

# Data Science

Lecture 9-1: Image Data Processing  
(Image Filtering)



UNIVERSITY  
OF AMSTERDAM

Lecturer: Yen-Chia Hsu

Date: Mar 2025

This lecture covers basic concepts of image filtering and feature extraction.

There are many different types of tasks in [Computer Vision](#). This lecture will only cover the image/video classification, which gives only one label to an image/video.

## Classification



## Semantic Segmentation



No spatial extent

## Object Detection



No objects, just pixels

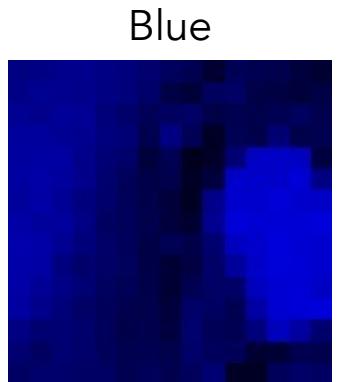
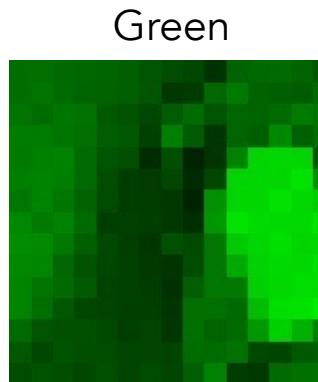
## Instance Segmentation



Multiple Object

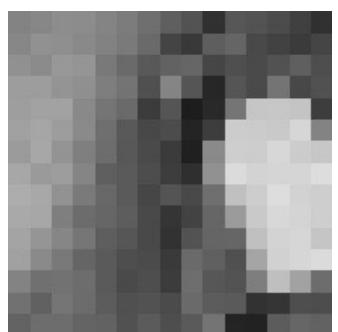
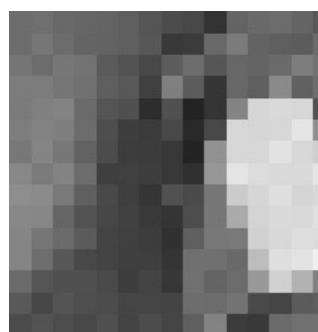
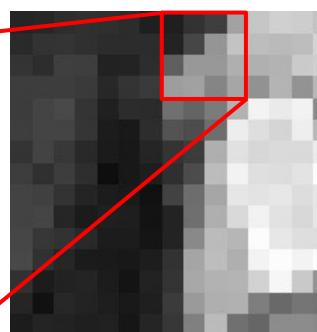
[This image is CC0 public domain](#)

People can see images directly, but computers read only numbers. Typically, computers store images as **pixels** with RGB channels with values ranging from 0 to 255.



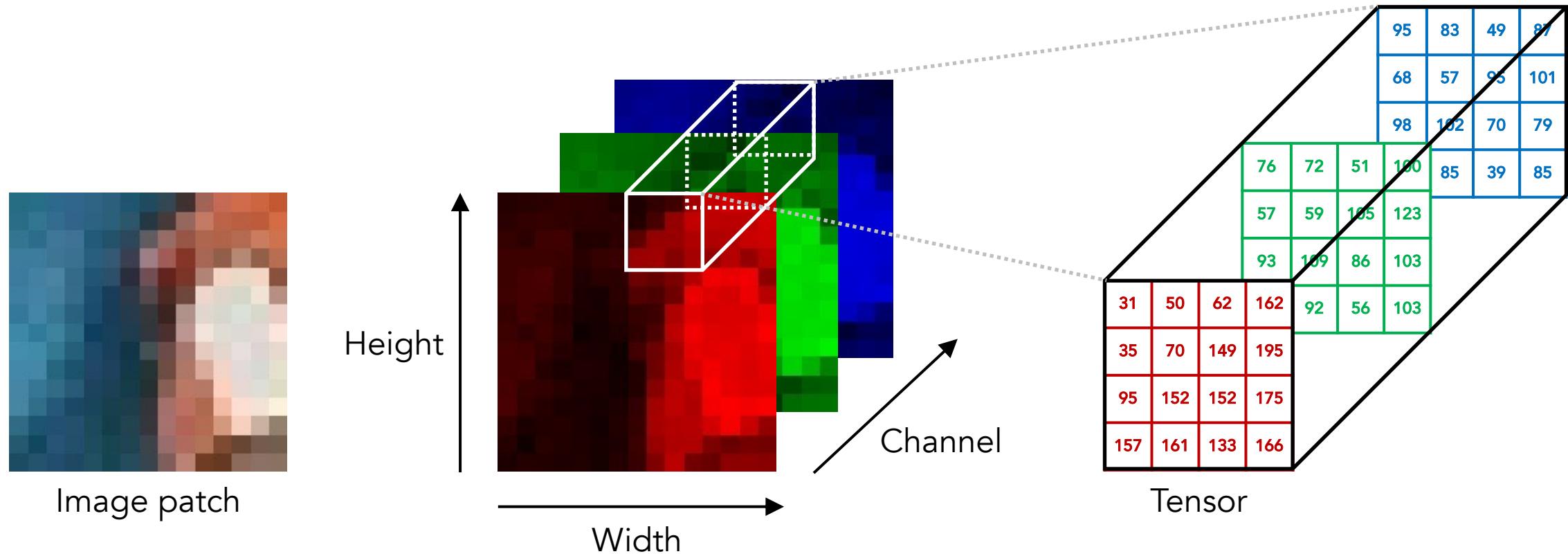
Colored visualization of the RGB channels

31	50	62	162
35	70	149	195
95	152	152	175
157	161	133	166



The intensity values (pixel values) per channel

Images can be represented as a 3D tensor (width\*height\*channels). For RGB images, we have 3 channels corresponding to the pixel intensity of red, green, and blue colors.



Before the deep learning era, Computer Vision used hand-crafted features. Researchers developed different image filters/kernels to extract features using **convolution**.



Input image

Discrete convolution: sum of the element-wise multiplication

Input				
7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

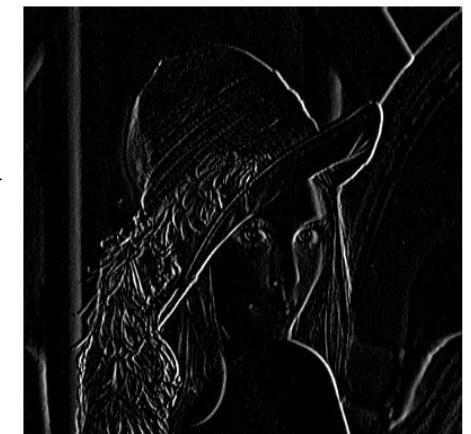
Filter  
(Kernel)

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

\*

6		

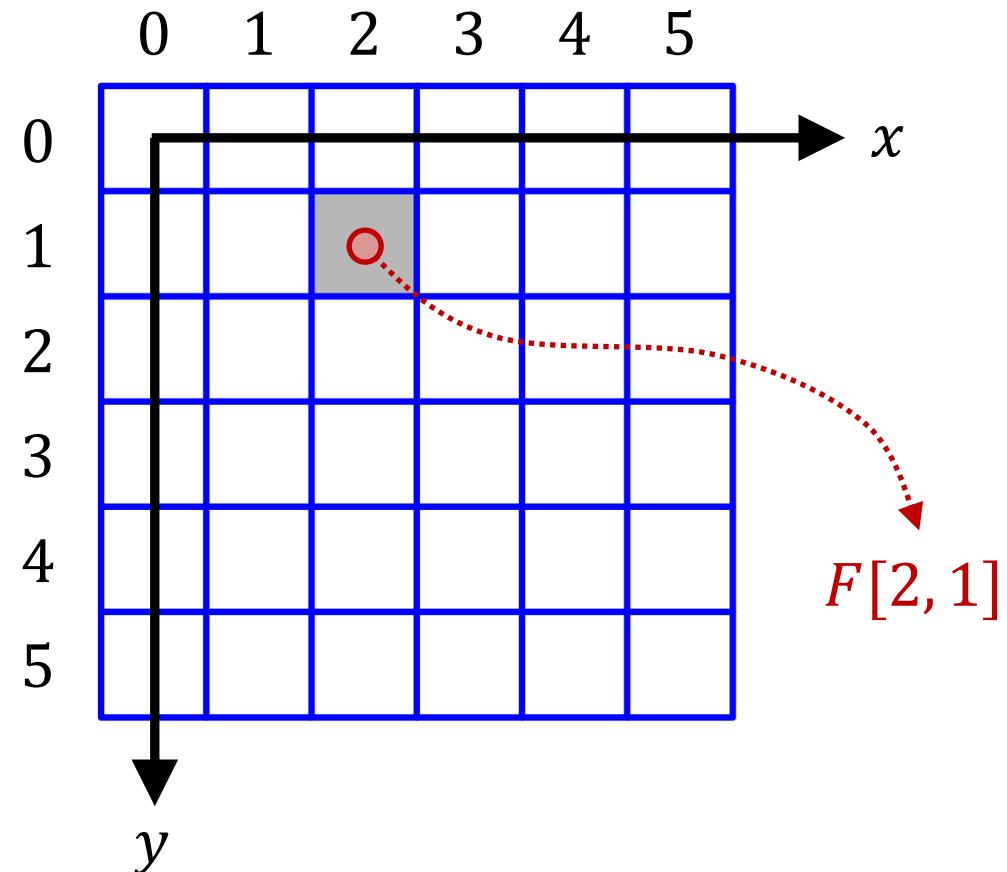
Output



Output image

$$\begin{aligned} & 7 \times 1 + 4 \times 1 + 3 \times 1 + \\ & 2 \times 0 + 5 \times 0 + 3 \times 0 + \\ & 3 \times -1 + 3 \times -1 + 2 \times -1 \\ & = 6 \end{aligned}$$

About image coordinates, the origin is at the top-left pixel. Notation  $F[x, y]$  means the value of the center of the pixel for image  $F$  at location  $[x, y]$  in the 2D array.

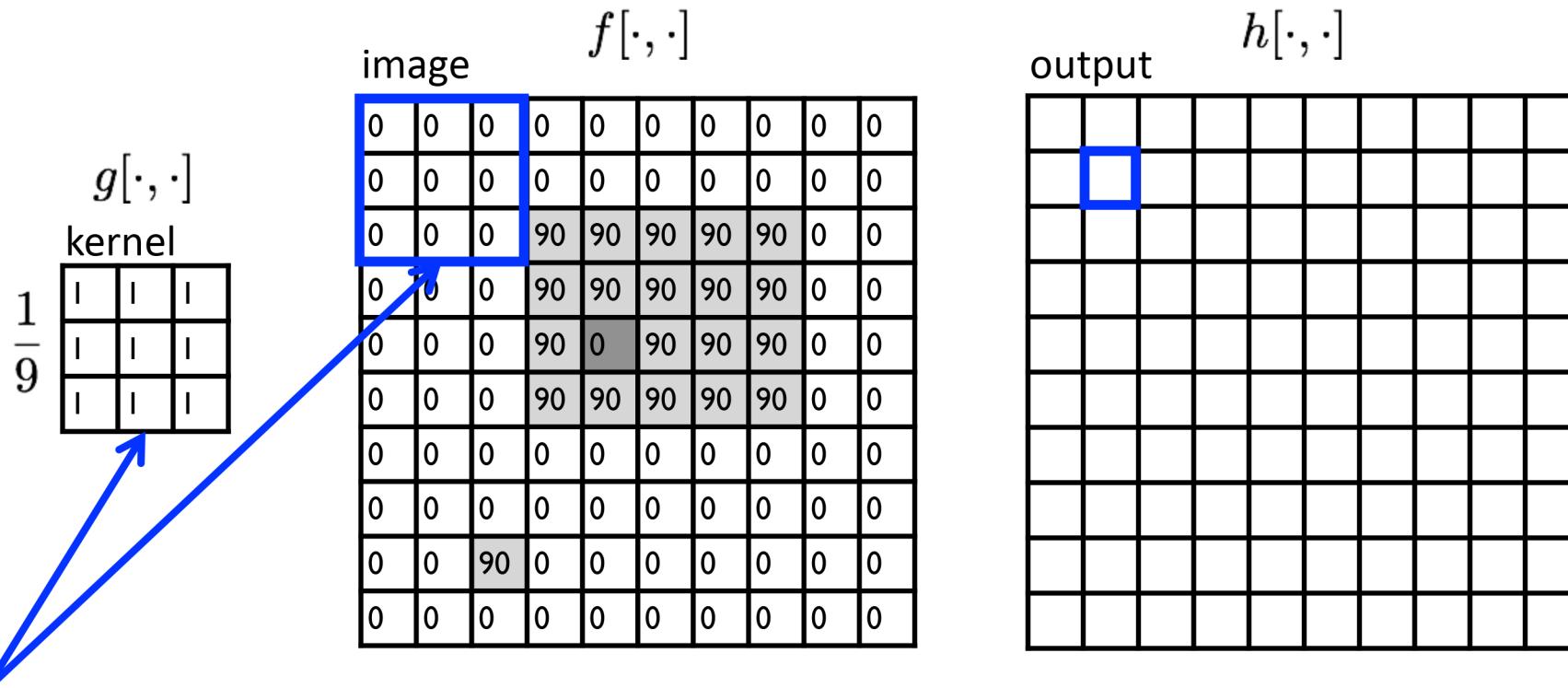


$F[x, y]$

$F$  is an array of numbers (pixels)

$x, y$  are integers (column/row indices)

# Let's run the box filter

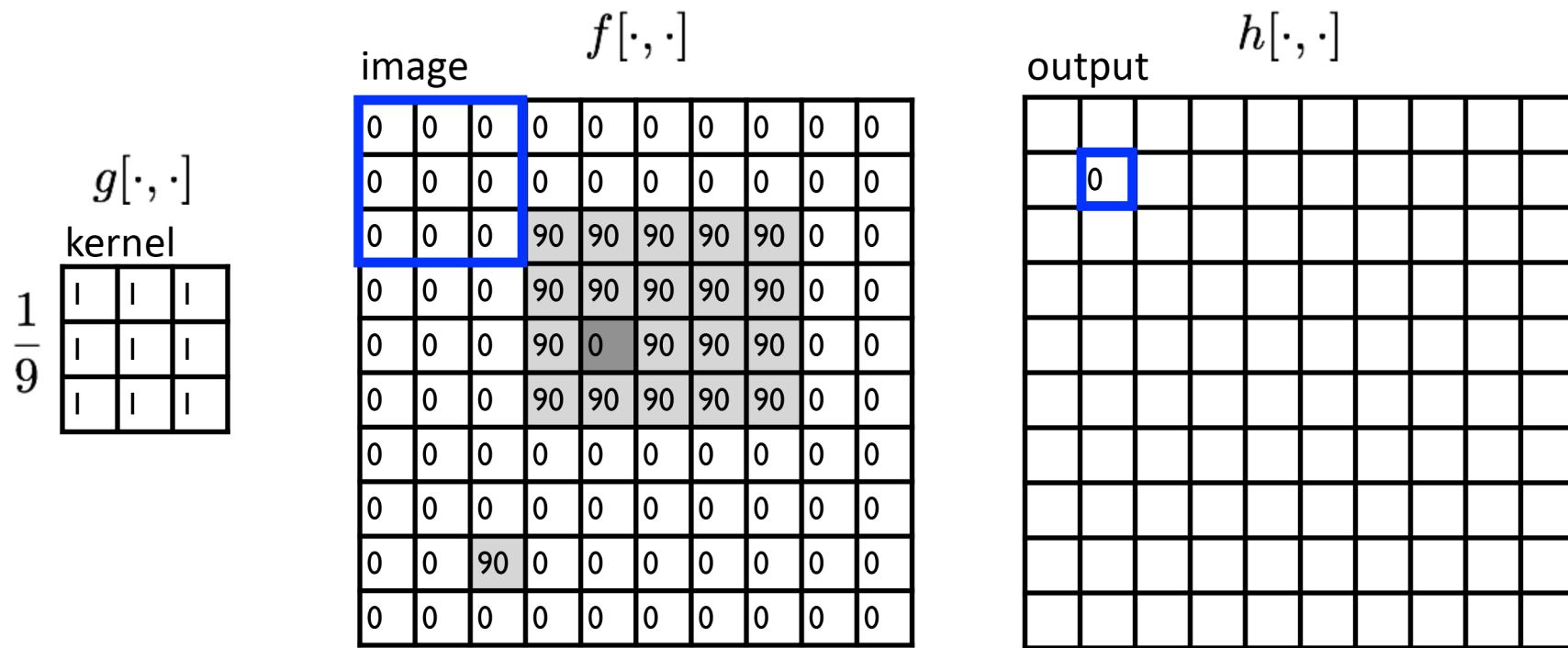


note that we assume that  
the kernel coordinates  
are centered

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

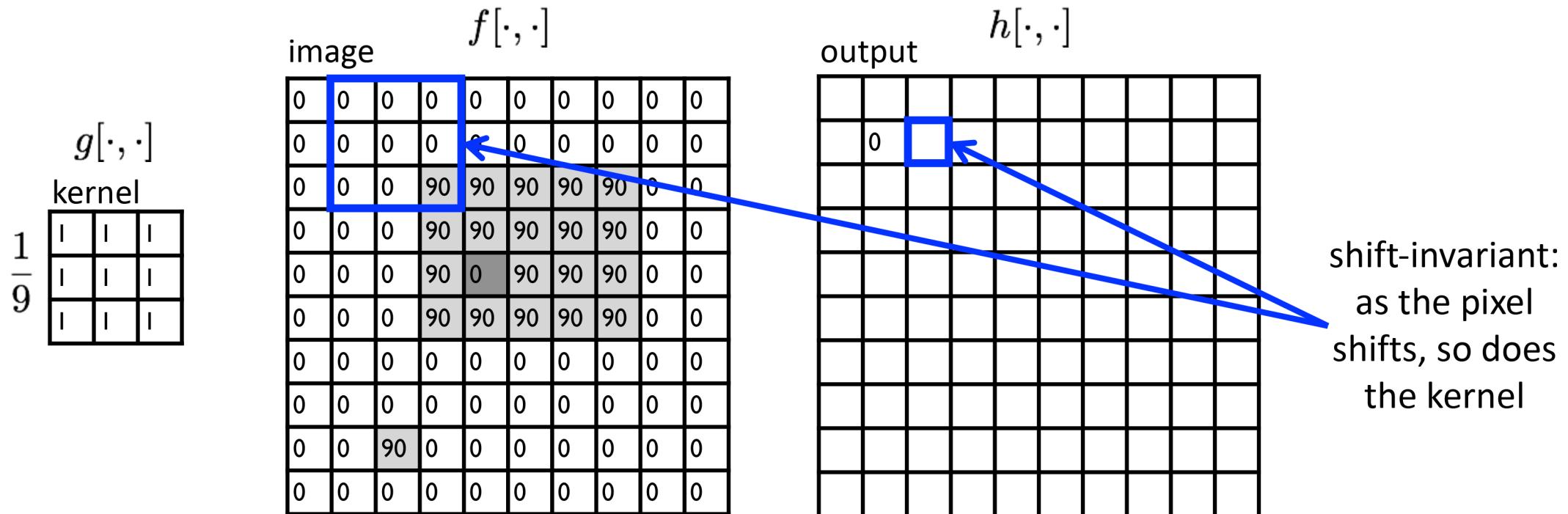
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

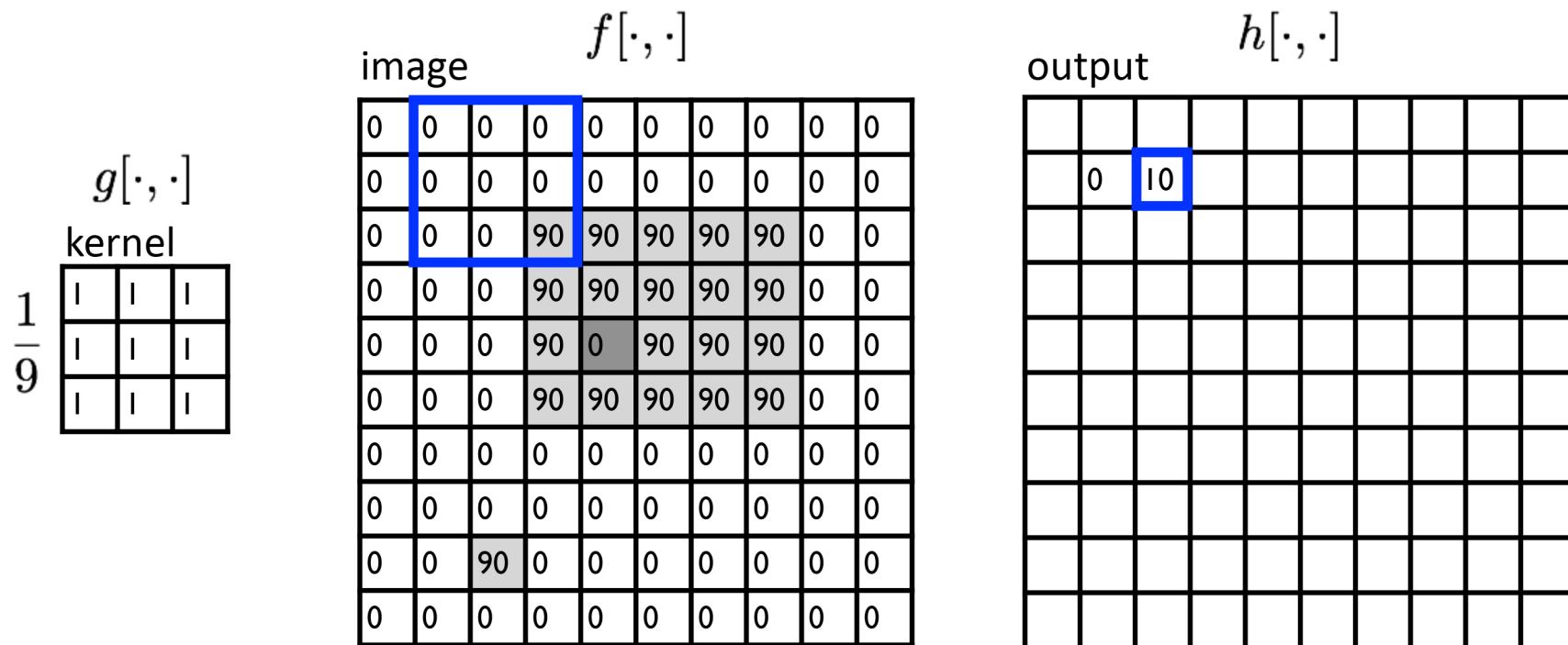
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

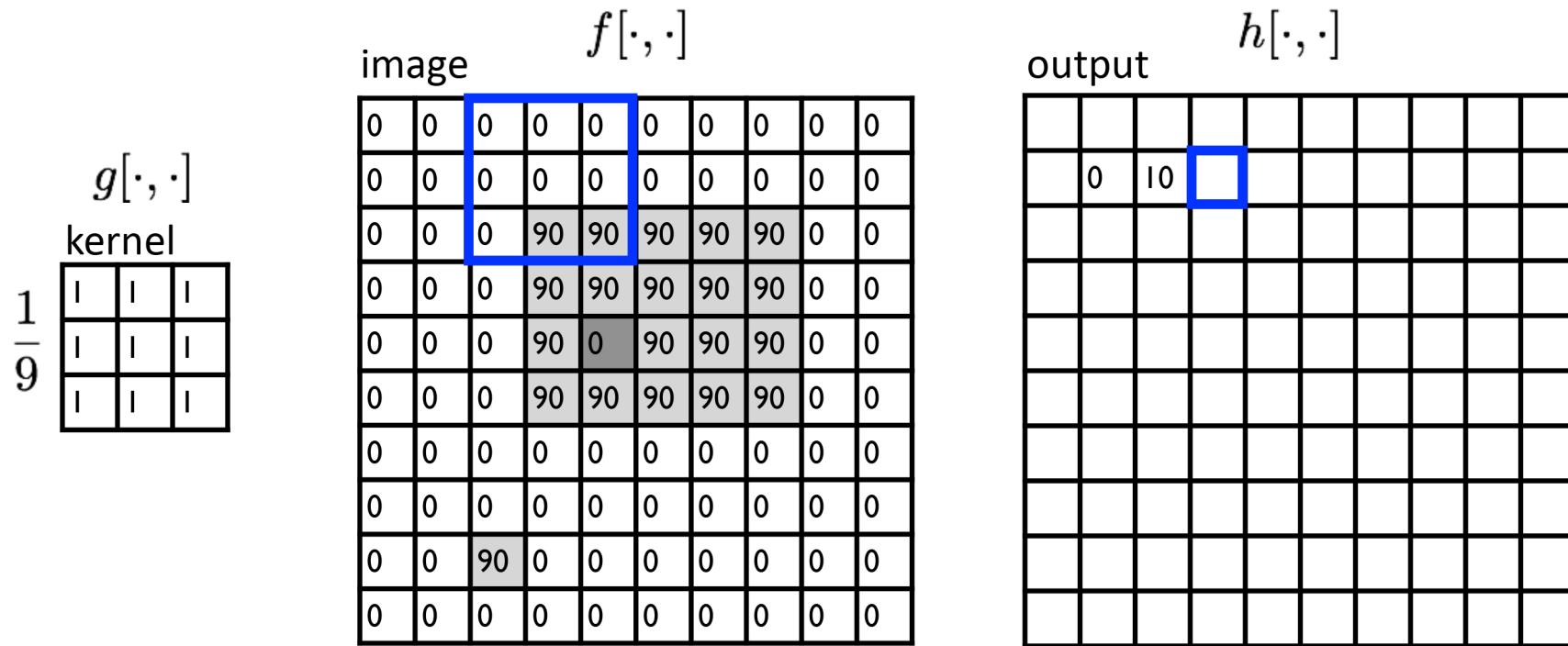
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

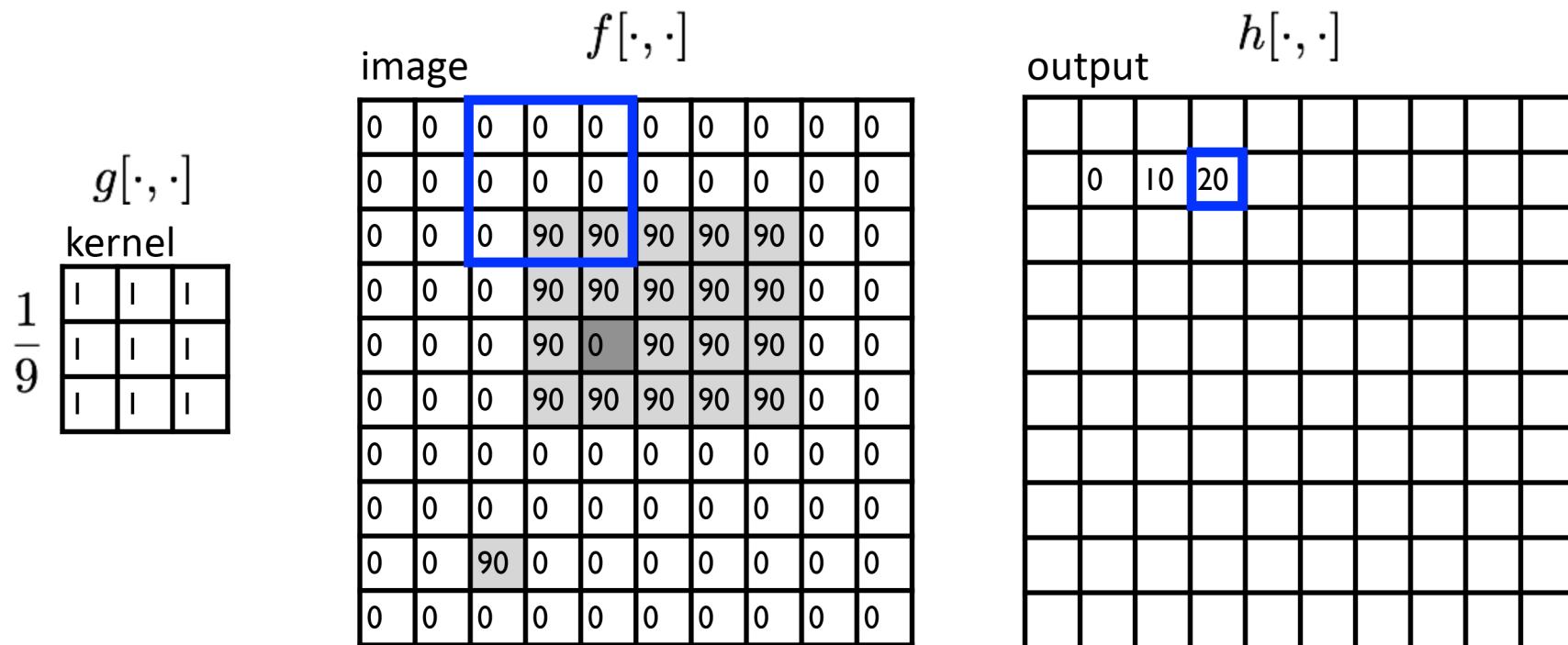
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output       $k, l$       filter      image (signal)

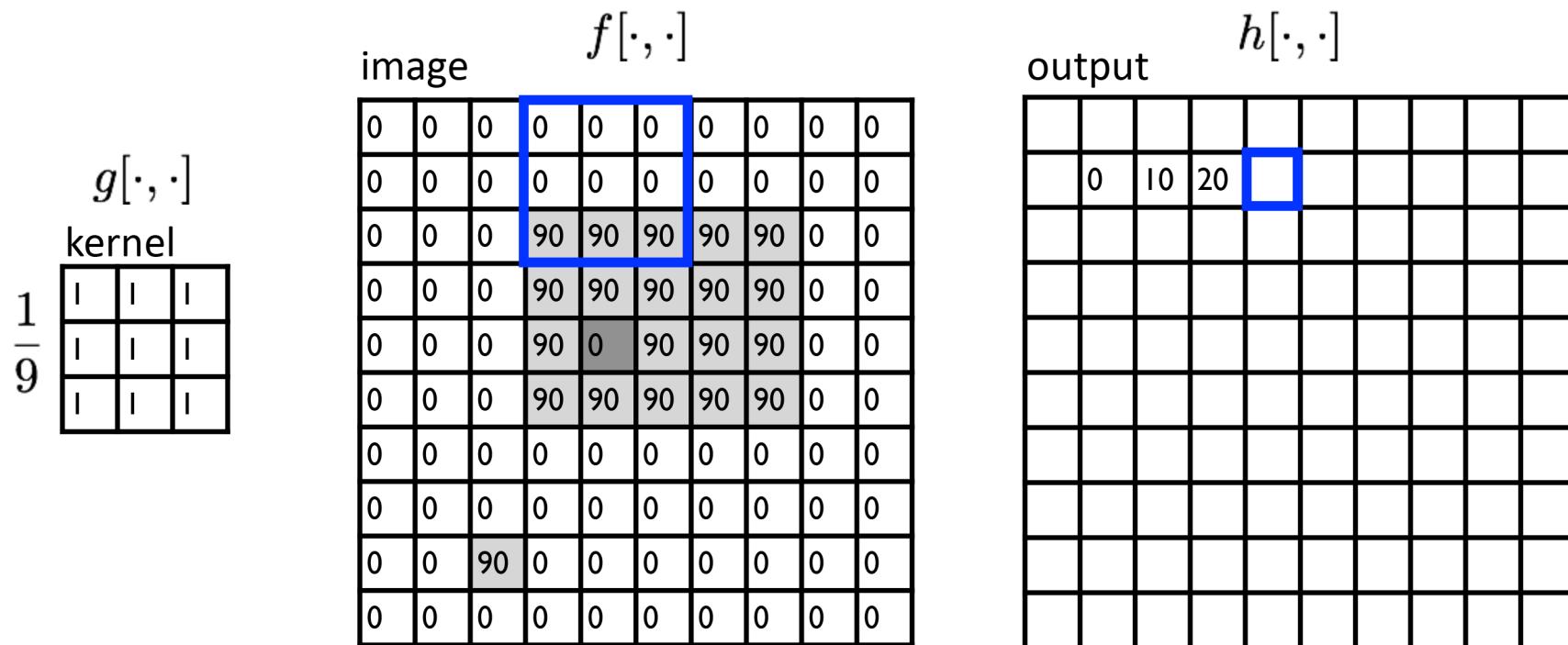
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

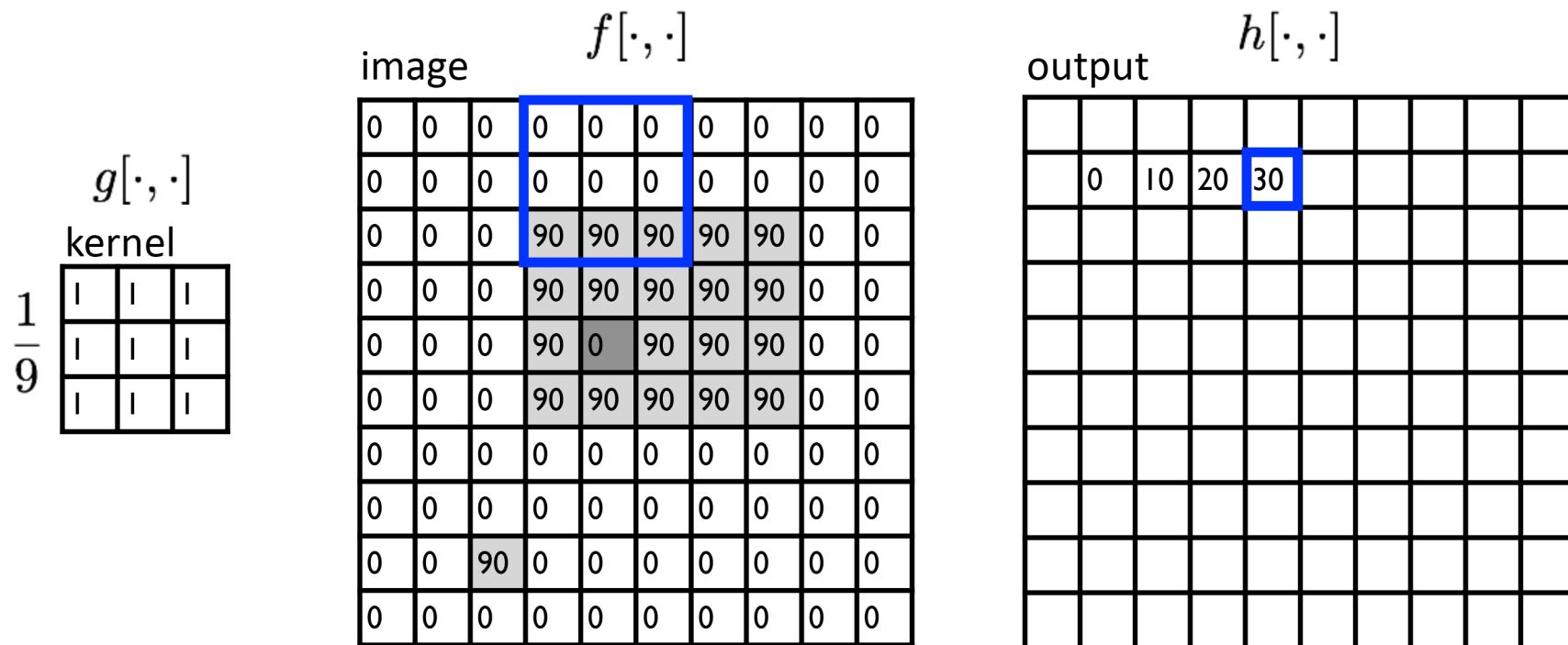
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

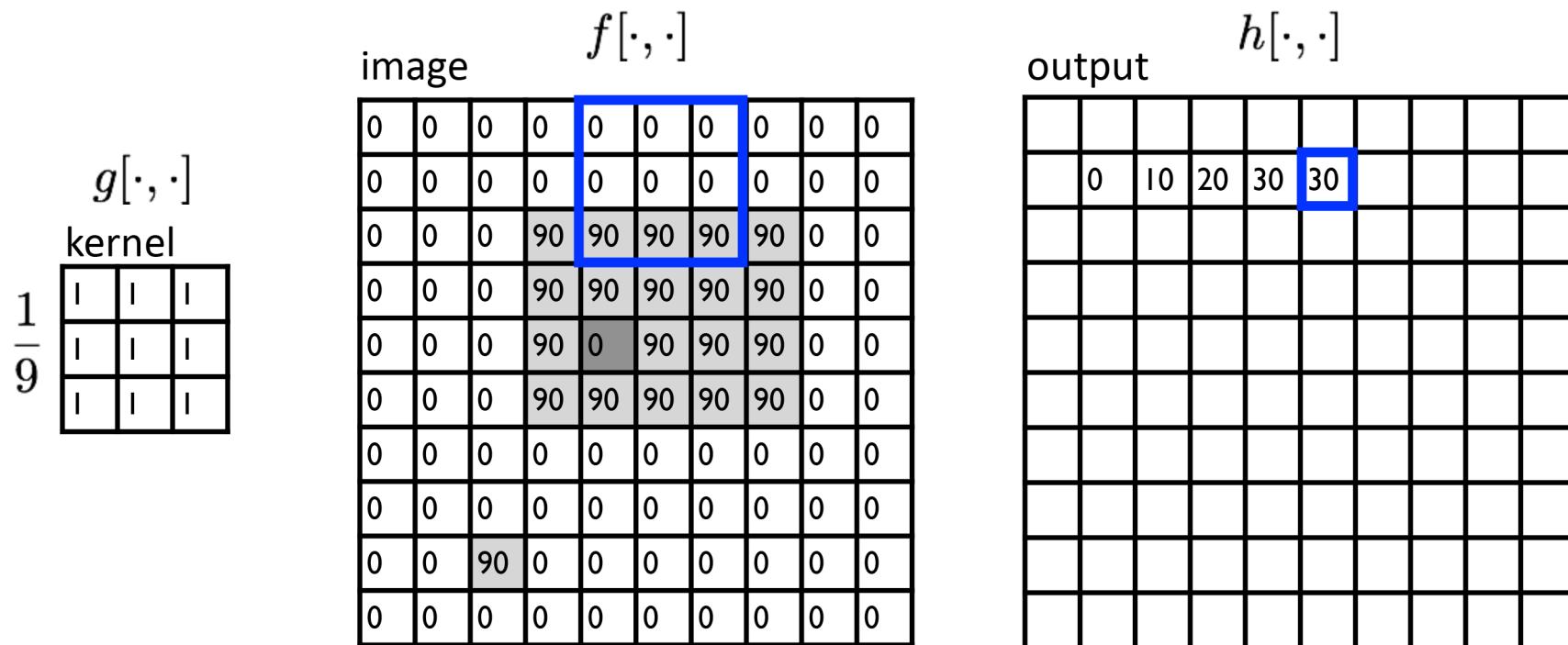
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

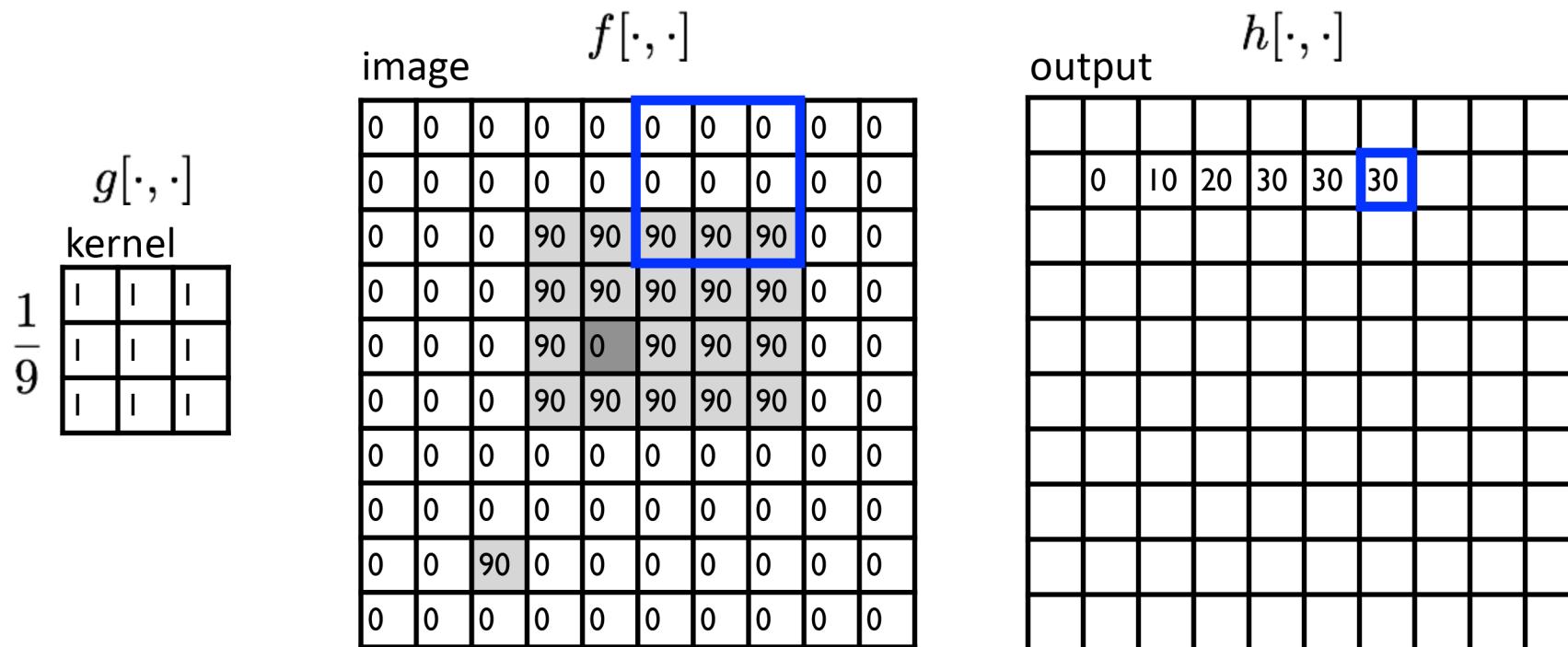
# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

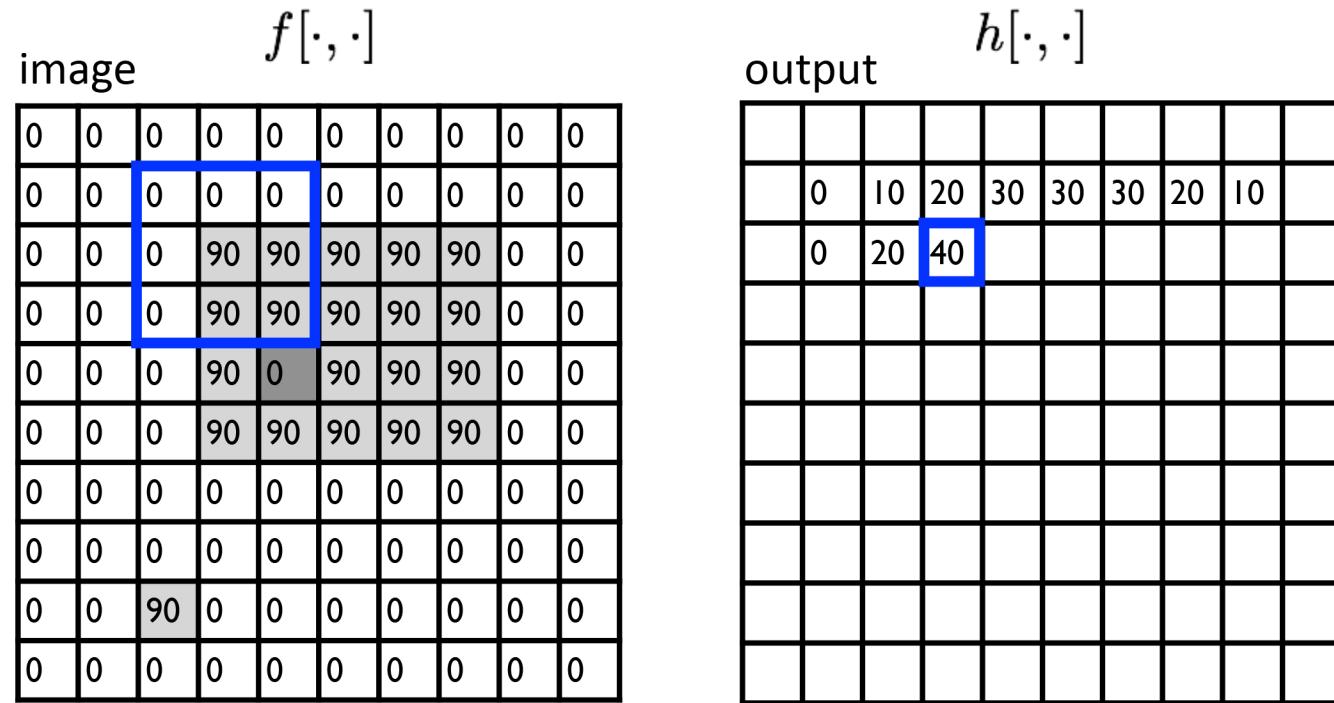
# Let's run the box filter

$$g[\cdot, \cdot]$$

kernel

$$\frac{1}{9}$$

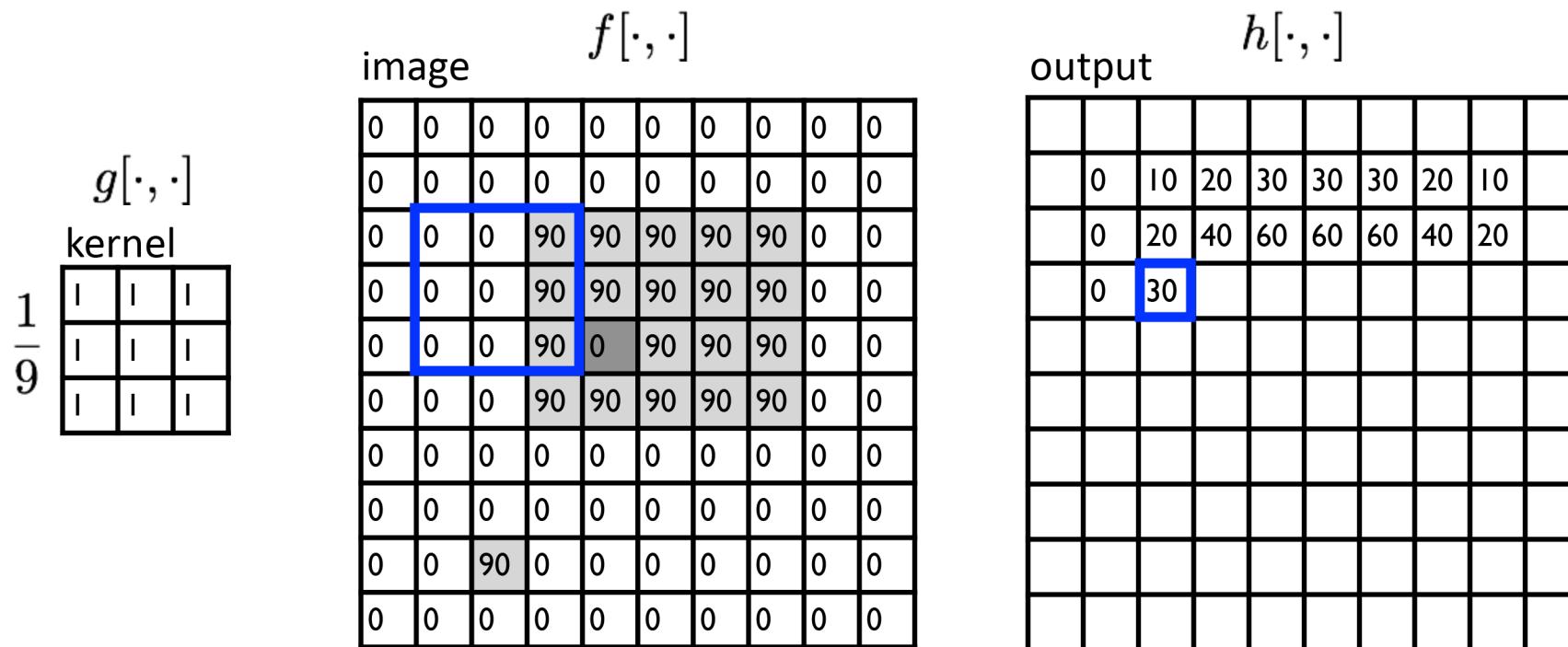
1	1	1
1	1	1
1	1	1



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# Let's run the box filter



$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# Let's run the box filter

$$g[\cdot, \cdot]$$

kernel

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

image	$f[\cdot, \cdot]$	output	$h[\cdot, \cdot]$
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 10 20 30 30 30 20 10	0 20 40 60 60 60 40 20
0 0 0 90 90 90 90 90 0 0	0 0 0 90 90 90 90 90 0 0	0 30 50 80 80 90 60 30	0 30 50 80 80 90 60 30
0 0 0 90 90 90 90 90 0 0	0 0 0 90 0 90 90 90 0 0	0 20 30 50 50 60 40 20	0 20 30 50 50 60 40 20
0 0 0 90 0 90 90 90 90 0 0	0 0 0 90 90 90 90 90 0 0	0 10 20 30 30 30 20 10	0 10 20 30 30 30 20 10
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	10 10 10 10 0 0 0 0	10 10 10 10 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	10	

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# Let's run the box filter

$$g[\cdot, \cdot]$$

kernel

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

image	$f[\cdot, \cdot]$	output	$h[\cdot, \cdot]$
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 10 20 30 30 30 20 10	0 20 40 60 60 60 40 20
0 0 0 90 90 90 90 90 0 0	0 0 0 90 90 90 90 90 0 0	0 30 50 80 80 90 60 30	0 30 50 80 80 90 60 30
0 0 0 90 90 90 90 90 0 0	0 0 0 90 0 90 90 90 0 0	0 20 30 50 50 60 40 20	0 10 20 30 30 30 20 10
0 0 0 90 0 90 90 90 90 0 0	0 0 0 90 90 90 90 90 0 0	10 10 10 10 0 0 0 0	10 10 10 10 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0	0 0 90 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

# ... and the result is

Diagram illustrating the convolution process:

Input image  $f[\cdot, \cdot]$ :

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Output  $h[\cdot, \cdot]$ :

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

$$h[m, n] = \sum_{k,l} g[k, l] f[m + k, n + l]$$

output      filter      image (signal)

What will be the result of the following convolution operation?



Input image



0	0	0
0	1	0
0	0	0

Filter  
(Kernel)

The following convolution operation produces an identical image.

Input image

\*

Filter  
(Kernel)

=

Identical image

0	0	0
0	1	0
0	0	0

What will be the result of the following convolution operation?



Input image



0	0	0
1	0	0
0	0	0

Filter  
(Kernel)

The following convolution operation **shifts the image left by one pixel**.

Input image

\*

Filter  
(Kernel)

=

Shifted left by 1 pixel

0	0	0
1	0	0
0	0	0

What will be the result of the following convolution operation?



$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

Filter  
(Kernel)

Input image

The following convolution operation **blurs the image**.

The diagram illustrates a convolution operation. On the left is a grayscale **Input image**, which is a blurry photograph of a person's face. In the center is a **Filter (Kernel)**, represented by a 3x3 grid of orange squares, each containing the value 1. To the left of the filter is a convolution symbol ( $*$ ). To the right of the filter is an equals sign (=). To the right of the equals sign is a grayscale **Blurred image**, which is a much smoother version of the input image, showing a more uniform gray where the original features were blurred.

$$\text{Input image} \quad * \quad \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \text{Blurred image}$$

Filter  
(Kernel)

What will be the result of the following convolution operation?


$$\text{Input image} \quad * \left( \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \right) - \frac{1}{9} \left( \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \right)$$

Filter  
(Kernel)

The following convolution operation **sharpens** the image.

The diagram illustrates a convolution operation for image sharpening. It shows the input image, the filter kernel, the scaling factor, the subtraction term, and the final output image.

Input image:



Filter kernel:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Scaling factor:

$$\text{Scale up the intensity}$$

Subtraction term:

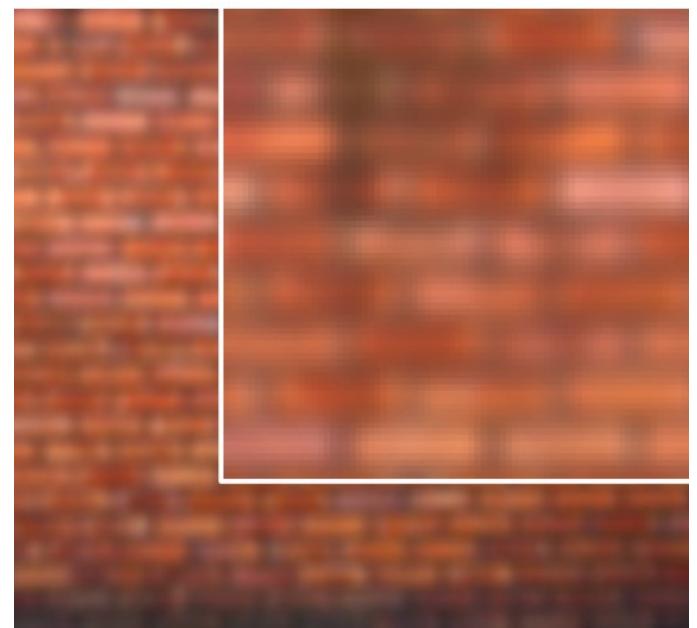
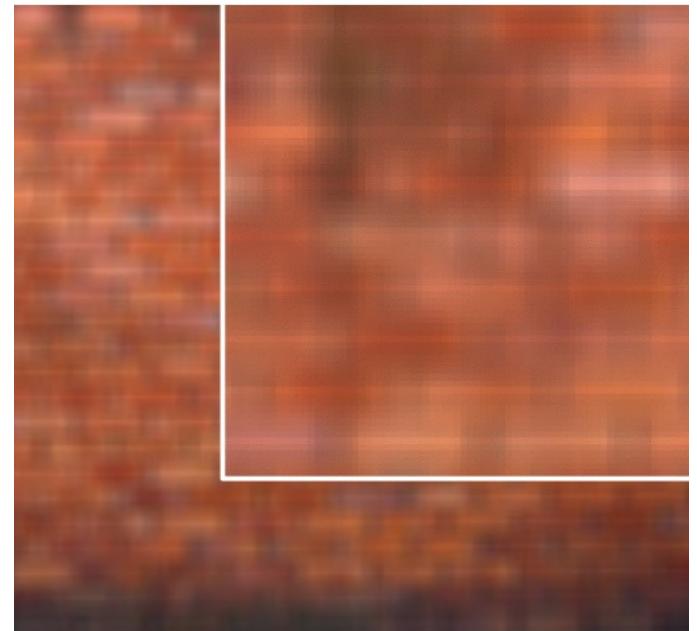
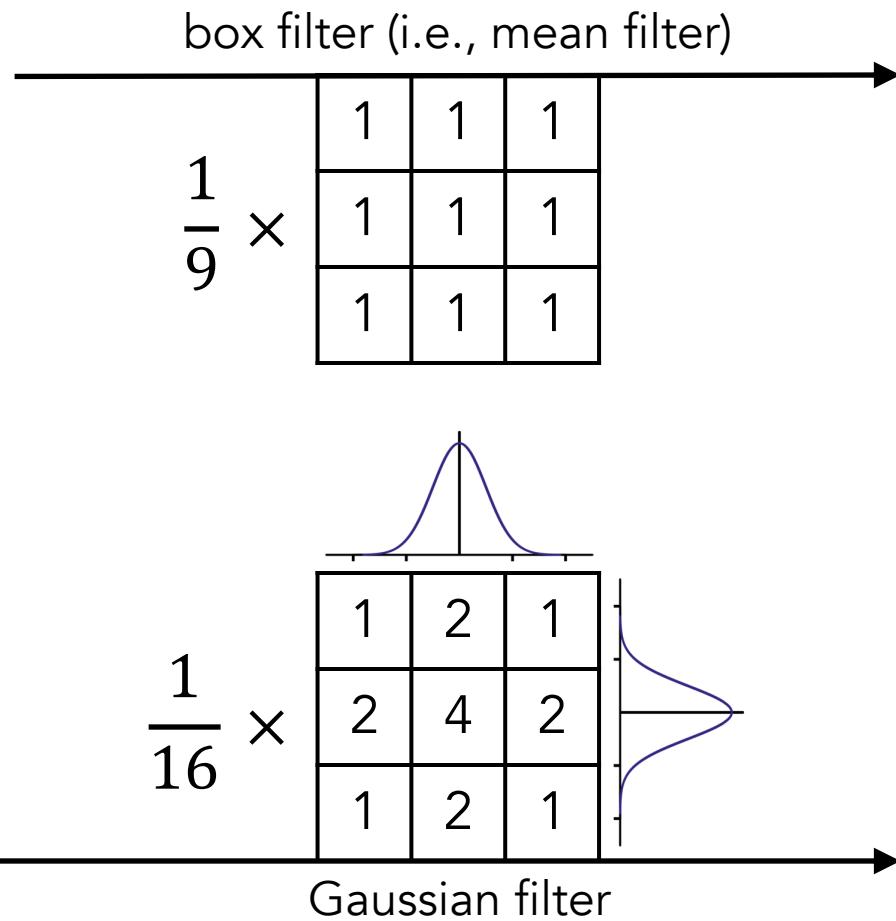
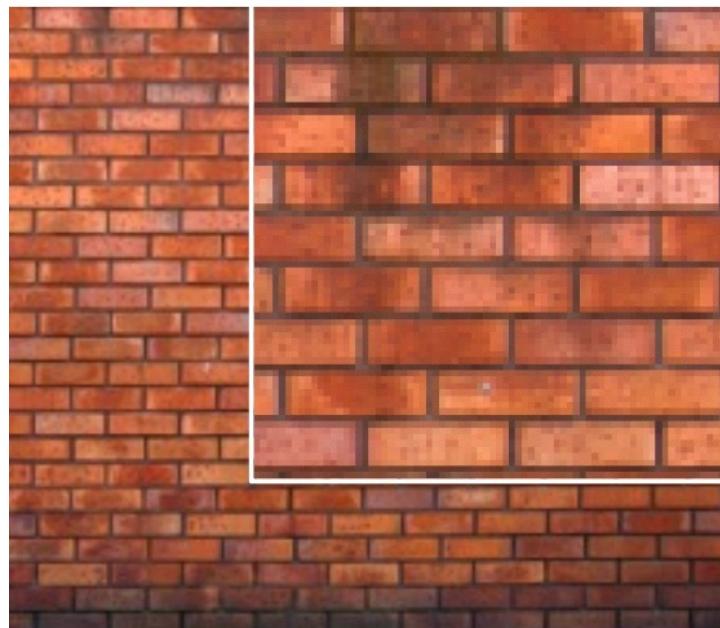
$$-\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Remove blurred signals

Final output:



We can use image filters to [blur images](#), such as using the box filter and the Gaussian filter (2D Gaussian distribution).

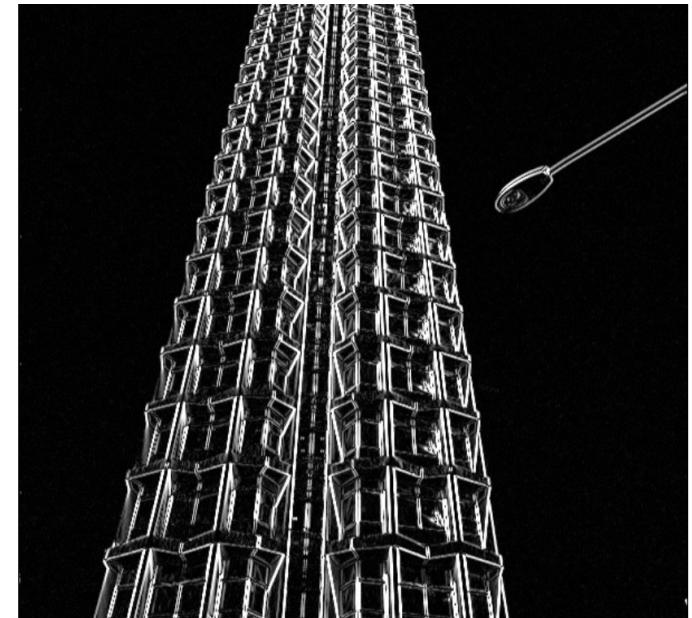


We can use image filters to **detect edges**, such as using the Sobel filter to compute the derivative of signals.



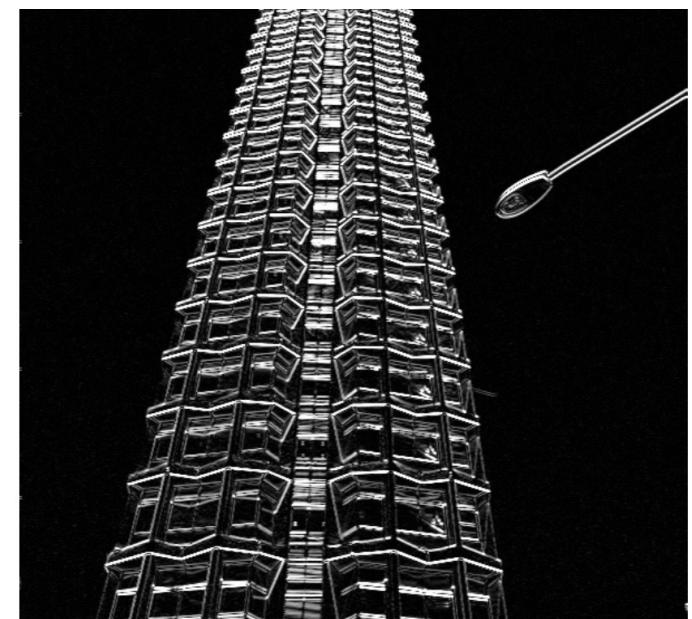
Horizontal Sobel filter →

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



Vertical Sobel filter →

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

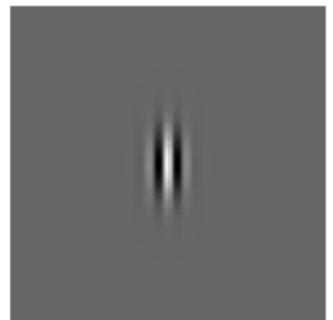
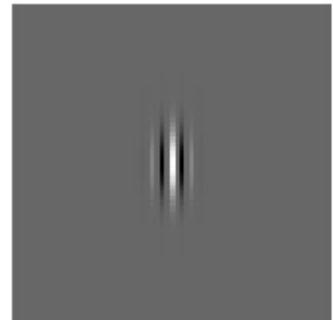


We can use various hand-crafted convolutional kernels (i.e., filter banks) to extract different kinds of information in the image, such as using the Gabor filters.



Input image

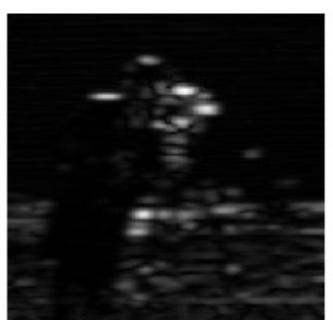
\*



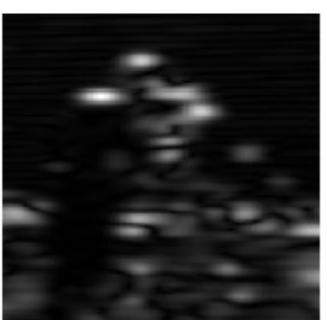
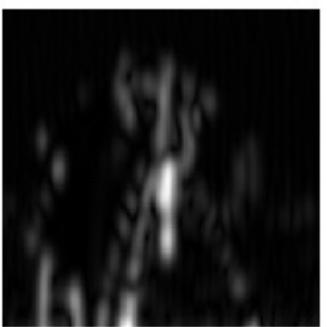
=



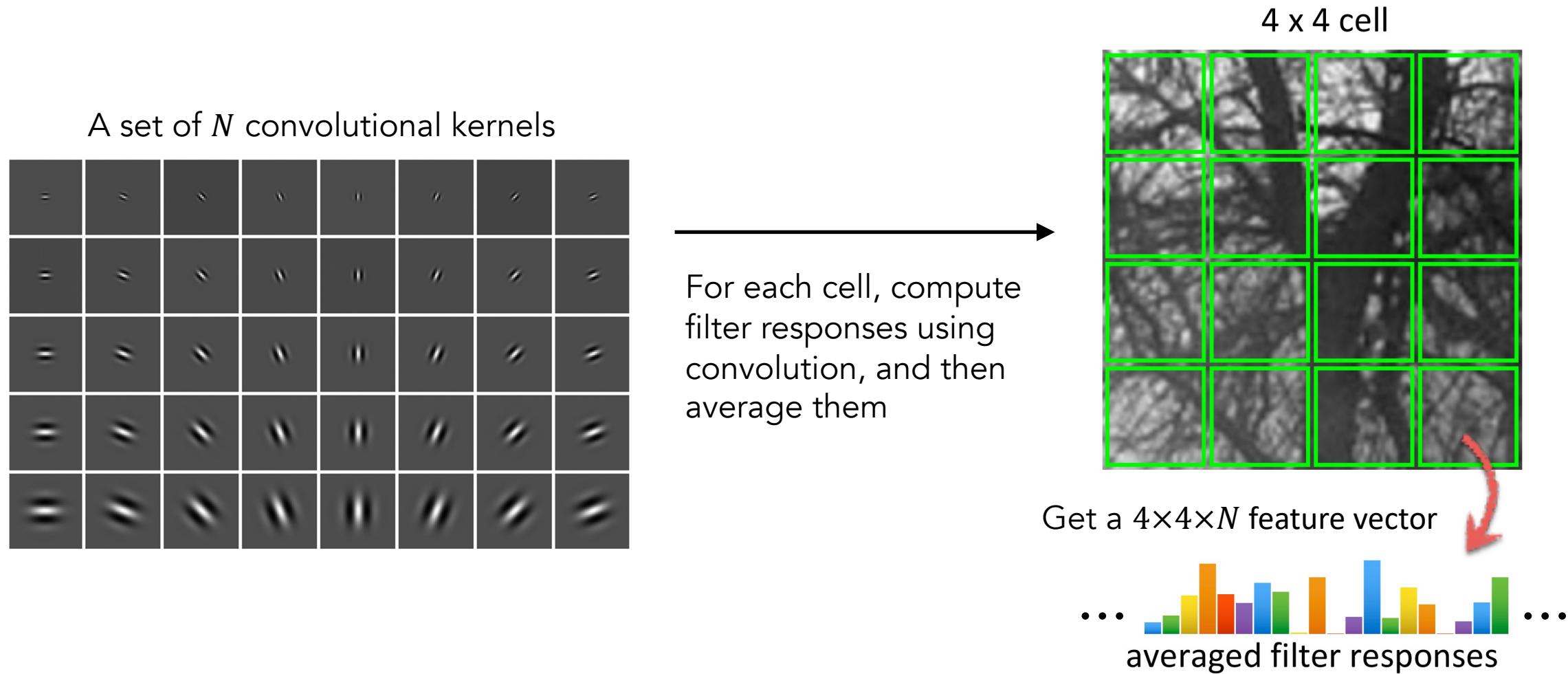
A partial set of Gabor filters



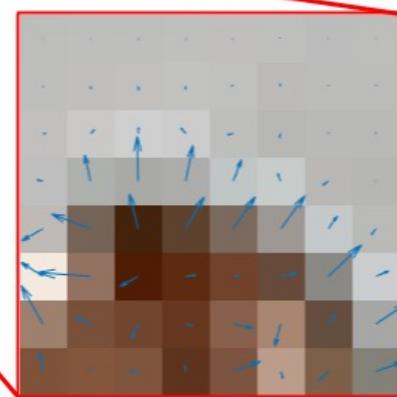
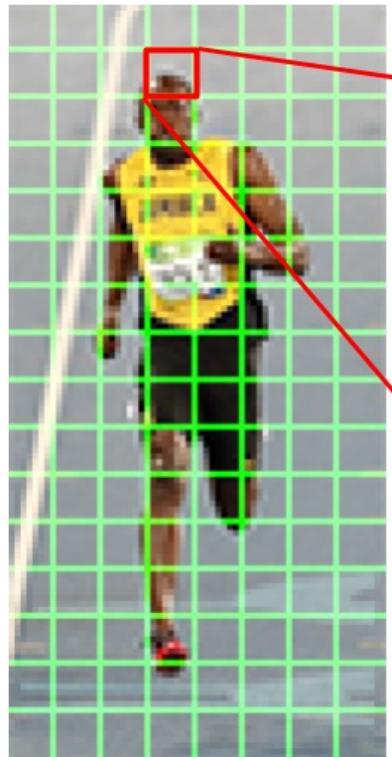
Output images



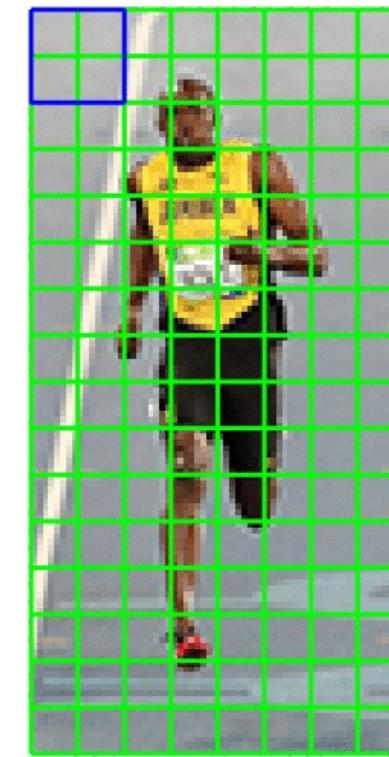
To construct features, we can perform convolution on image patches using a set of kernels (i.e., filter banks) and then aggregate them into a feature vector.



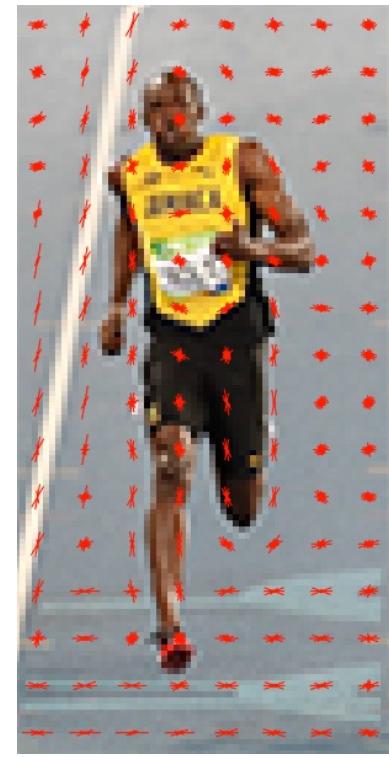
There are many ways of constructing feature vectors, such as Histogram of Gradients.



Compute gradients using Sobel filter



Aggregate gradients into histograms



# Take-Away Messages

- Computers represent images as tensors with numbers in several channels (e.g., red, green, and blue).
- The origin (i.e., index [0,0]) of the image coordinate is at the top-left corner of the image.
- Convolution (the discrete version) means sliding a kernel/filter over the image to compute the sum of element-wise multiplication, which can also be seen as a dot product.
- We can use convolution to perform many different image-filtering operations, such as shifting pixels, blurring/sharpening images, and detecting edges.
- We can use a filter bank (i.e., a collection of kernels) to perform convolution on image patches, and the result (i.e., a set of filter responses) can be used as features.



# Questions?