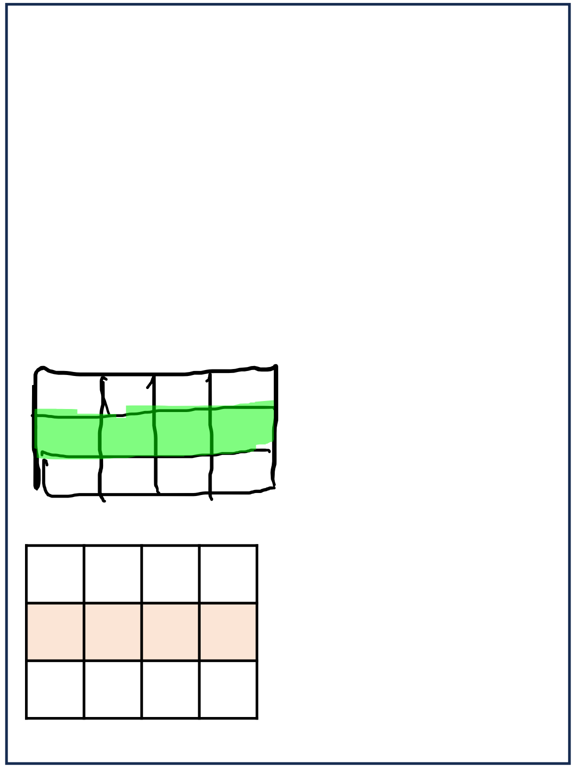
Number of new cells added = 10

🔑 a % 3 = 1

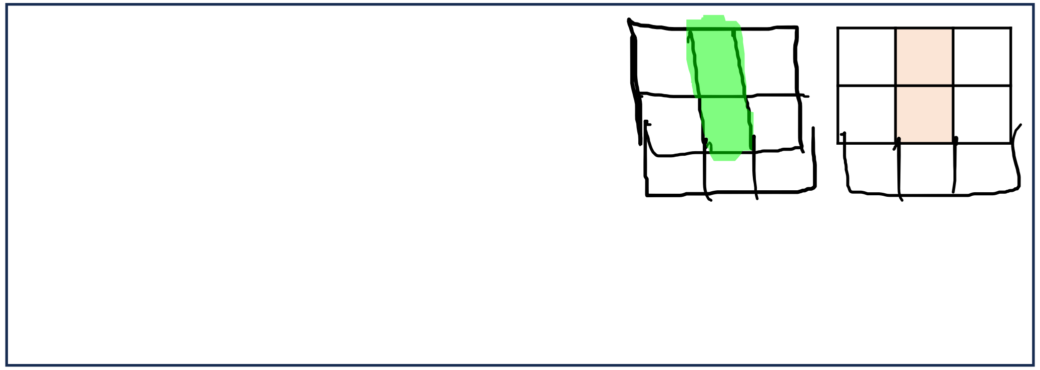
Scale to 2 channels, 3 filters

Copy, add rows and columns
Show your work

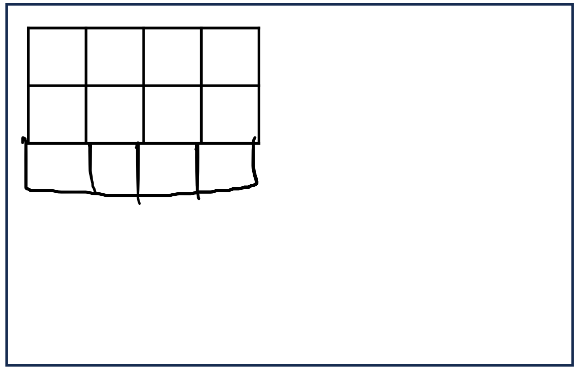
X



K



Z



a b

shape(K) = (3, 6)

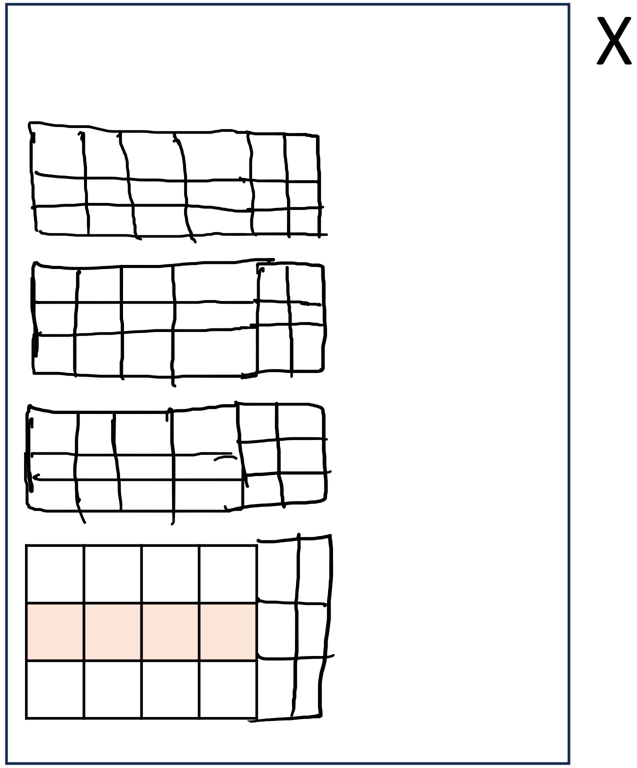
shape(X) = (6, 4)

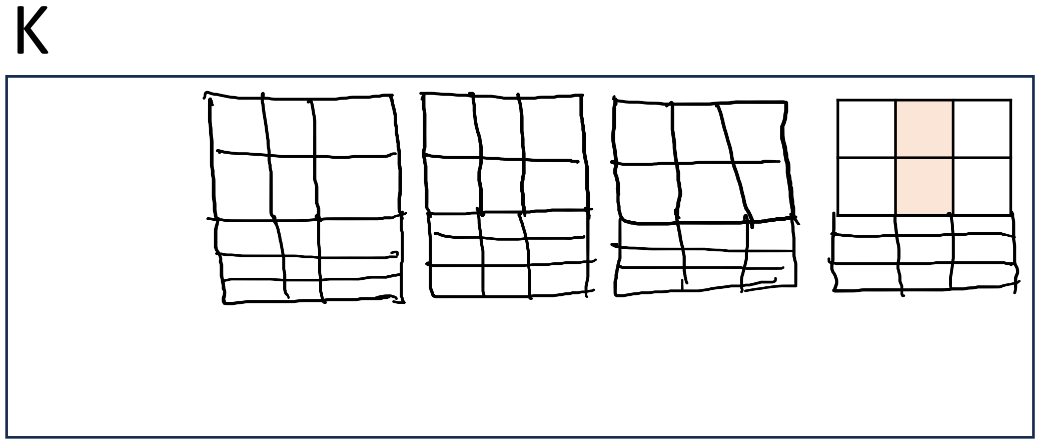
shape(Z) = (3, 4)

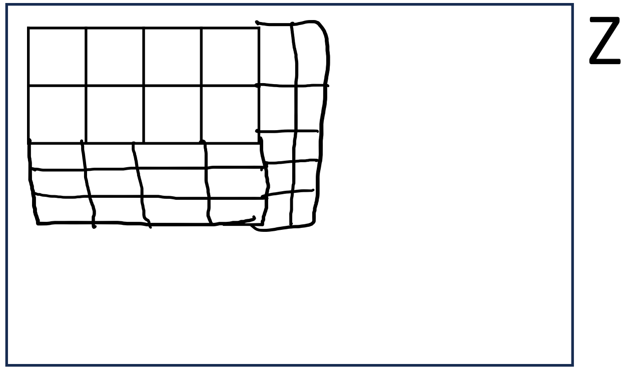
🔑 sum(a)=12; sum(b)=14

Scale to 4 channels, 5 filters, 6 inputs

Copy, add rows and columns
Show your work







a b

shape(K) = (5, 12)

shape(X) = (12, 6)

shape(Z) = (5, 6)

🔑 sum(a)=22; sum(b)=24

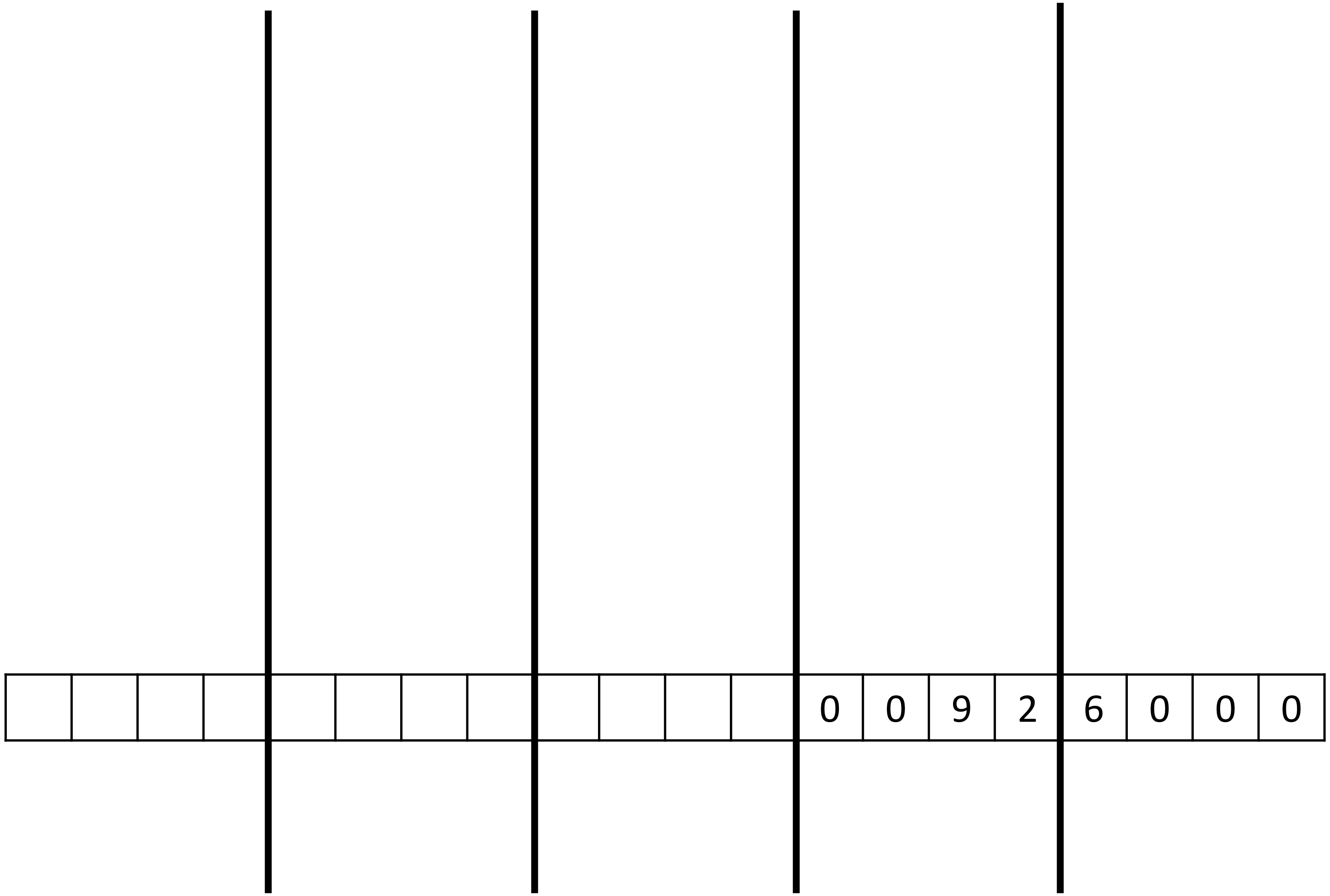
2D Filters

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Flatten

0	2	3	0
0	0	4	0
3	5	0	0
0	0	9	2
6	0	0	0



												0	0	9	2	6	0	0	0

Center (Self)

0	2	3	0
0	0	4	0
3	5	0	0
0	0	9	2
6	0	0	0

	C	

C=3

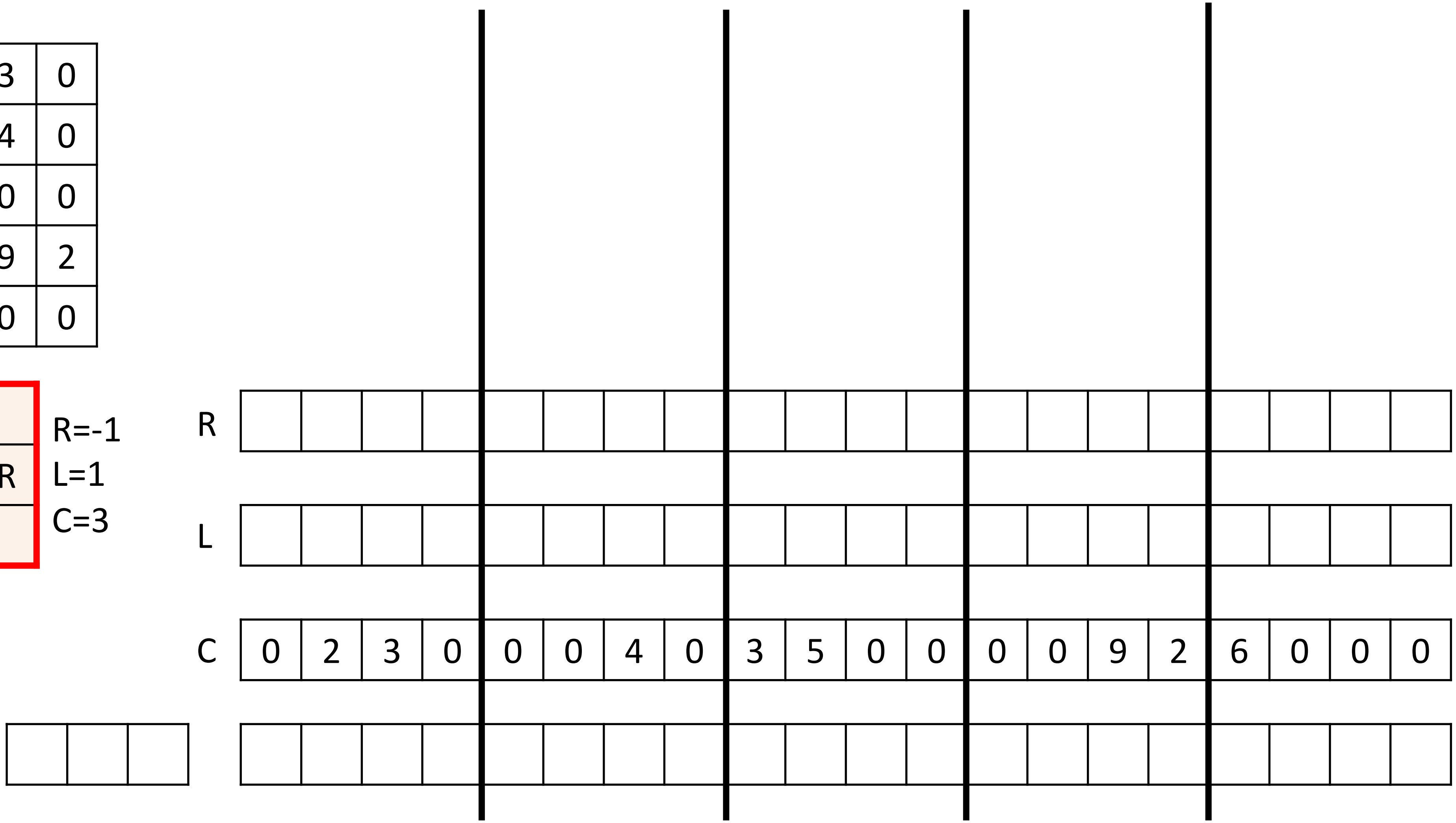
C	0	2	3	0	0	0	4	0	3	5	0	0	0	0	9	2	6	0	0	0

Left/Right Neighbors

0	2	3	0
0	0	4	0
3	5	0	0
0	0	9	2
6	0	0	0

L	C	R

R=-1
L=1
C=3



R																				
L																				
C	0	2	3	0	0	0	4	0	3	5	0	0	0	0	9	2	6	0	0	0

Above/Below Neighbors

0	2	3	0
0	0	4	0
3	5	0	0
0	0	9	2
6	0	0	0

	A	
L	C	R
	B	

A=-2
B=2
R=-1
L=1
C=3

A									0	0	4	0	3	5	0	0				
B									0	0	9	2	6	0	0	0				
R	2	3	0	x	0	4	0	x	5	0	0	x	0	9	2	x	0	0	0	x
L	x	0	2	3	x	0	0	4	x	3	5	0	x	0	0	9	x	6	0	0
C	0	2	3	0	0	0	4	0	3	5	0	0	0	0	9	2	6	0	0	0

Corner Neighbor

0	2	3	0
0	0	4	0
3	5	0	0
0	0	9	2
6	0	0	0

Q	T	
L	C	R
	B	

Q=-3
A=-2
B=2
R=-1
L=1
C=3

Q													x	3	5	0	x	0	0	9
A	x	x	x	x	0	2	3	0	0	0	4	0	3	5	0	0	0	0	9	2
B	0	0	4	0	3	5	0	0	0	0	9	2	6	0	0	0	x	x	x	x
R	2	3	0	x	0	4	0	x	5	0	0	x	0	9	2	x	0	0	0	x
L	x	0	2	3	x	0	0	4	x	3	5	0	x	0	0	9	x	6	0	0
C	0	2	3	0	0	0	4	0	3	5	0	0	0	0	9	2	6	0	0	0

Left/Right Neighbors

0	2	3	0
0	0	4	0
0	3	5	0
2	0	9	0
0	0	0	6

L	C	R

R=1
L=-1
C=1

🔑 sum(a)=5; sum(b)=4

Calculate a, b. Pad with -1 instead of X.

R	2	3	0	-1	0	4	0	-1	3	5	0	-1	0	9	0	-1	0	0	6	-1
L	-1	0	2	3	-1	0	4	0	-1	0	3	5	-1	2	0	9	-1	0	0	0
C	0	2	3	0	0	0	4	0	0	3	5	0	2	0	9	0	0	0	0	6
	3	5	1	-4	1	4	0	-1	4	8	2	-6	3	7	9	-10	1	0	6	5

1	-1	1

a

b

Corner Neighbors

🔑 sum(a)=17; sum(b)=~~10~~; sum(c) = -2

↓

11

0	2	3	0
0	0	4	0
0	3	5	0
2	0	9	0
0	0	0	6

A		
	C	
		B

A=-1
B=1
C=2

Calculate a, b, c. Pad with -1 instead of X.

A	-1	-1	-1	-1	-1	0	2	3	-1	0	0	4	-1	0	3	5	-1	2	0	9
B	0	4	0	-1	3	5	0	-1	0	9	0	-1	0	0	6	-1	-1	-1	-1	-1
C	0	2	3	0	0	0	4	0	0	3	5	0	2	0	9	0	0	0	0	6
	1	9	7	0	4	5	6	-4	1	9	10	-8	3	0	21	-6	0	-3	-1	2
	a				b												c			

-1	1	2

Math: Cross-Correlation vs. Convolution in 2D

$$H \qquad F[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i \qquad u, j \qquad v]$$

$$H \qquad F[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i \qquad u, j \qquad v]$$

Math: Cross-Correlation vs. Convolution in 2D

Complete the equations by replacing ? with the right expressions. Adjust the sizes of the font or textbox if necessary.

$$H \otimes F[x,y] = \sum_{p=-k}^{k} \sum_{q=-k}^{k} H[x+p]F[y+q]$$

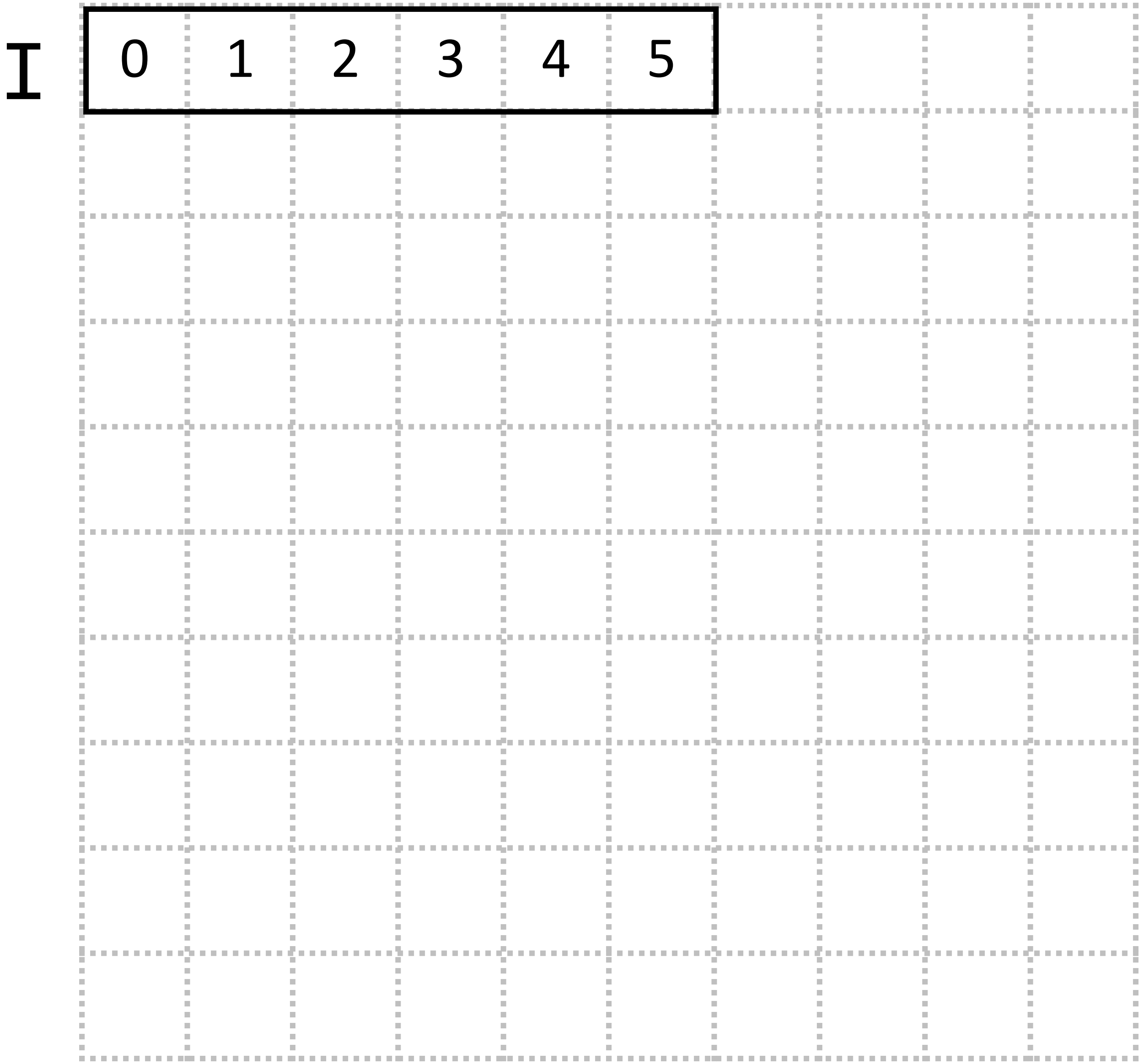
$$H * F[x,y] = \sum_{p=-k}^{k} \sum_{q=-k}^{k} H[x-p]F[y-q]$$

NumPy by Hand ✍️
Broadcast

CSCI 5722/4722 Computer Vision

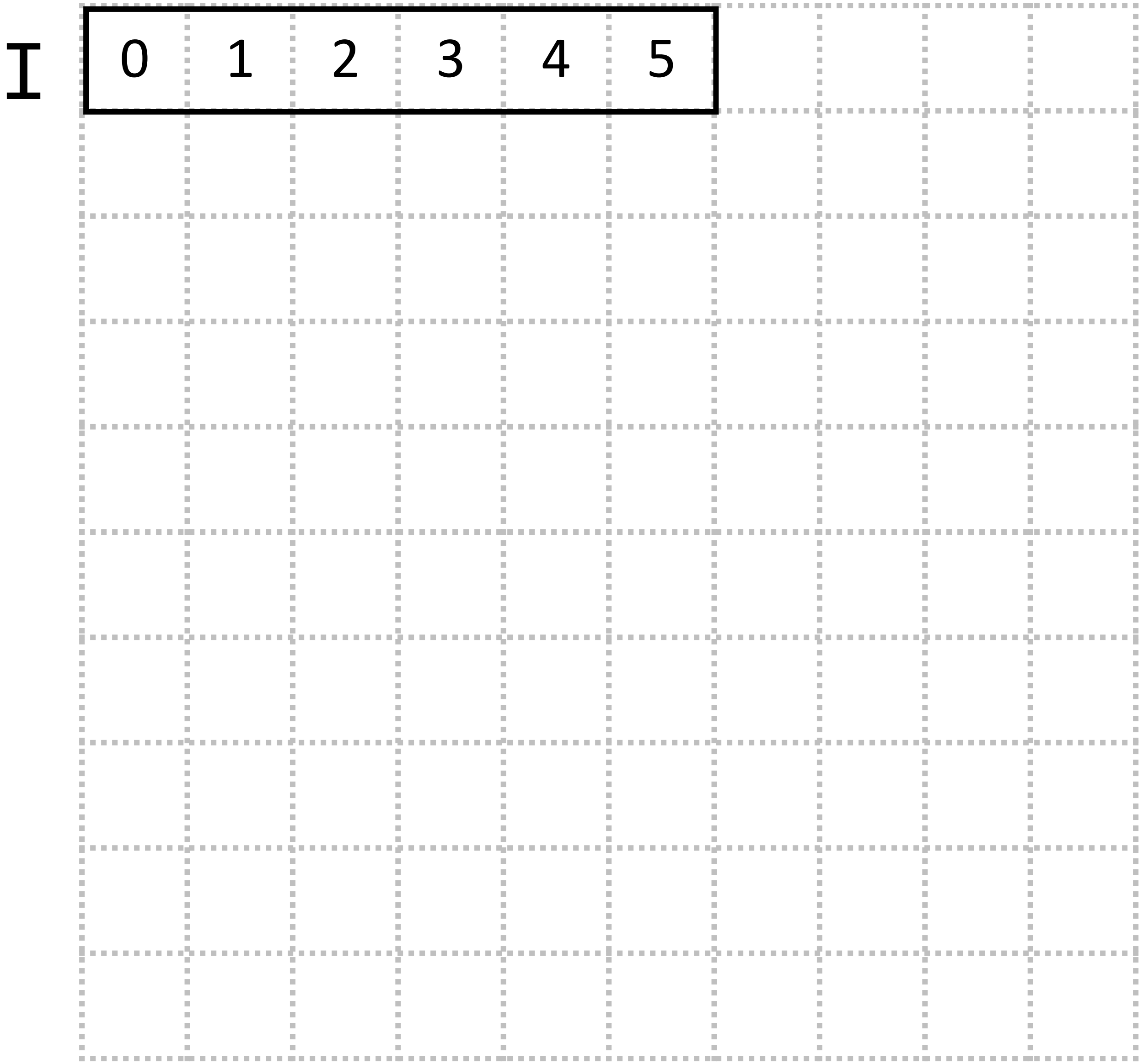
University of Colorado
Boulder

```
1. a = I * 2
```



1D

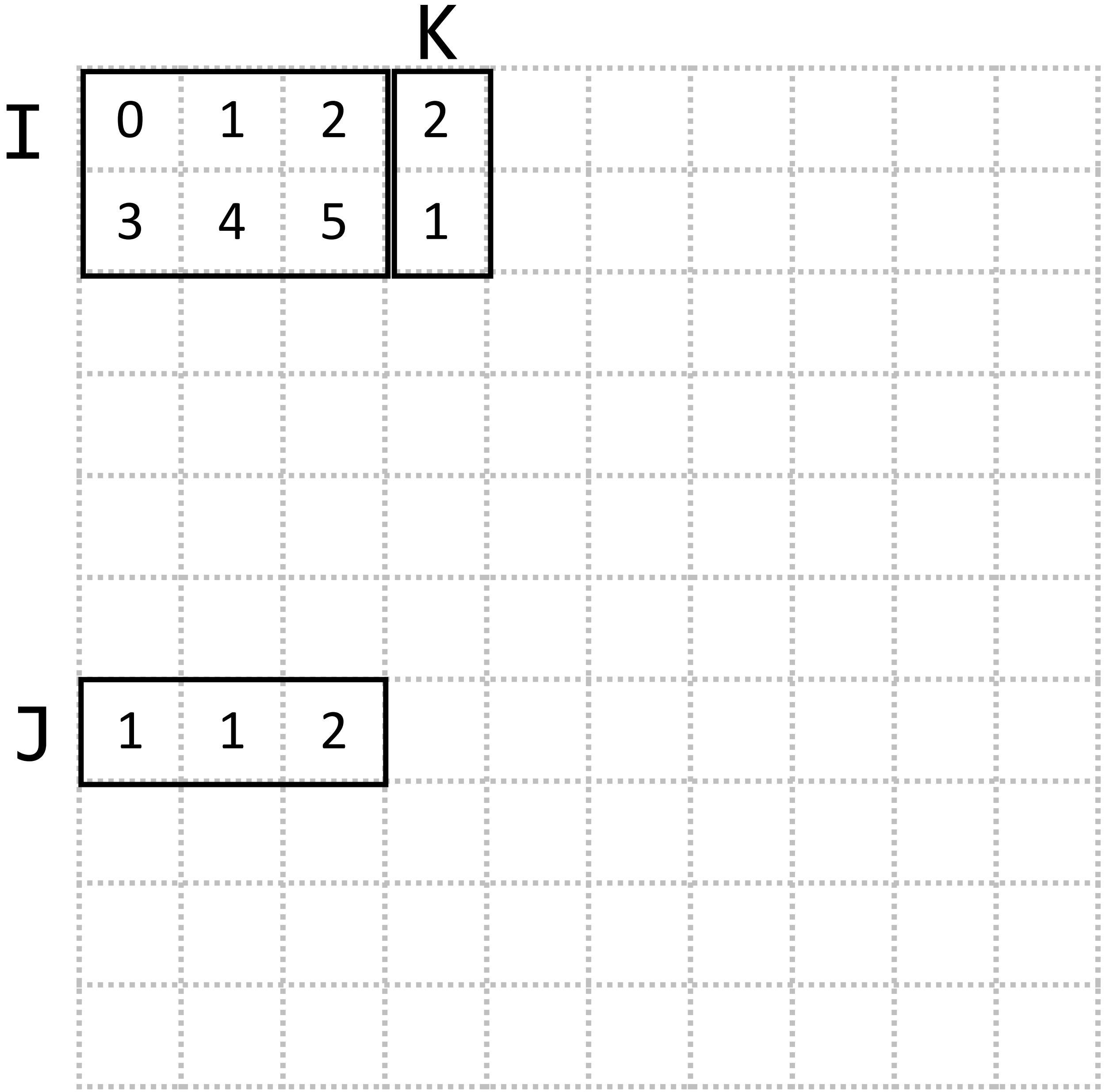
```
1. a = I * 2

2. b = a + 3

3. c = np.ones(2)

4. d = b - c
```



I	0	1	2	3	4	5

2D

```
1. a = I * 2

2. b = a + J

3. c = I + K
```



I

0	1	2
3	4	5

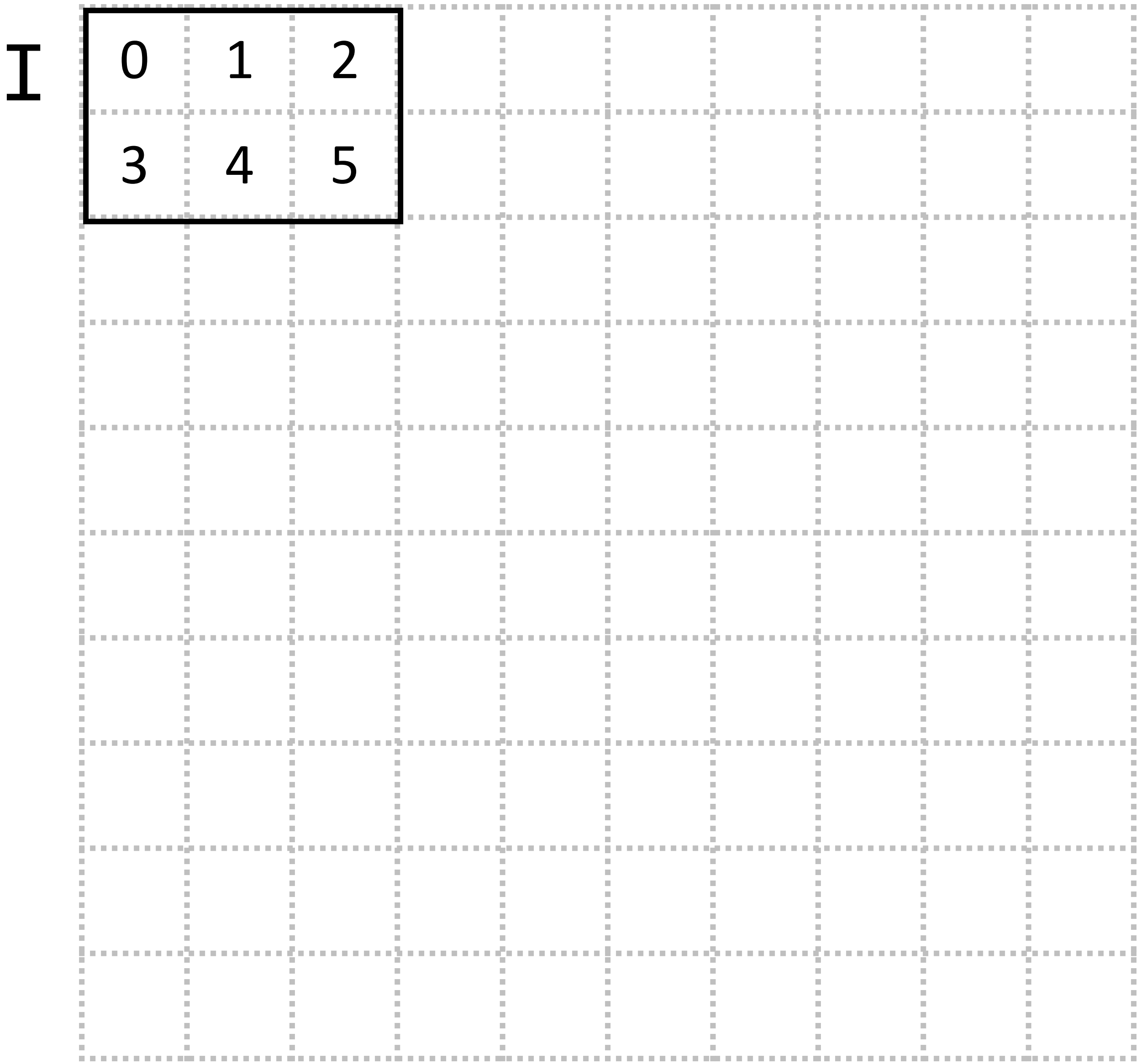
K

2
1

J

1	1	2

2D

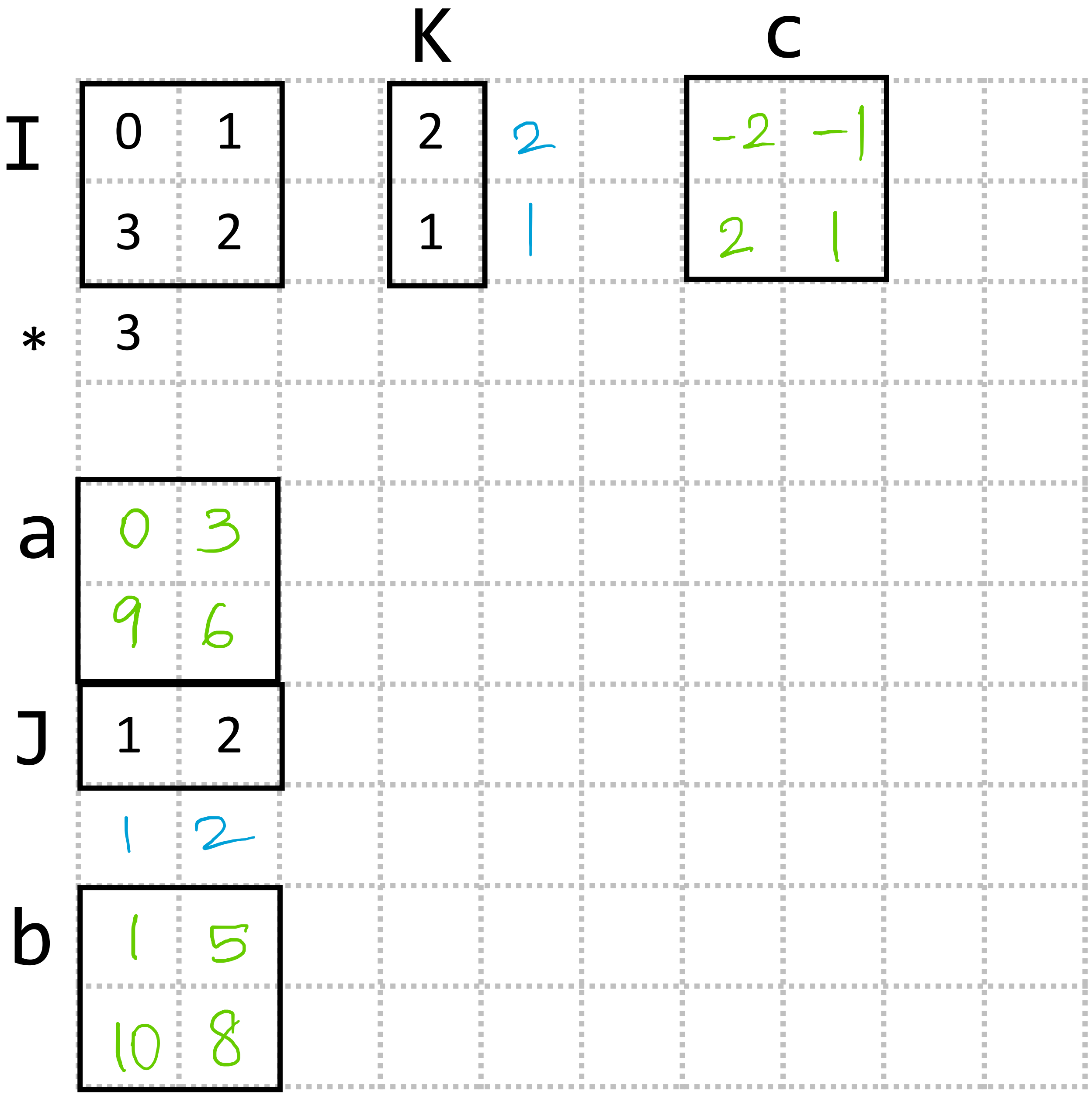


I

0	1	2
3	4	5

✓ ○ Execute by ✍

```
1. a = I * 3

2. b = a + J

3. c = I - K
```



I $\begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix}$ * 3

K $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ $\begin{matrix} 2 \\ 1 \end{matrix}$

c $\begin{bmatrix} -2 & -1 \\ 2 & 1 \end{bmatrix}$

a $\begin{bmatrix} 0 & 3 \\ 9 & 6 \end{bmatrix}$

J $\begin{bmatrix} 1 & 2 \end{bmatrix}$

$\begin{matrix} 1 & 2 \end{matrix}$

b $\begin{bmatrix} 1 & 5 \\ 10 & 8 \end{bmatrix}$

🔑 sum(b)=24; sum(c)=0

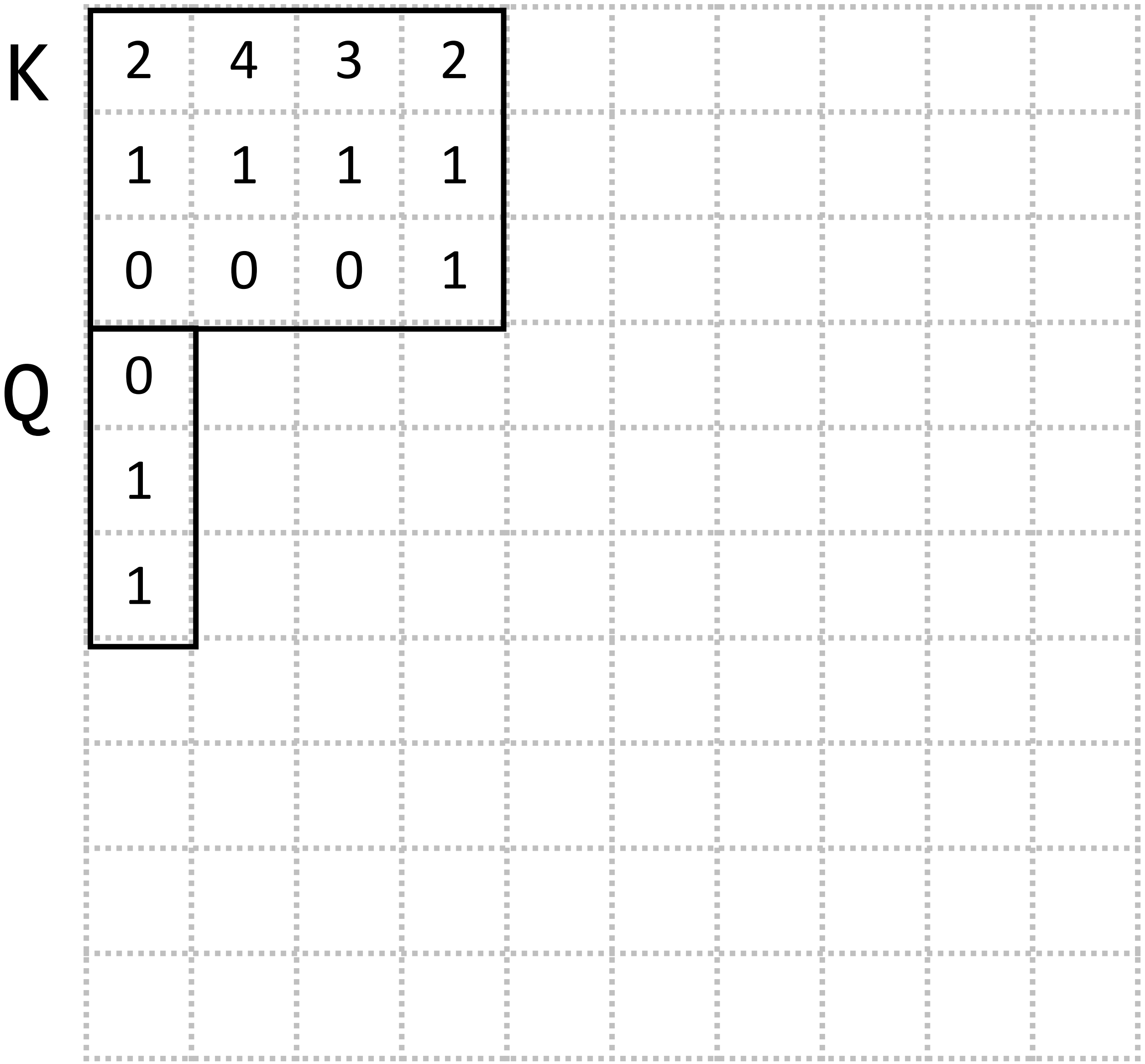
NumPy by Hand ✍️ Compare

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

L1

```
1.  a = K - Q
2.  b = np.abs(a)
3.  c = np.sum(b, axis=0)
```



K

2	4	3	2
1	1	1	1
0	0	0	1

Q

0
1
1

☑ □ Compute L1 ✍️

```
1. a = K - Q

2. b = np.abs(a)

3. c = np.sum(b, axis=0)
```

K	1	4	3	3
	2	2	1	2
Q	1	1	1	1
	3	3	3	3
a	0	3	2	2
	-1	-1	-2	-1
b	0	3	2	2
	1	1	2	1
c	1	4	4	3

🔑 sum(c)=12

☑ □ Compute L1 ✍

```
1. a = K - Q
2. b = np.abs(a)
3. c = np.sum(b, axis=1)
```

			Q		C
K	1	3	1	2	1
	3	2	1	2	2
	1	1	1	2	1
	2	2	1	2	1
a	0	1	0	1	
	2	0	2	0	
	0	-1	0	1	
	1	0	1	0	
			b		

🔑 sum(c)=5

Code with ChatGPT [Filtering]

CSCI 5722/4722 Computer Vision

University of Colorado Boulder

Ask ChatGPT to generate python code to convert your “travel” photo to a grayscale image. Submit the results below.

Code from ChatGPT

```
{Replace this code block, adjust the font size if necessary
to fit}

import numpy as np
from PIL import Image

def flip_image_vertically(input_path, output_path):
  # Open the image and convert it to a numpy array
  with Image.open(input_path) as img:
    img_array = np.array(img)

  # Flip the image array vertically
  flipped_array = np.flipud(img_array)

  # Convert the flipped array back to an image
  flipped_image = Image.fromarray(flipped_array)

  # Save the flipped image
  flipped_image.save(output_path)

# Example usage
flip_image_vertically('path/to/your/image.jpg',
'path/to/save/flipped_image.jpg’)
```

RGB

Grayscale

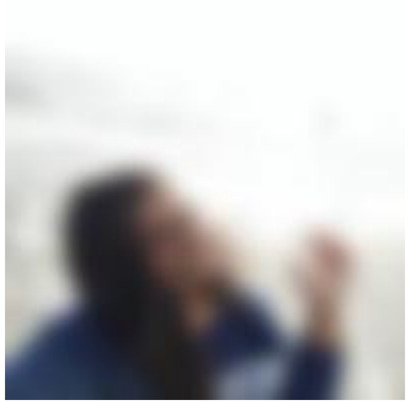
Ask ChatGPT to generate python code to resize a photo to (200,200) and apply a ”Gaussian” filter to it. Use the code to process your travel photo with kernel sizes 5 and 15. Submit the results below.

Code from ChatGPT

{Replace this code block, adjust the font size if necessary to fit}

```
import numpy as np
from PIL import Image

def flip_image_vertically(input_path, output_path):
    # Open the image and convert it to a numpy array
    with Image.open(input_path) as img:
        img_array = np.array(img)

    # Flip the image array vertically
    flipped_array = np.flipud(img_array)

    # Convert the flipped array back to an image
    flipped_image = Image.fromarray(flipped_array)

    # Save the flipped image
    flipped_image.save(output_path)

# Example usage
flip_image_vertically('path/to/your/image.jpg',
'path/to/save/flipped_image.jpg’)
```

Kernel size = 5



Kernel size = 15