https://introml.mit.edu/

# 6.390 Intro to Machine Learning

## Lecture 7: Convolutional Neural Networks

Shen Shen

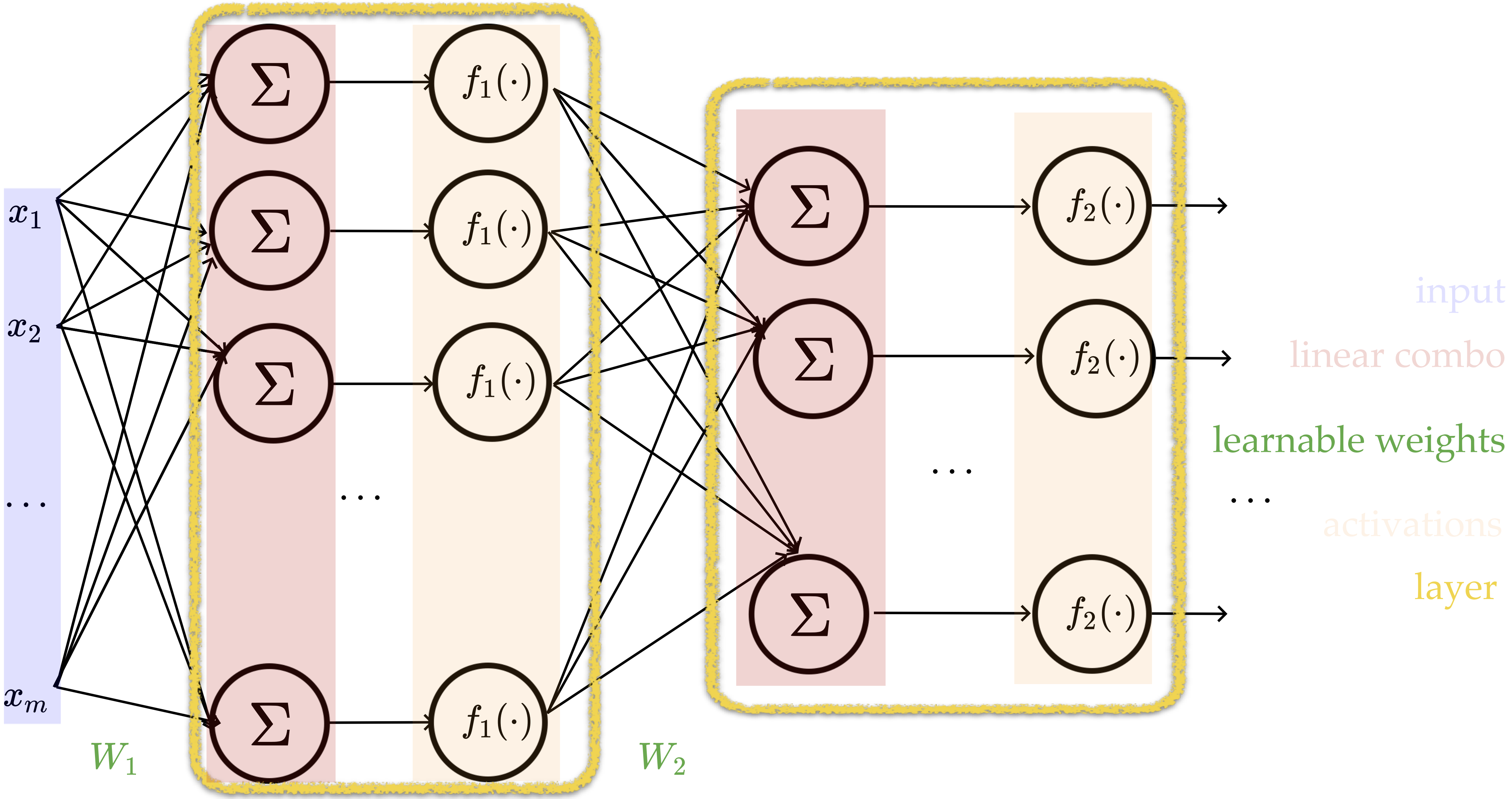
March 22, 2024

(videos edited from 3b1b; some slides adapted from Phillip Isola and Kaiming He)

# Outline

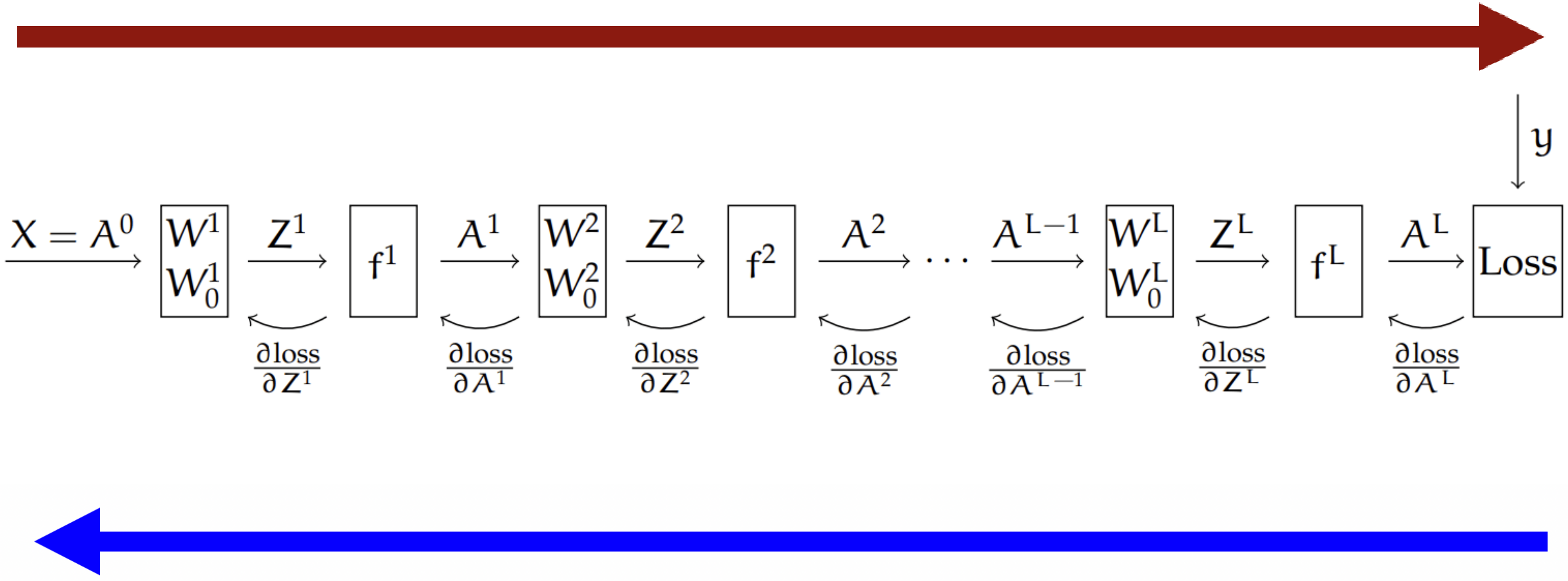
- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary

# A (feed-forward) neural network is



# Recap: Backpropogation

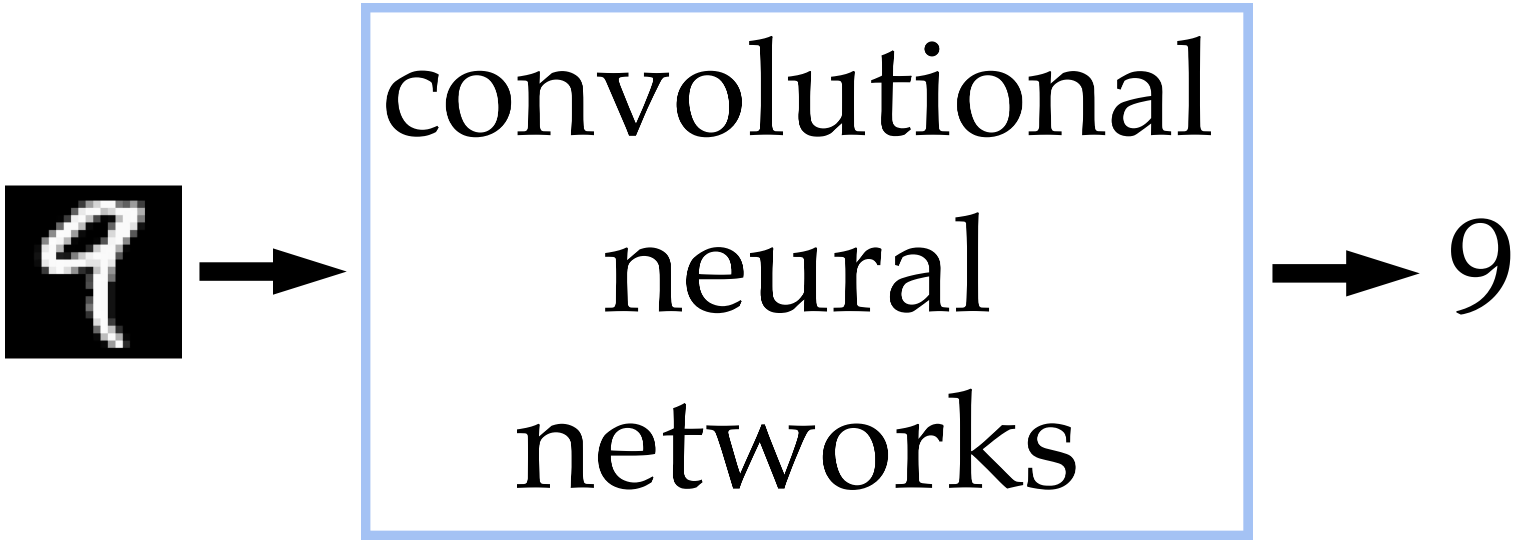
Forward propagation to obtain the output (model's guess)




Backpropagation to obtain gradients with respect to the loss

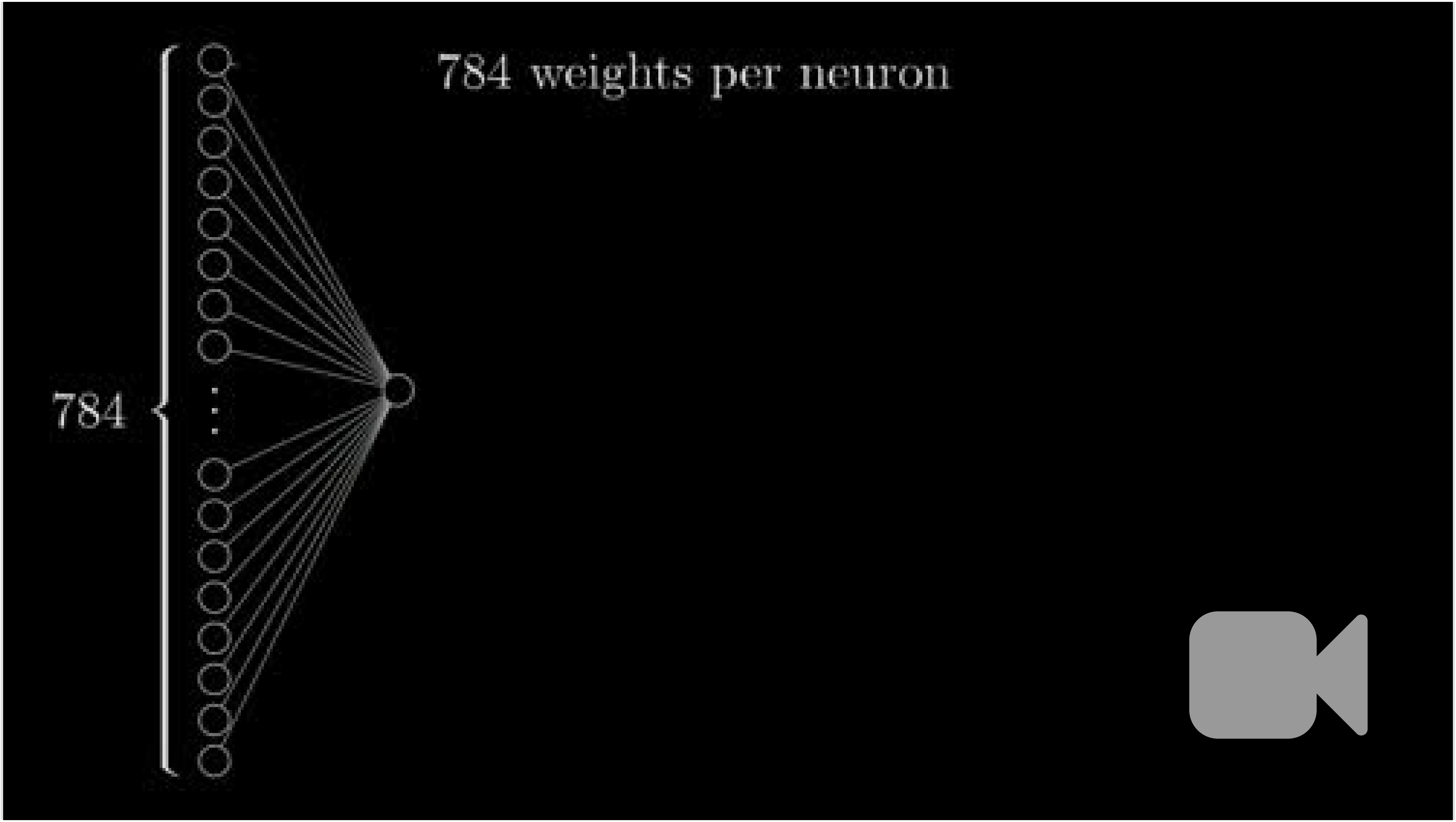
# Outline

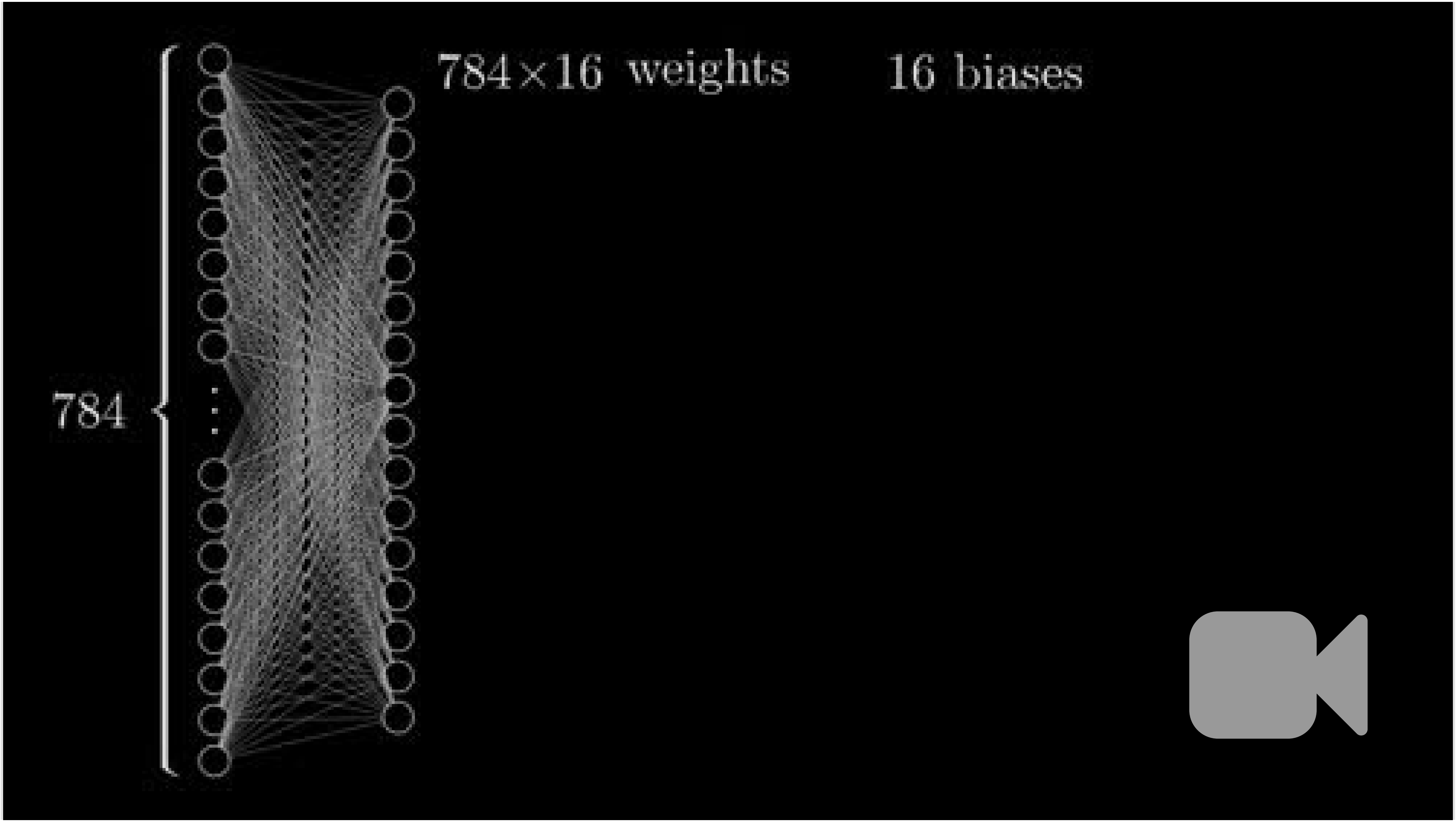
- Recap (fully-connected net)
- $\boxed{\text{Motivation and big picture ideas of CNN}}$
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary

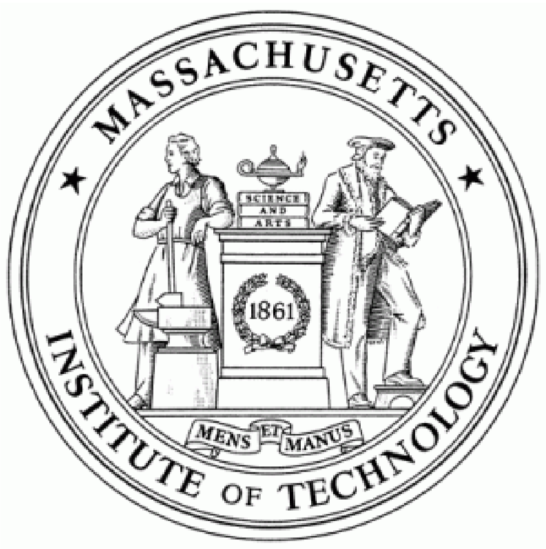



1. Why do we need a special network for images?
2. Why is CNN (the) special network for images?

Why do we
need a special net for images?


784 weights per neuron
784


784×16 weights
16 biases
784




👈 426-by-426 grayscale image

Use the same small 2-layer network?

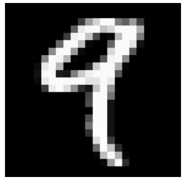
need to learn ~3M parameters

Imagine even higher-resolution images, or more complex tasks...

**Q:** Why do we need a specialized network?

**A:** fully-connected nets don't scale well to (interesting) images

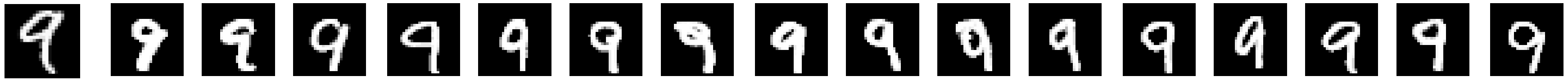
Why do we think



is

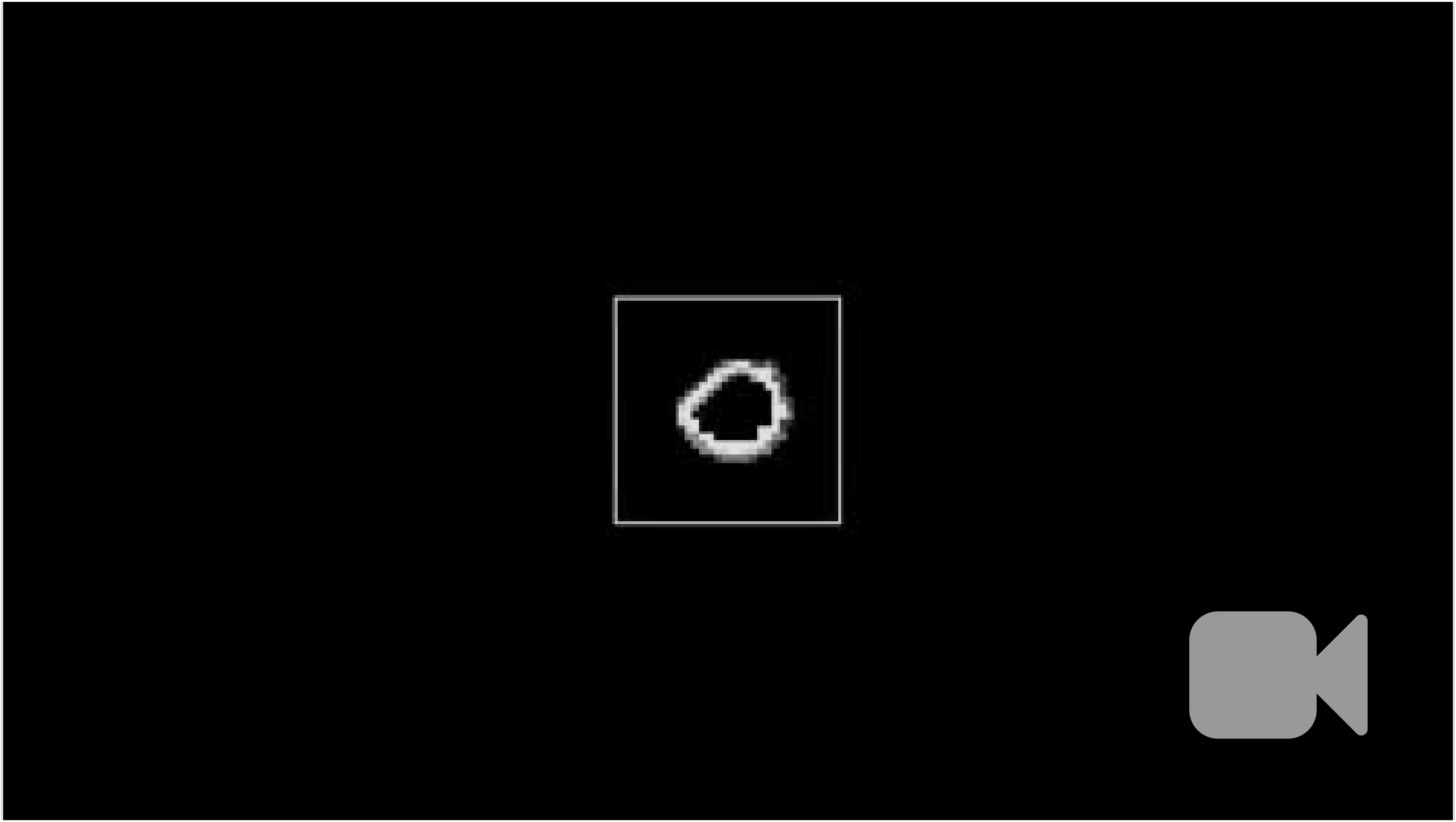
9?

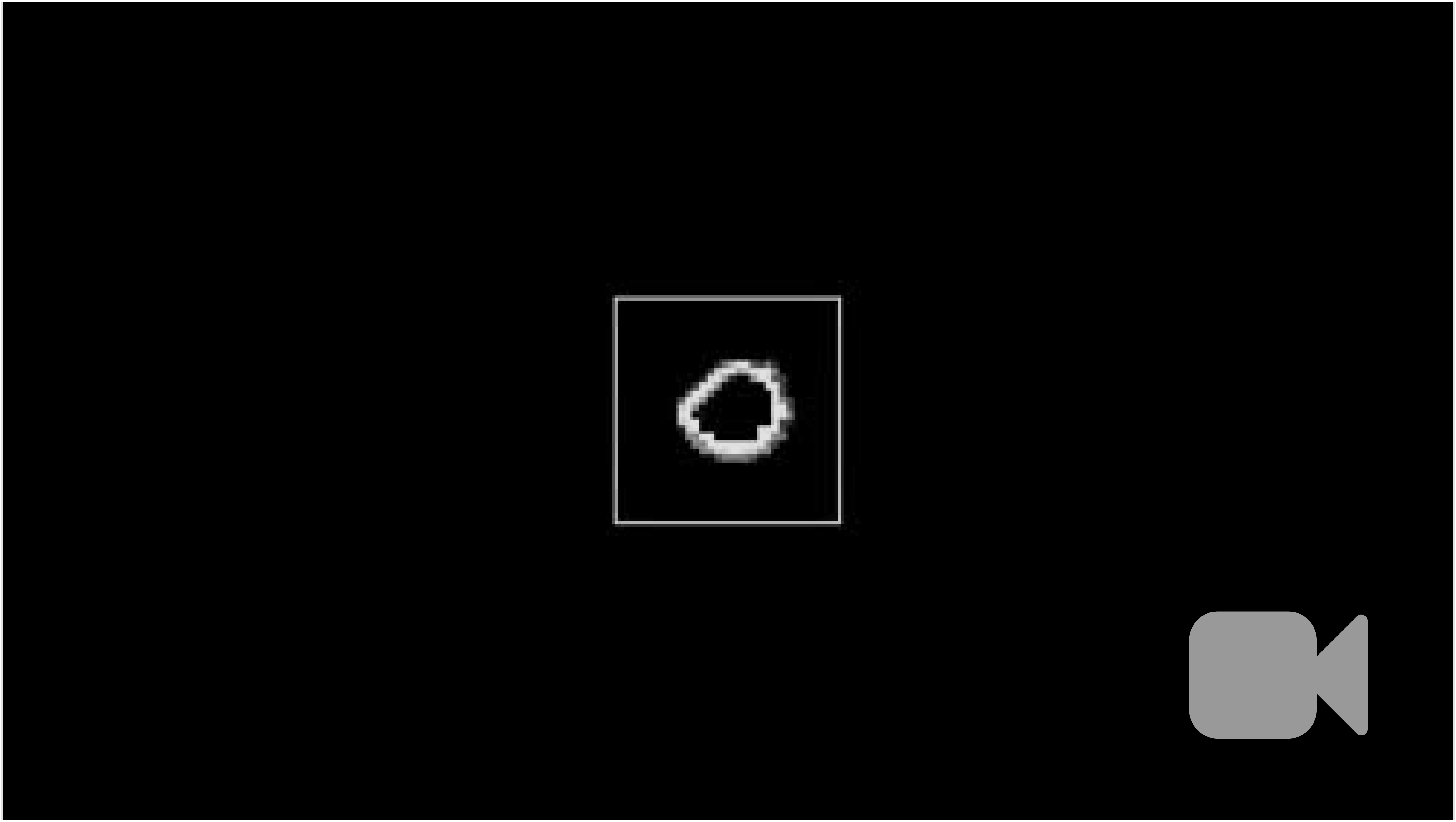
Why do we think any of



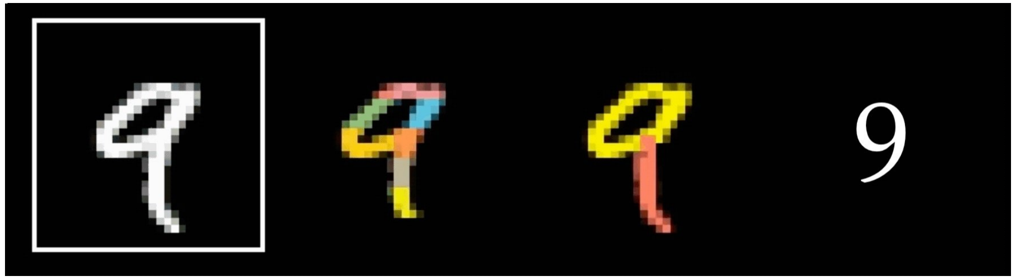
is a

9?

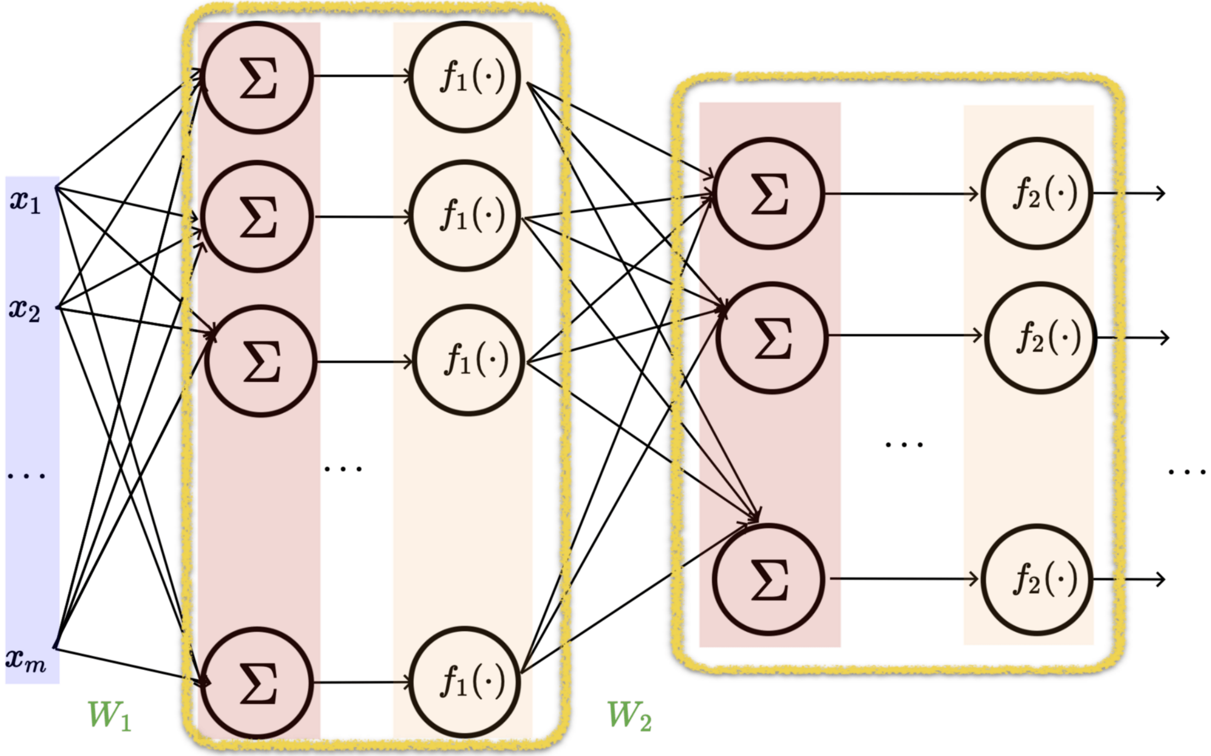




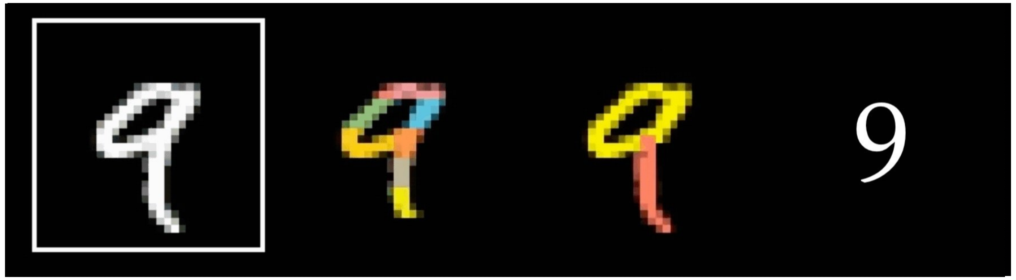
- Visual hierarchy



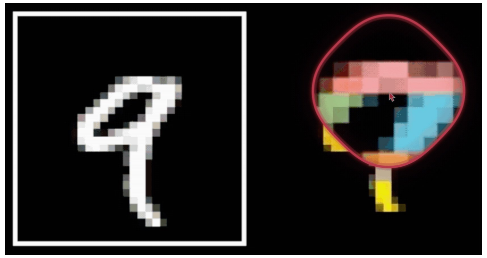

layering would help take care of that



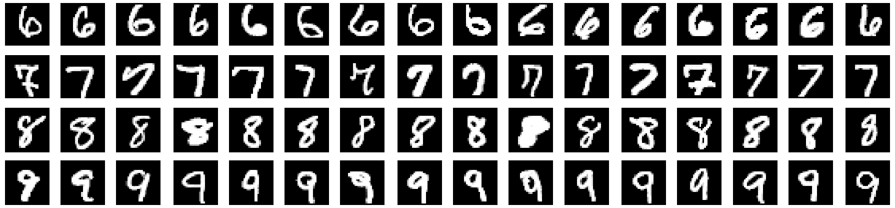
- Visual hierarchy




- Spatial locality



- Translational invariance



CNN cleverly exploits

- Visual hierarchy
- Spatial locality
- Translational invariance

via

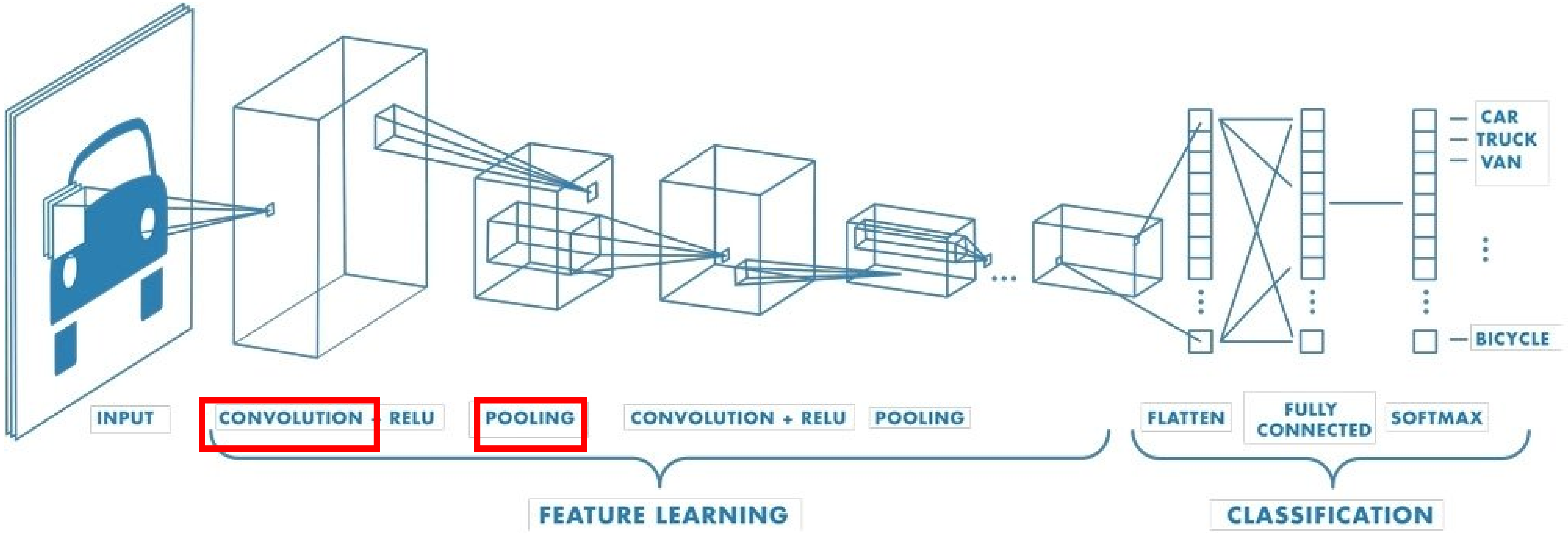
- layering (with nonlinear activations)
- convolution
- pooling

to handle images efficiently

cleverly exploits

via

to handle efficiently


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

# Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary

# Convolutional layer might sound foreign, but...

| Layer | Forward-pass {do} | Back-prop {learn} |
|---|---|---|
| Fully-connected | Dot-product | Neurons |
| Convolutional | Convolution | Filters (kernels) |

input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

filter

| | |
|---|---|
| -1 | 1 |

$$(0 * -1) + (1 * 1) = 1$$

output image

| | | | |
|---|---|---|---|
| 1 | | | |

input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

filter

| | |
|---|---|
| -1 | 1 |

$$(1 * -1) + (0 * 1) = -1$$

output image

| | | | |
|---|---|---|---|
| 1 | -1 | | |

input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

filter

| | |
|---|---|
| -1 | 1 |

$$(0 * -1) + (1 * 1) = 1$$

output image

| | | | |
|---|---|---|---|
| 1 | -1 | 1 | |

input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

filter

| | |
|---|---|
| -1 | 1 |

$$(1 * -1) + (1 * 1) = 0$$

output image

| | | | |
|---|---|---|---|
| 1 | -1 | 1 | 0 |

input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | -1 | 1 | 1 |

filter

| | |
|---|---|
| -1 | 1 |

$(0 * -1) + (1 * 1) = 1$

$(-1 * -1) + (1 * 1) = 2$

output image

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

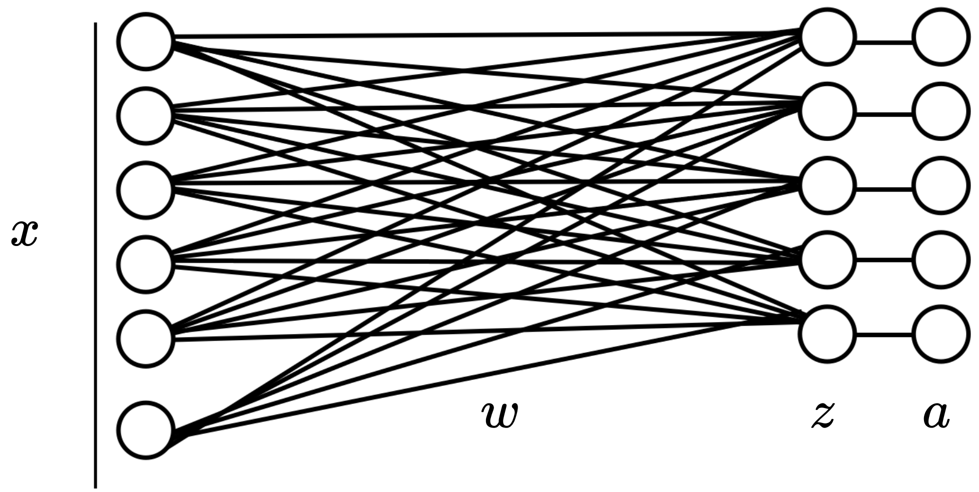
- 'look' locally
- parameter sharing
- "template" matching

- 'look' locally

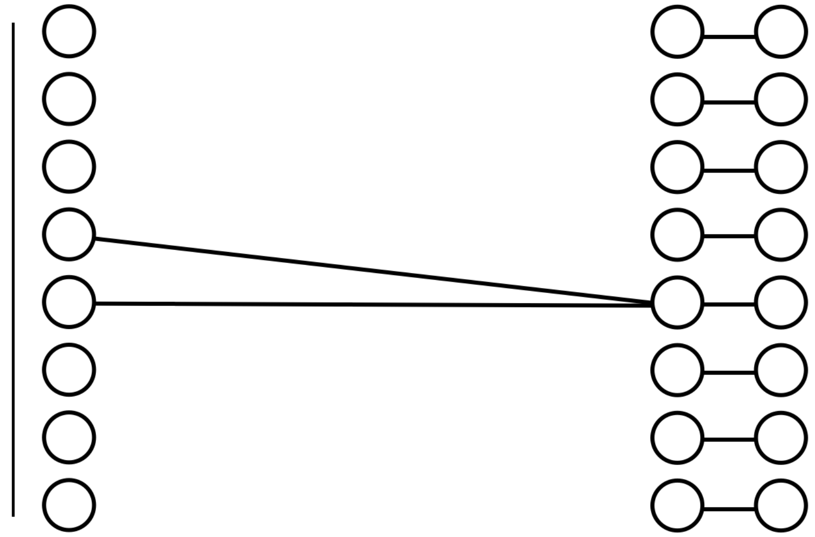
| input image | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | -1 | 1 | 1 |

| filter | | |
|---|---|---|
| | -1 | 1 |

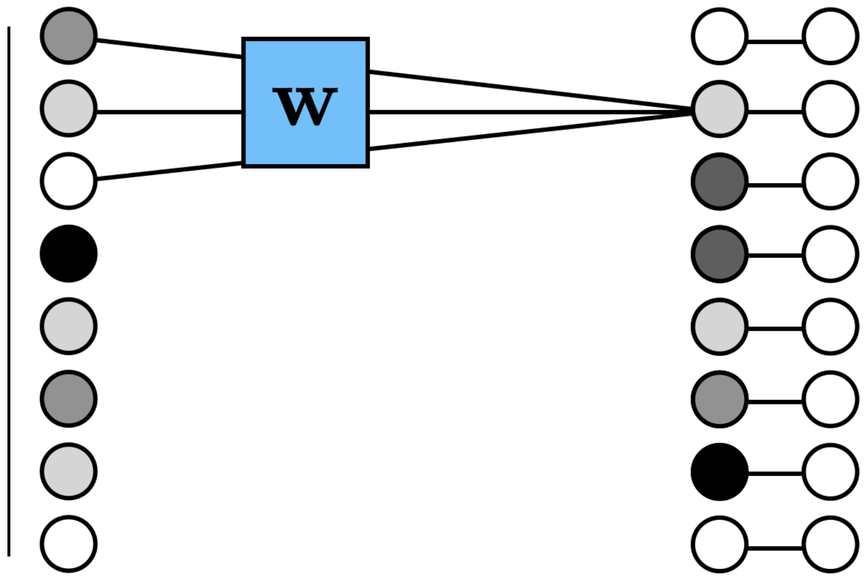
| output image | | | | |
|---|---|---|---|---|
| | 1 | -1 | 2 | 0 |

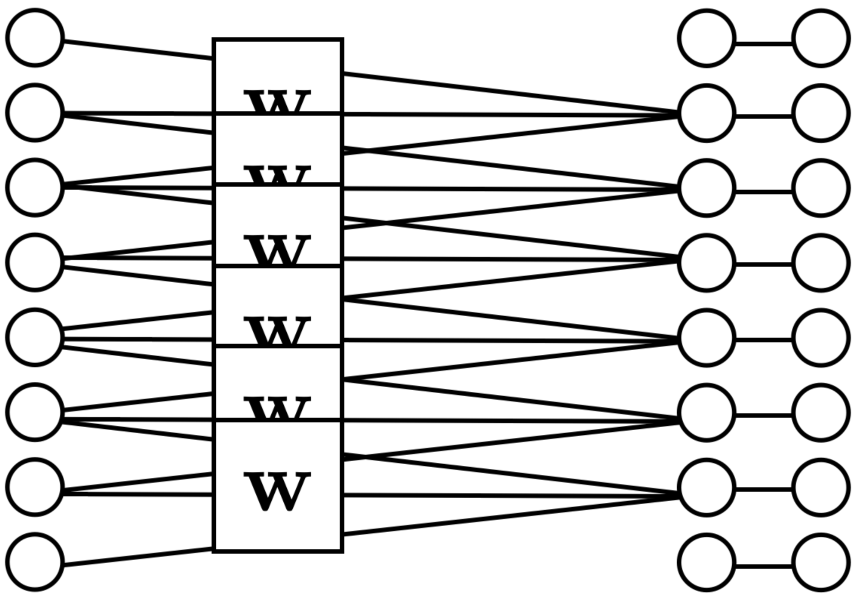



fully-connected layer



- parameter sharing



- parameter sharing

$$\begin{array}{|c|c|c|c|c|}\hline 0 & 1 & 0 & 1 & 1 \\ \hline \end{array}$$

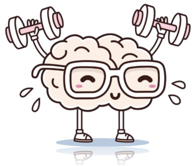
convolve with $\begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array}$

or dot with

$$\begin{array}{|c|c|c|c|}\hline -1 & 0 & 0 & 0 \\ \hline 1 & -1 & 0 & 0 \\ \hline 0 & 1 & -1 & 0 \\ \hline 0 & 0 & 1 & -1 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array}$$

=

$$\begin{array}{|c|c|c|c|}\hline 1 & -1 & 1 & 0 \\ \hline \end{array}$$



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

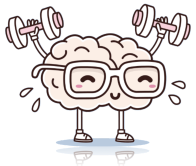
convolve with ?

=

dot-product with ?

=

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |



| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

convolve with

| |
|---|
| 1 |

dot-product with

$$I_{5\times 5}$$

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |

# Convolution: a 2-D example

input

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

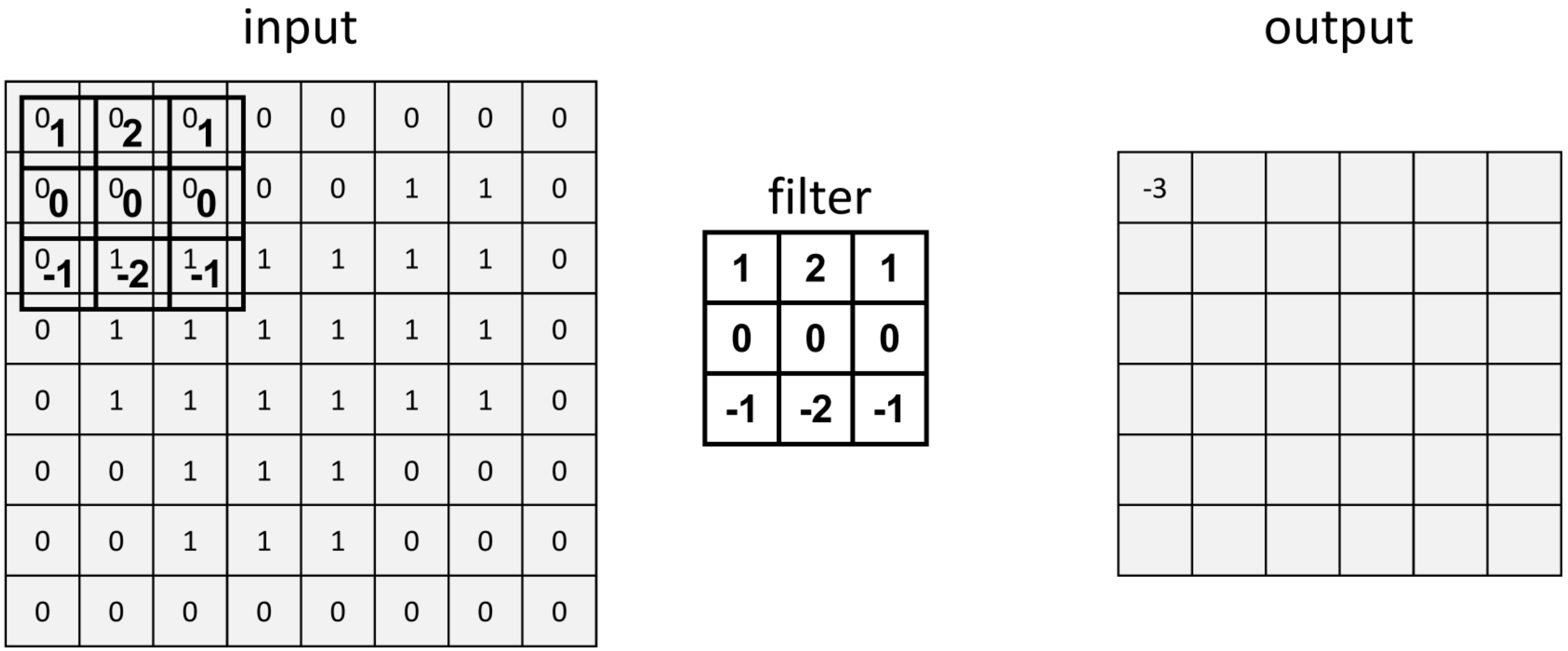
filter

| | | |
|---|---|---|
| **1** | **2** | **1** |
| **0** | **0** | **0** |
| **-1** | **-2** | **-1** |

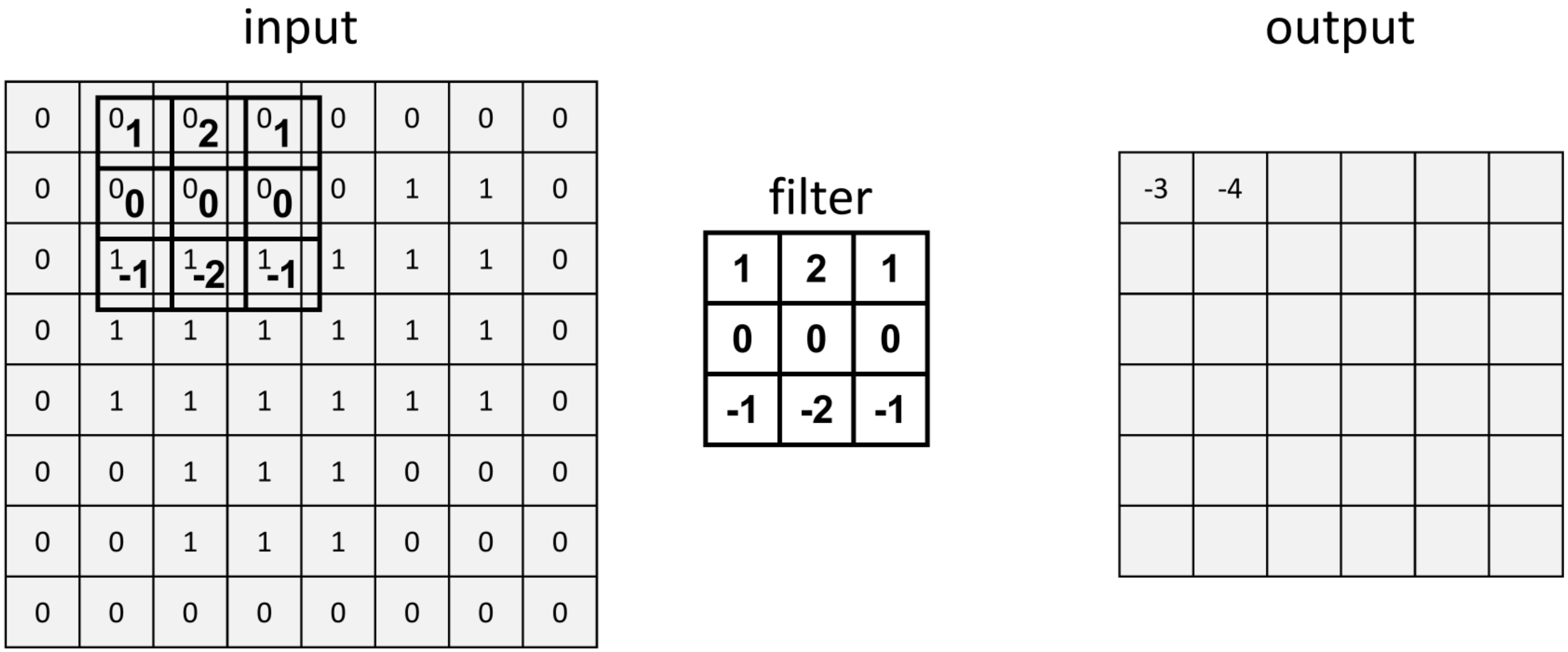
output

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# Convolution: a 2-D example



# Convolution: a 2-D example



# Convolution: a 2-D example

input

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 **1** | 0 **2** | 0 **1** | 0 | 0 | 0 |
| 0 | 0 | 0 **0** | 0 **0** | 0 **0** | 1 | 1 | 0 |
| 0 | 1 | 1 **-1** | 1 **-2** | 1 **-1** | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

filter

| | | |
|---|---|---|
| **1** | **2** | **1** |
| **0** | **0** | **0** |
| **-1** | **-2** | **-1** |

output

| | | | | | |
|---|---|---|---|---|---|
| -3 | -4 | -4 | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# Convolution: a 2-D example

input

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 **1** | 0 **2** | 0 **1** | 0 | 0 |
| 0 | 0 | 0 | 0 **0** | 0 **0** | 1 **0** | 1 | 0 |
| 0 | 1 | 1 | 1 **-1** | 1 **-2** | 1 **-1** | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

filter

| | | |
|---|---|---|
| **1** | **2** | **1** |
| **0** | **0** | **0** |
| **-1** | **-2** | **-1** |

output

| | | | | | |
|---|---|---|---|---|---|
| -3 | -4 | -4 | -4 | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# Convolution: a 2-D example

input

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

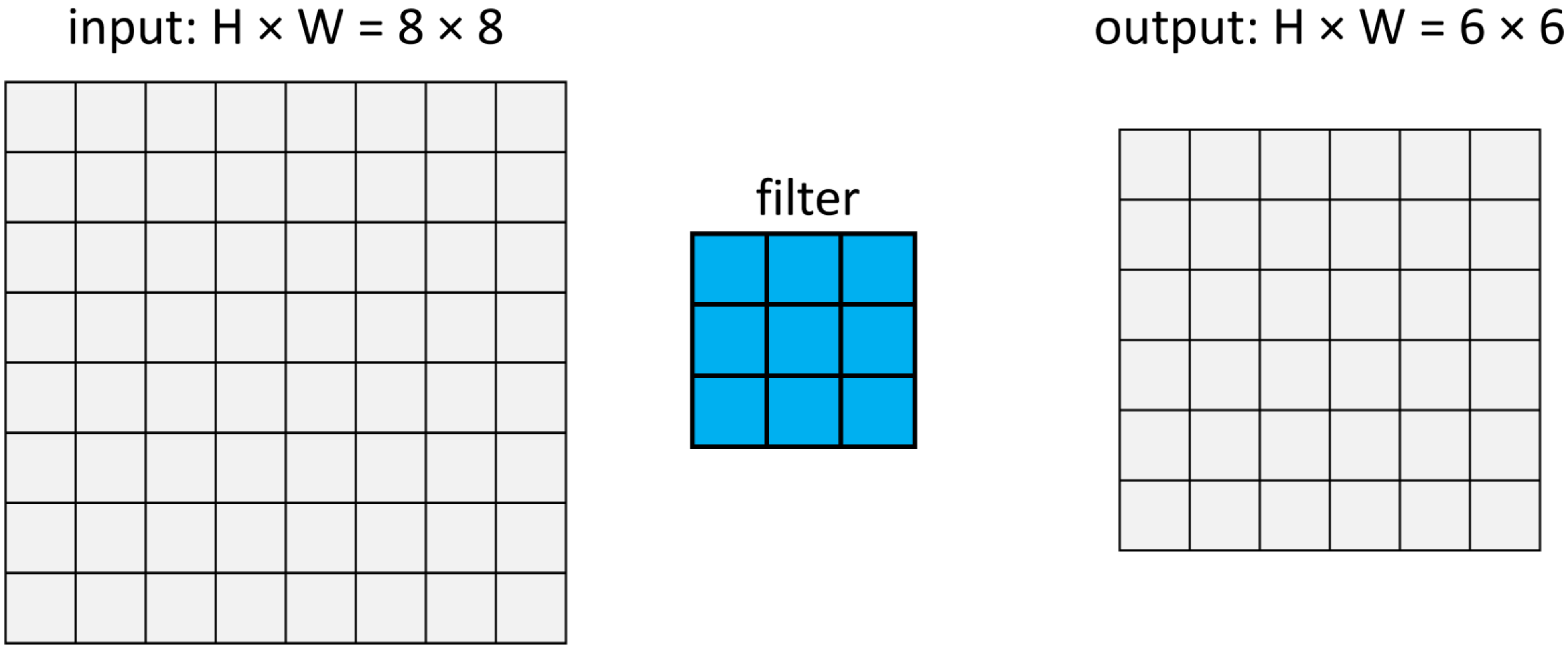
filter

| | | |
|---|---|---|
| **1** | **2** | **1** |
| **0** | **0** | **0** |
| **-1** | **-2** | **-1** |

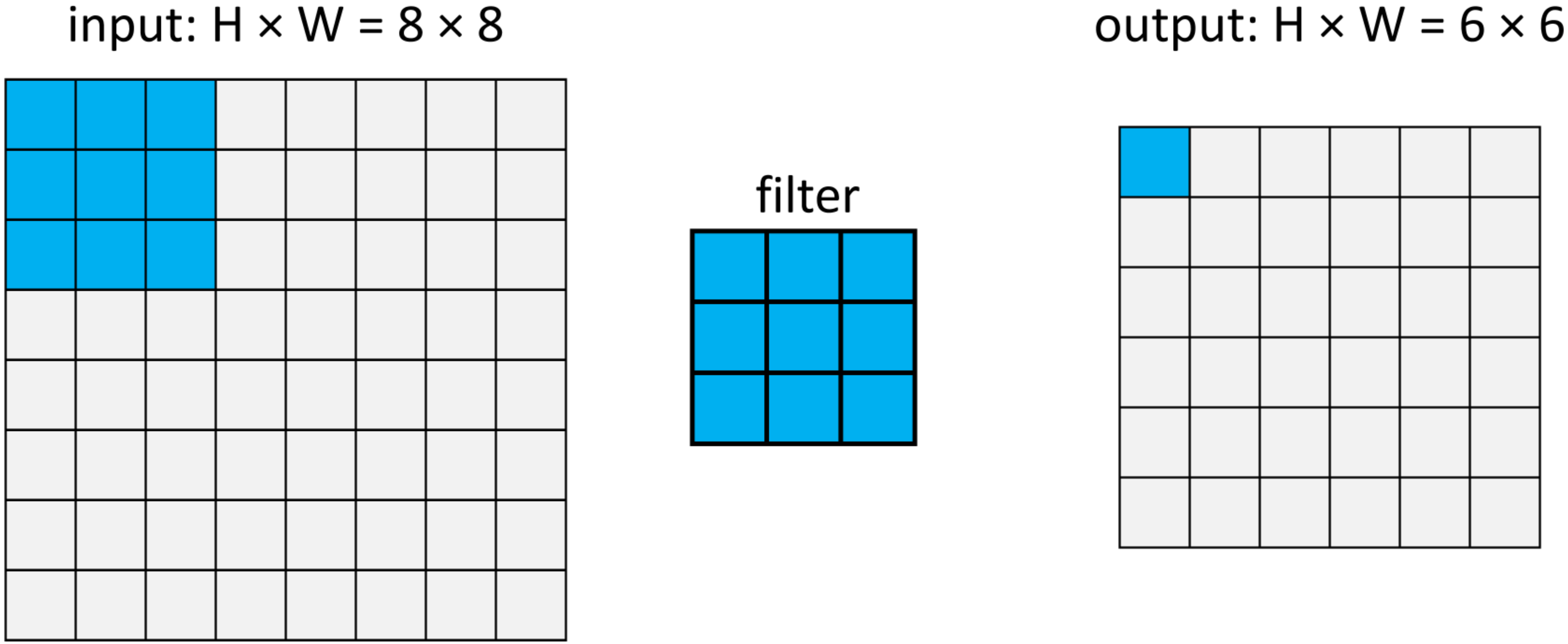
output

| | | | | | |
|---|---|---|---|---|---|
| -3 | -4 | -4 | -4 | -4 | -3 |
| -3 | -4 | -4 | -3 | -1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 3 | 3 |
| 2 | 1 | 0 | 1 | 3 | 3 |
| 1 | 3 | 4 | 3 | 1 | 0 |

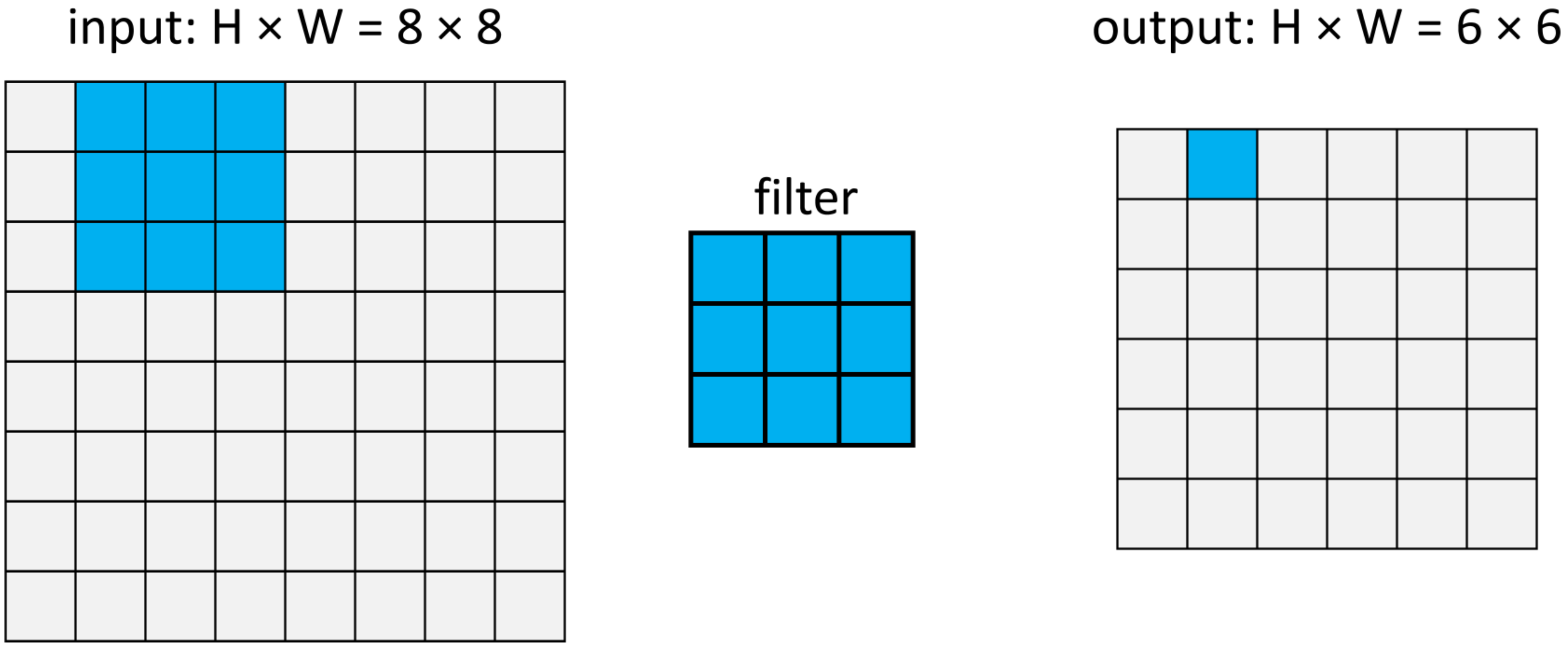
# Convolution: padding



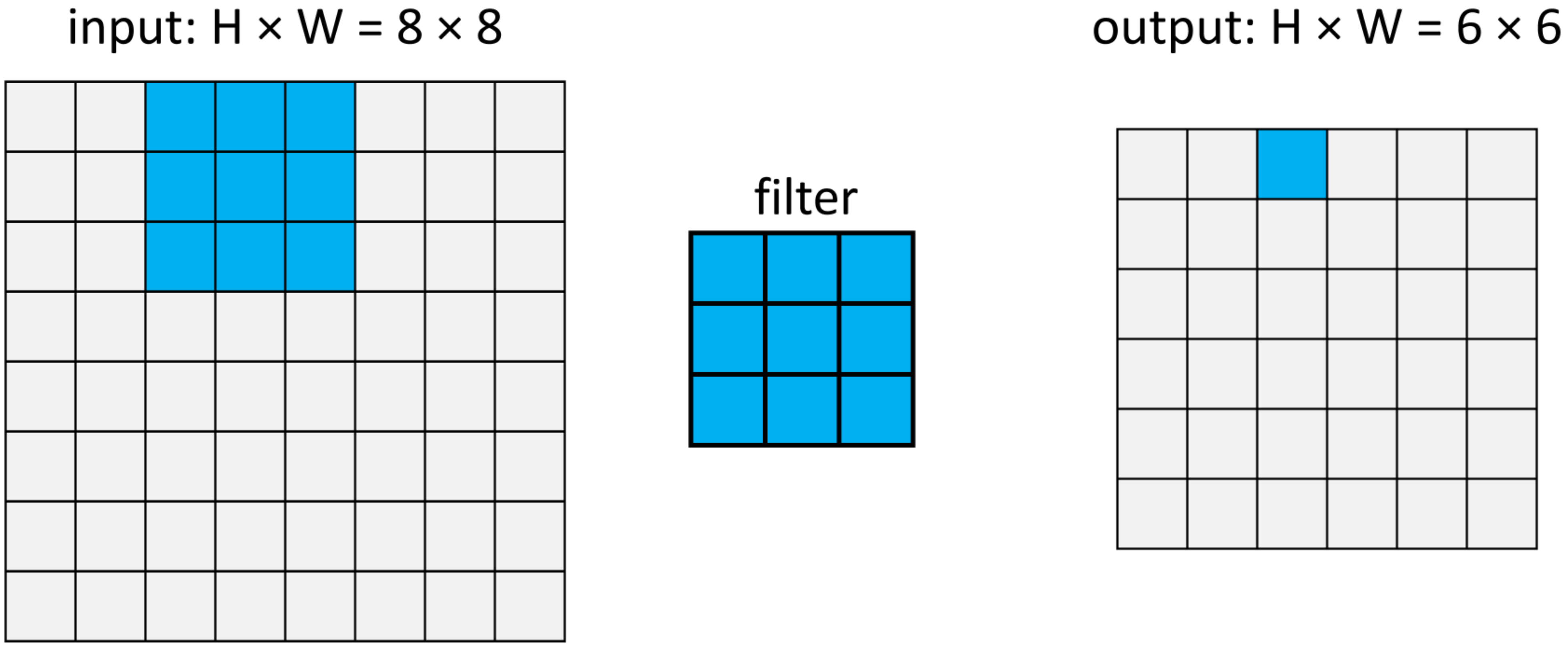
# Convolution: padding



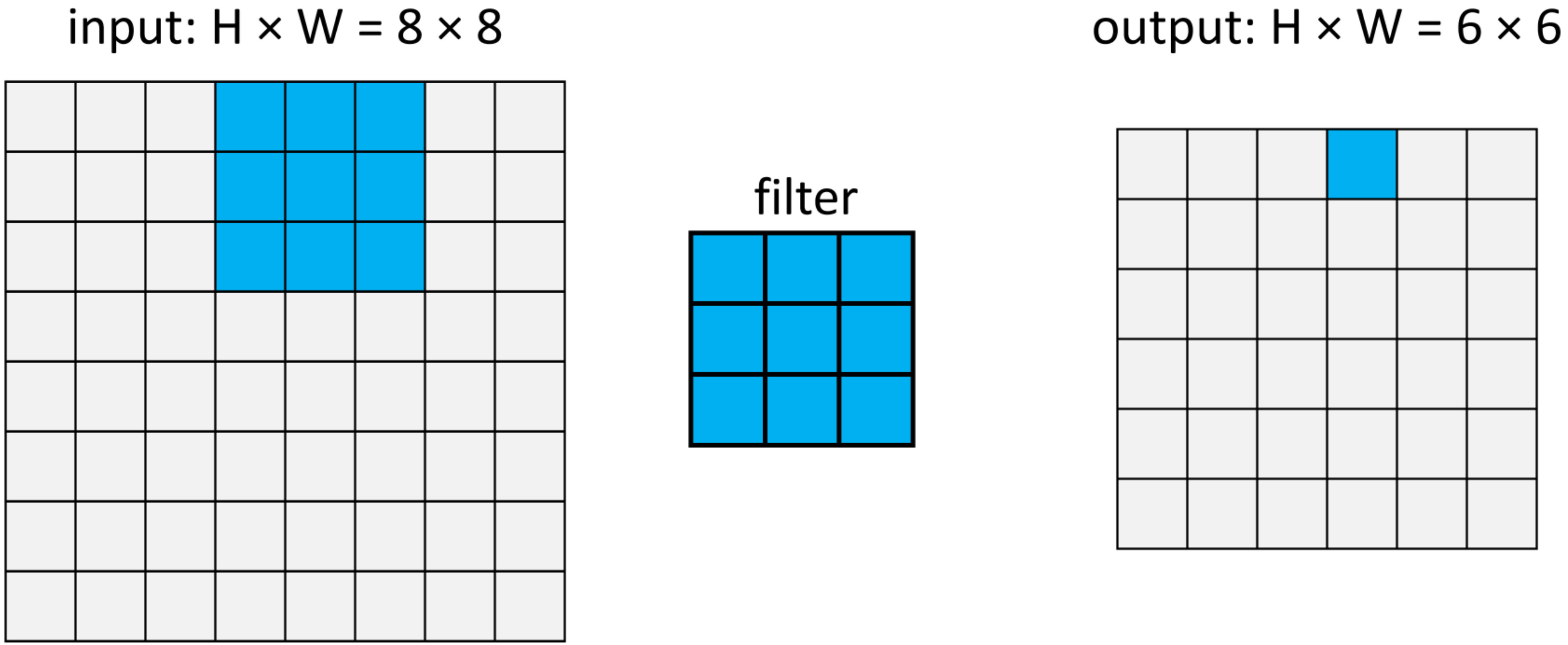
# Convolution: padding



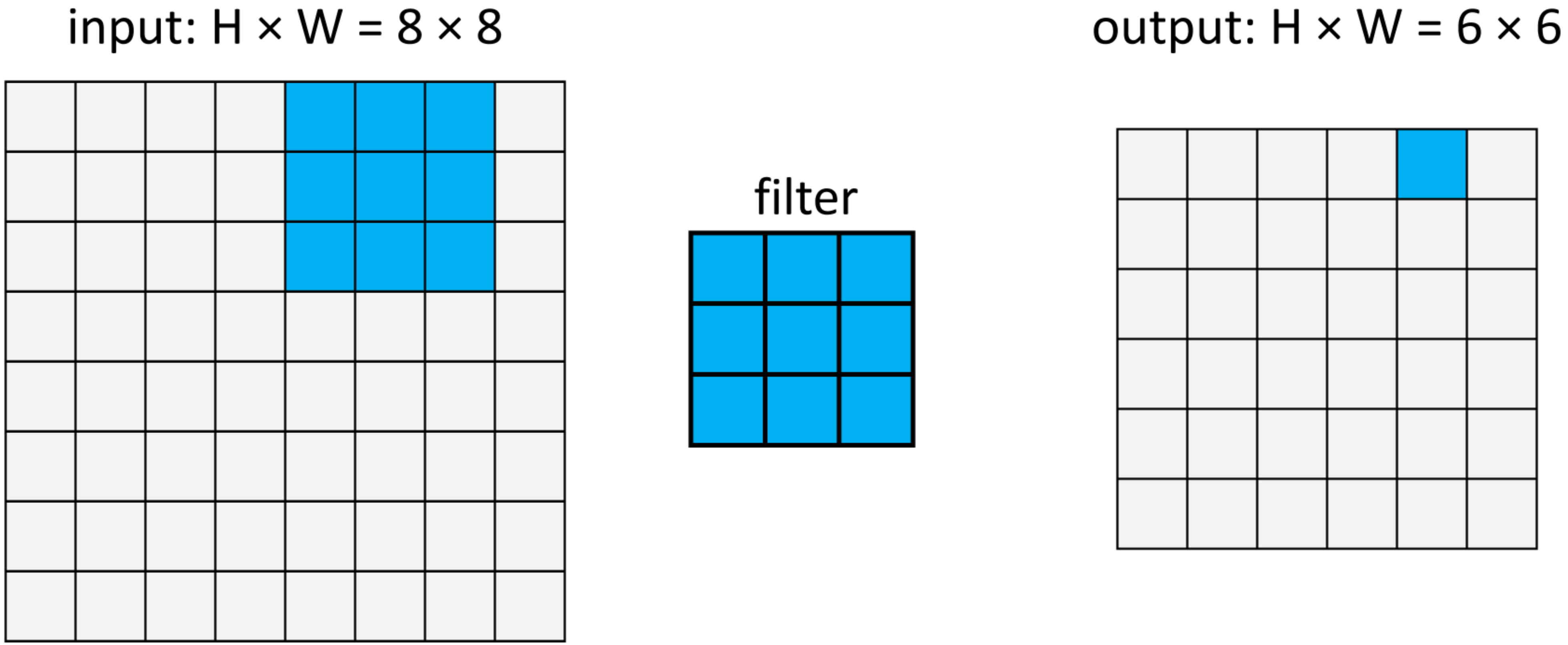
# Convolution: padding



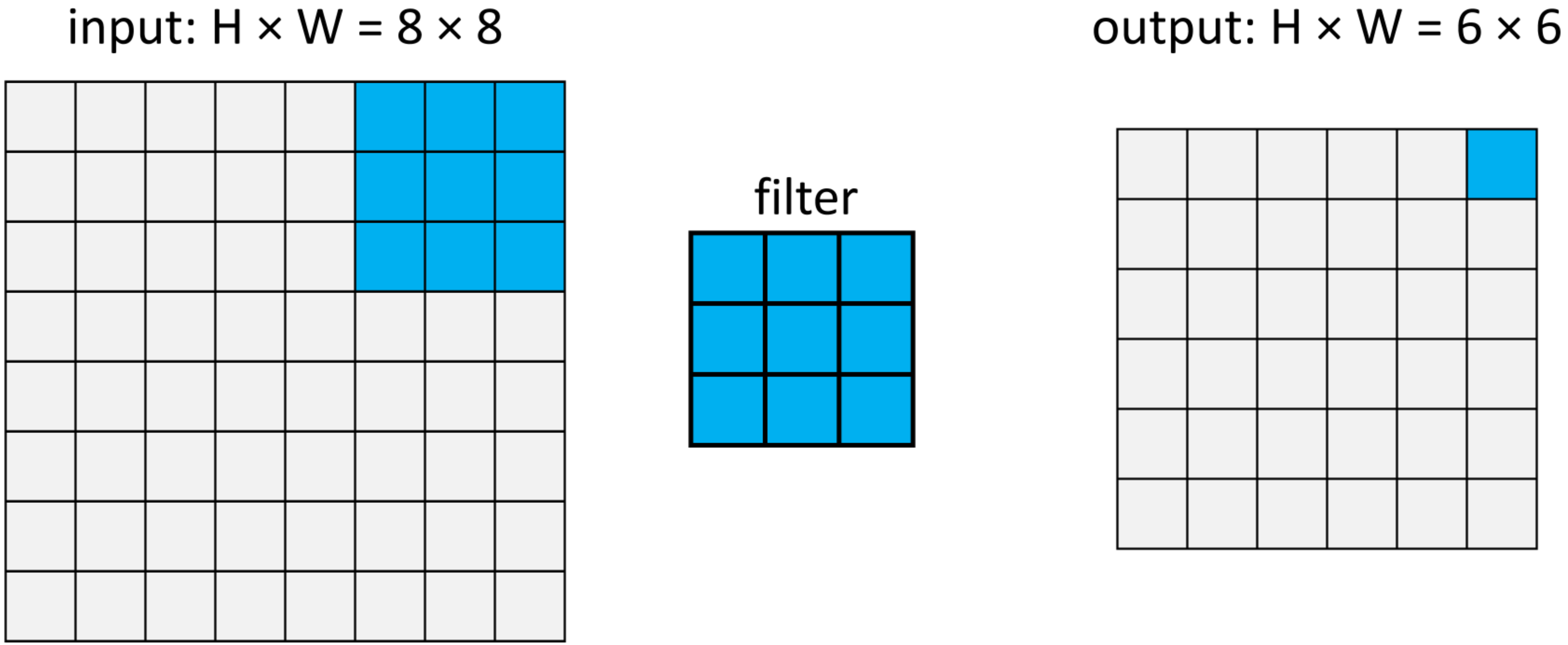
# Convolution: padding



# Convolution: padding

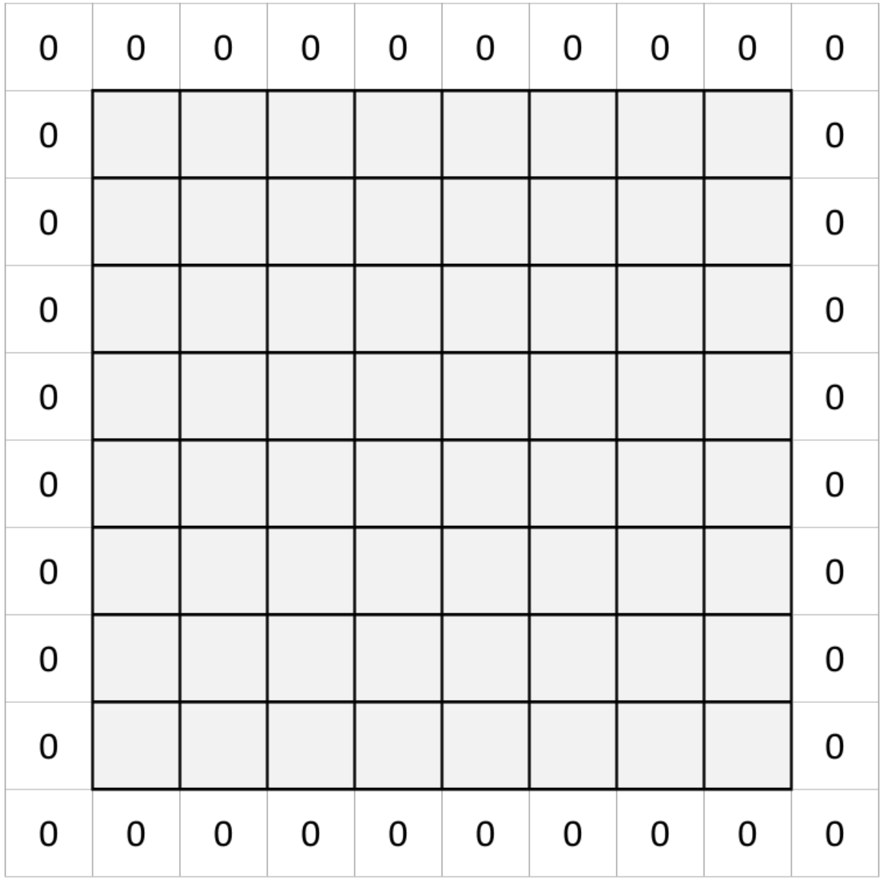


# Convolution: padding

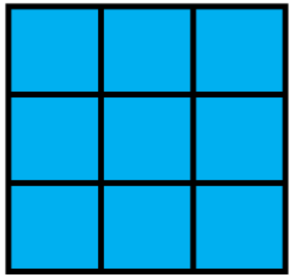


# Convolution: padding

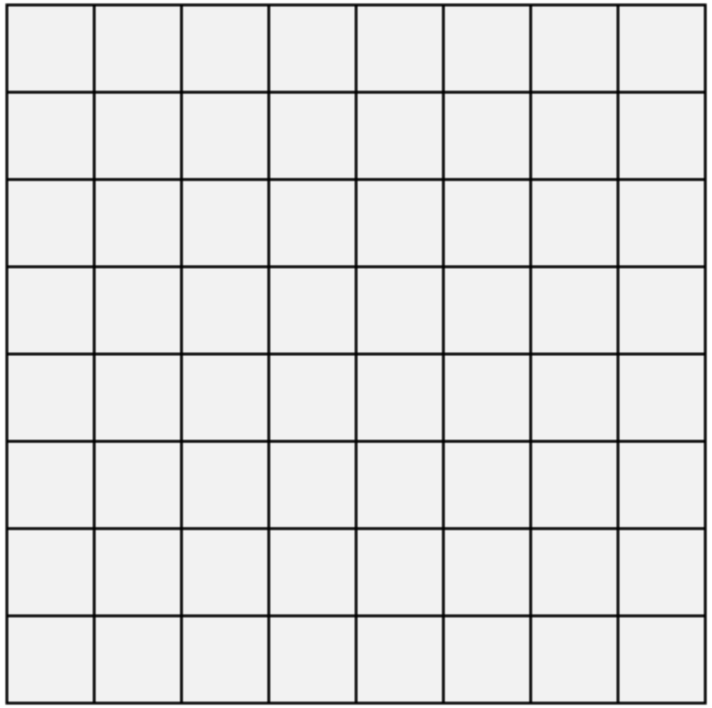
input: $8 \times 8$, + pad



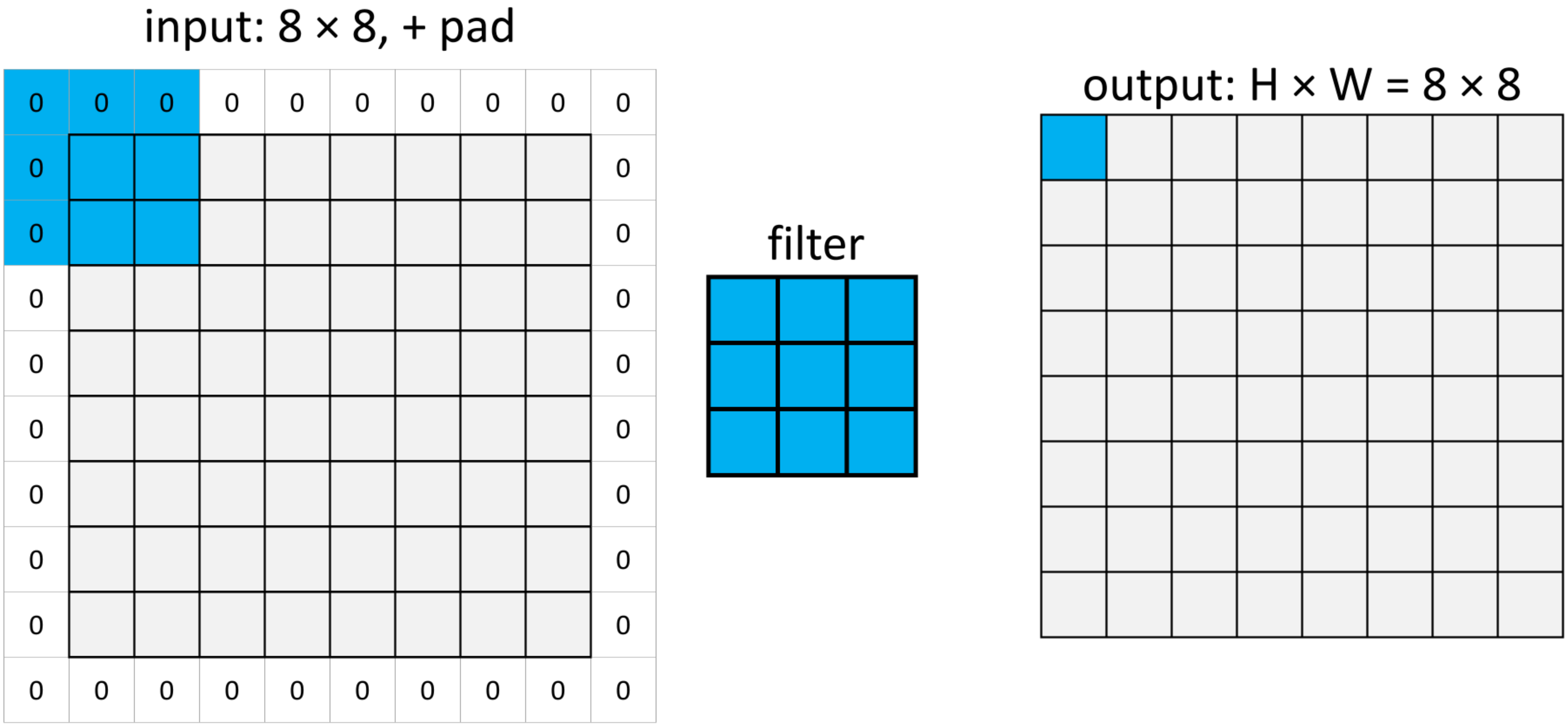
filter



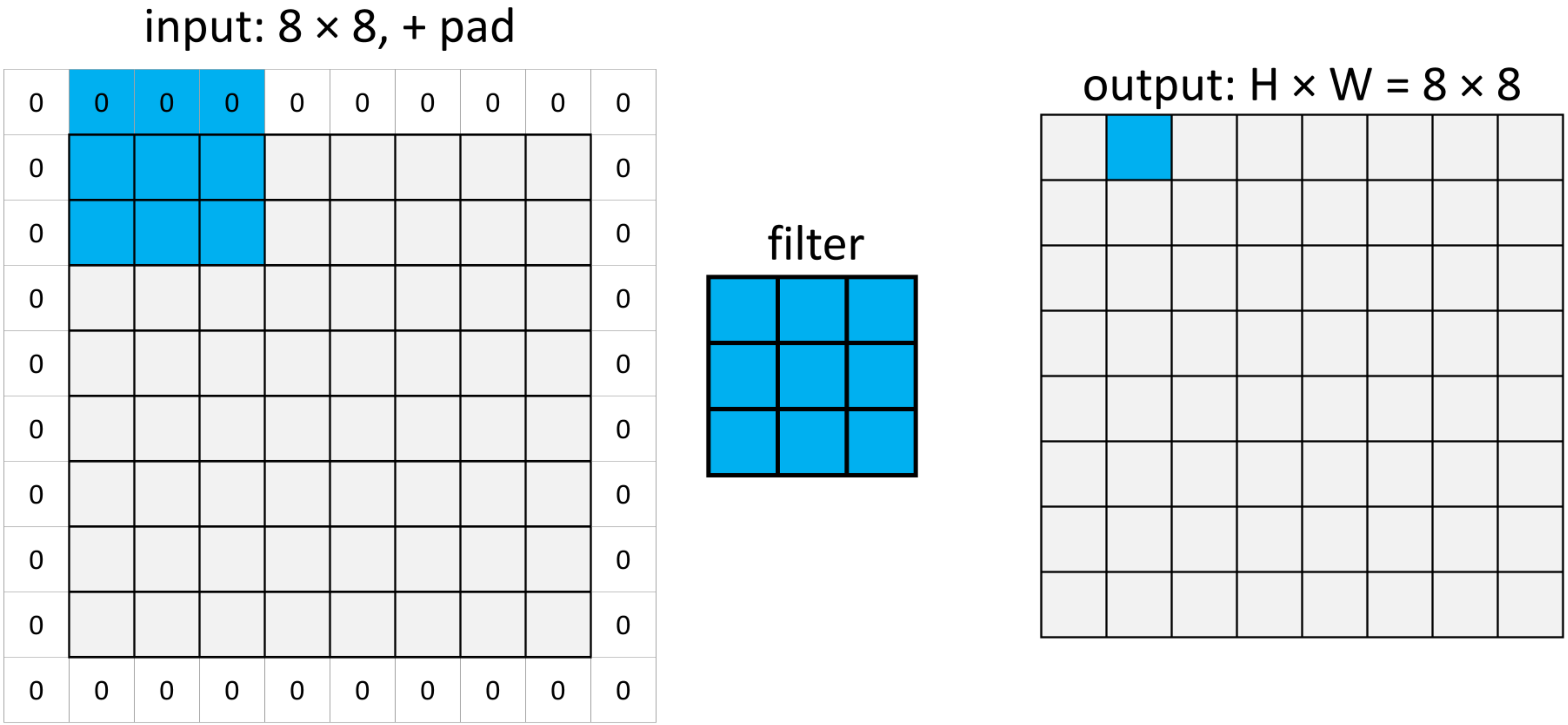
output: $H \times W = 8 \times 8$



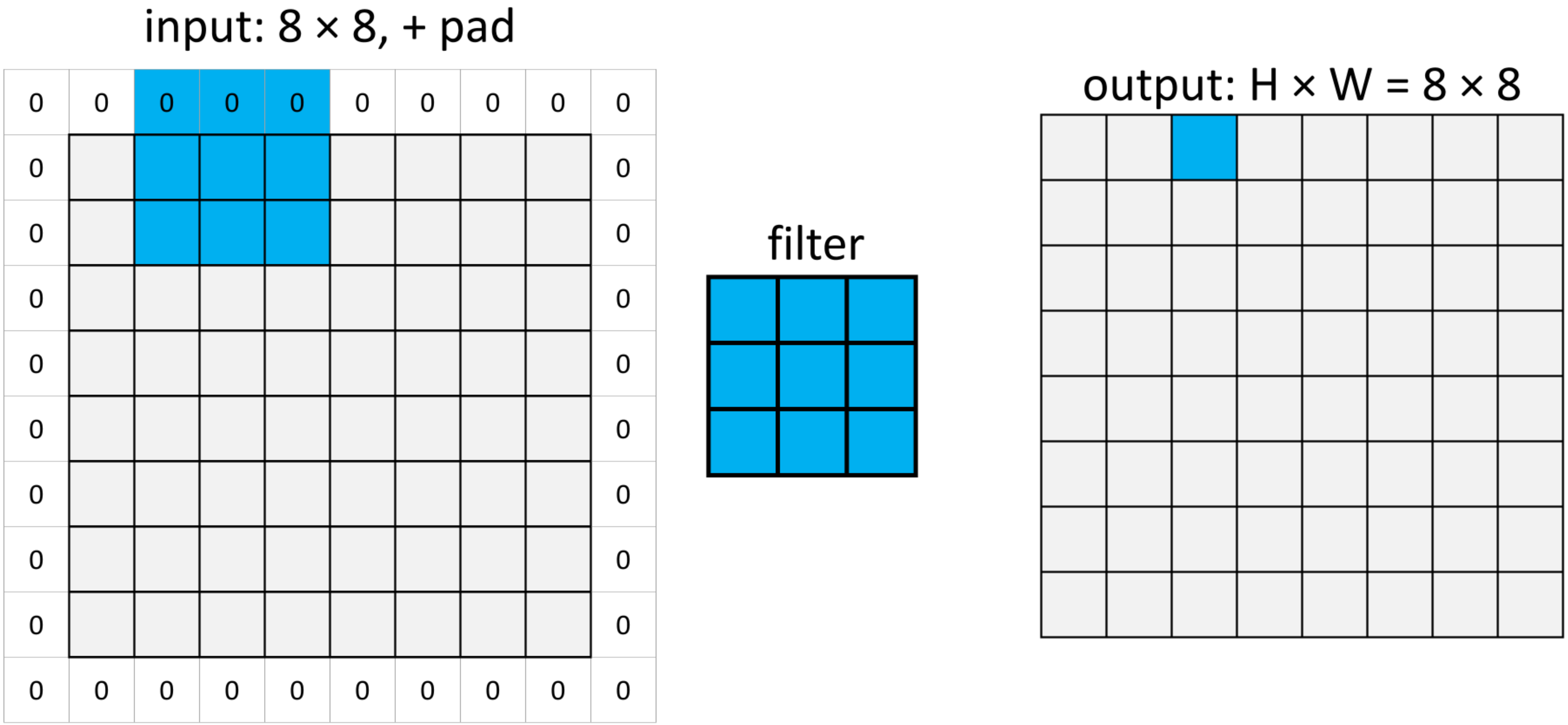
# Convolution: padding



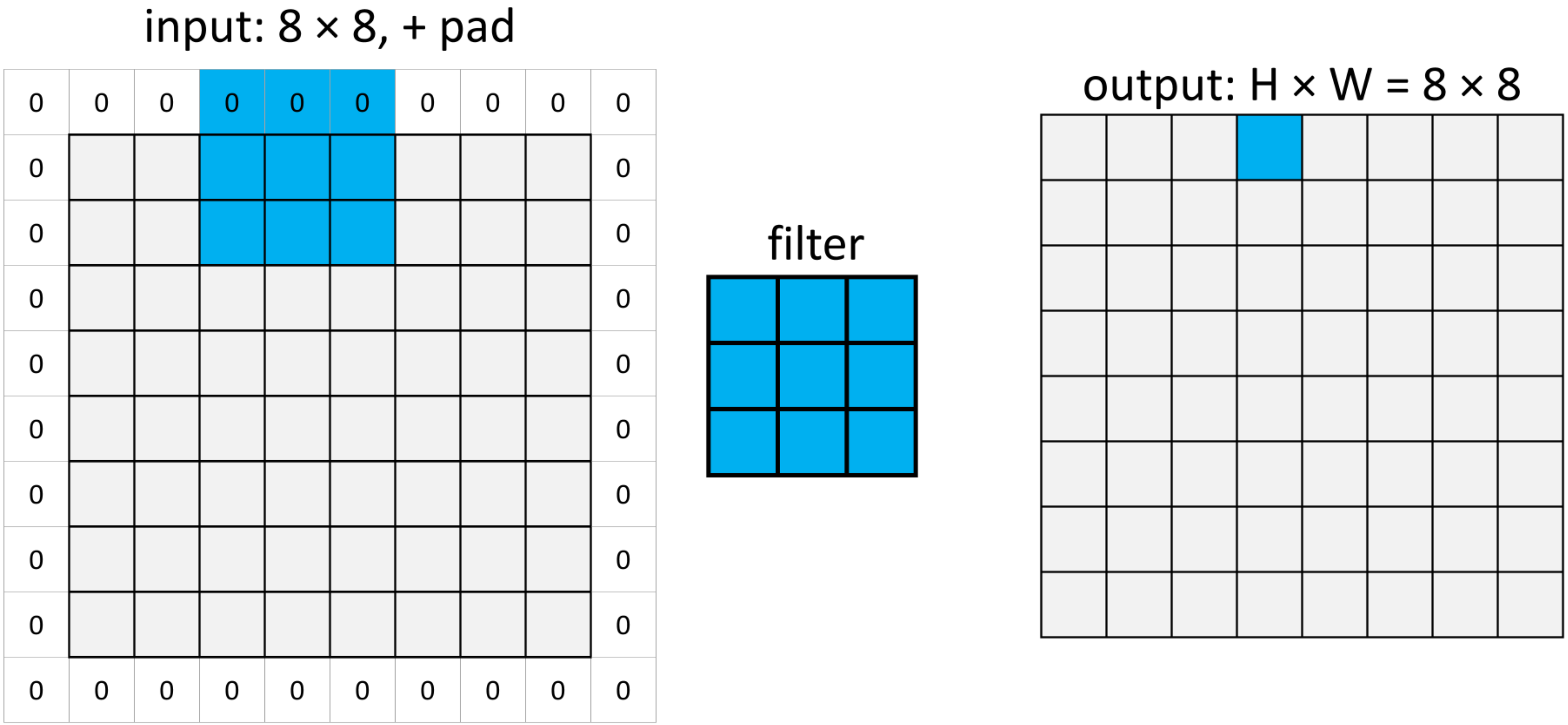
# Convolution: padding



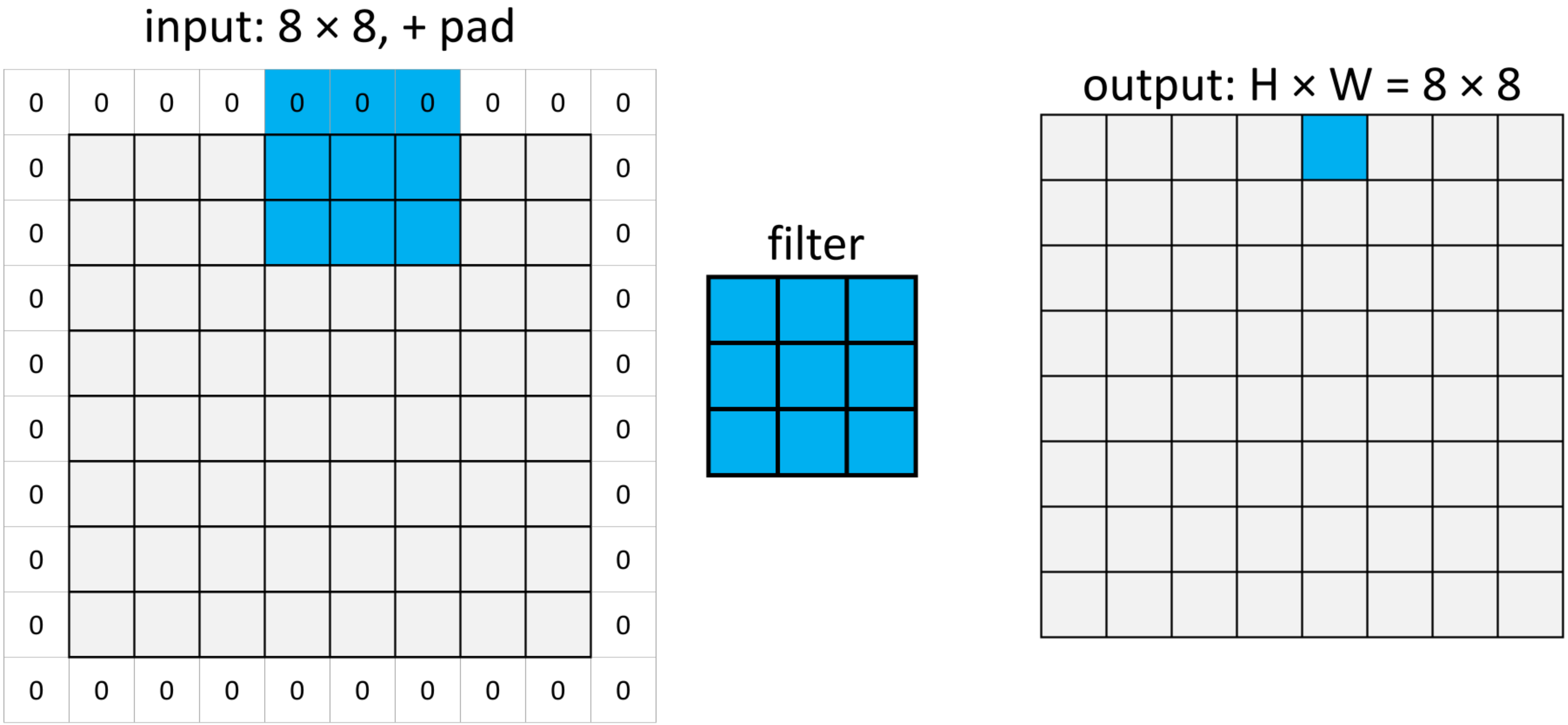
# Convolution: padding



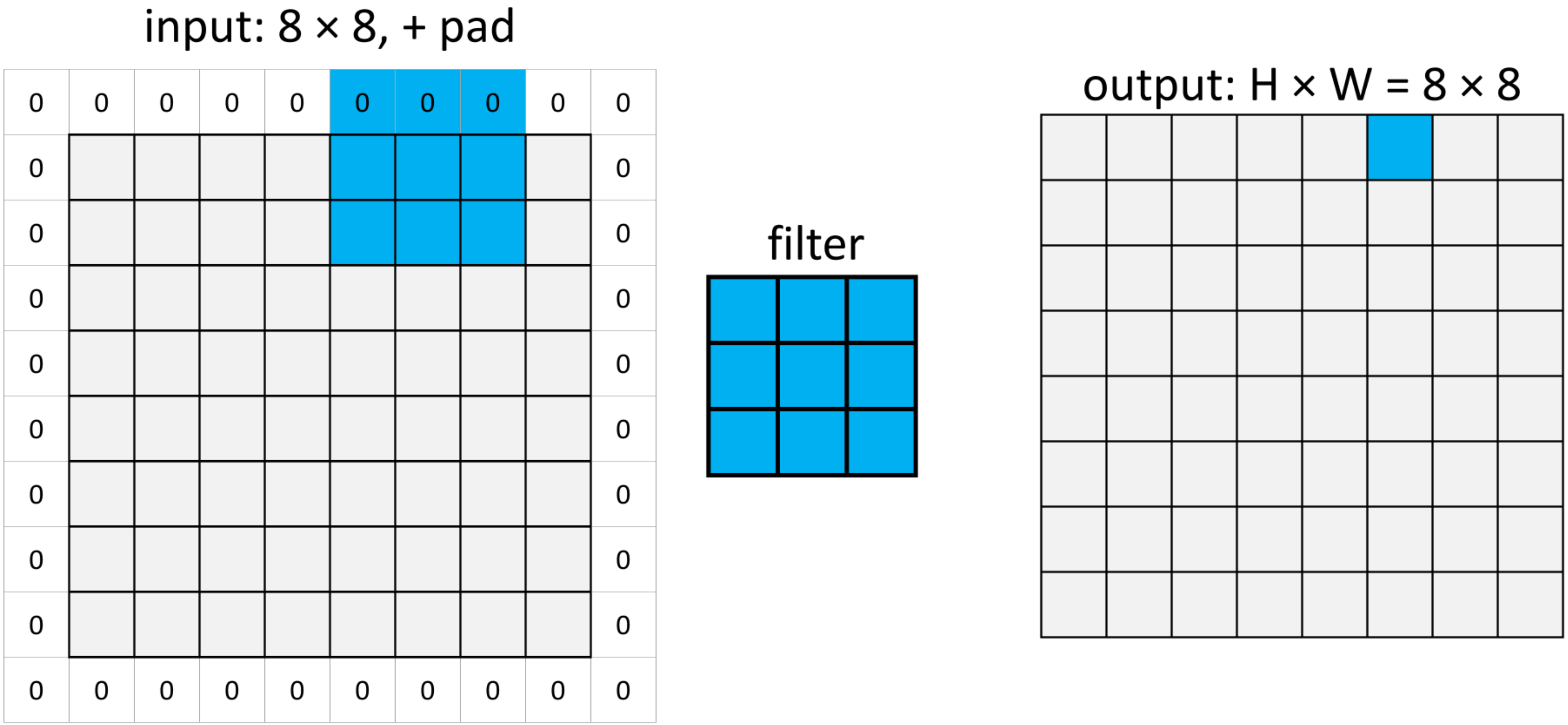
# Convolution: padding



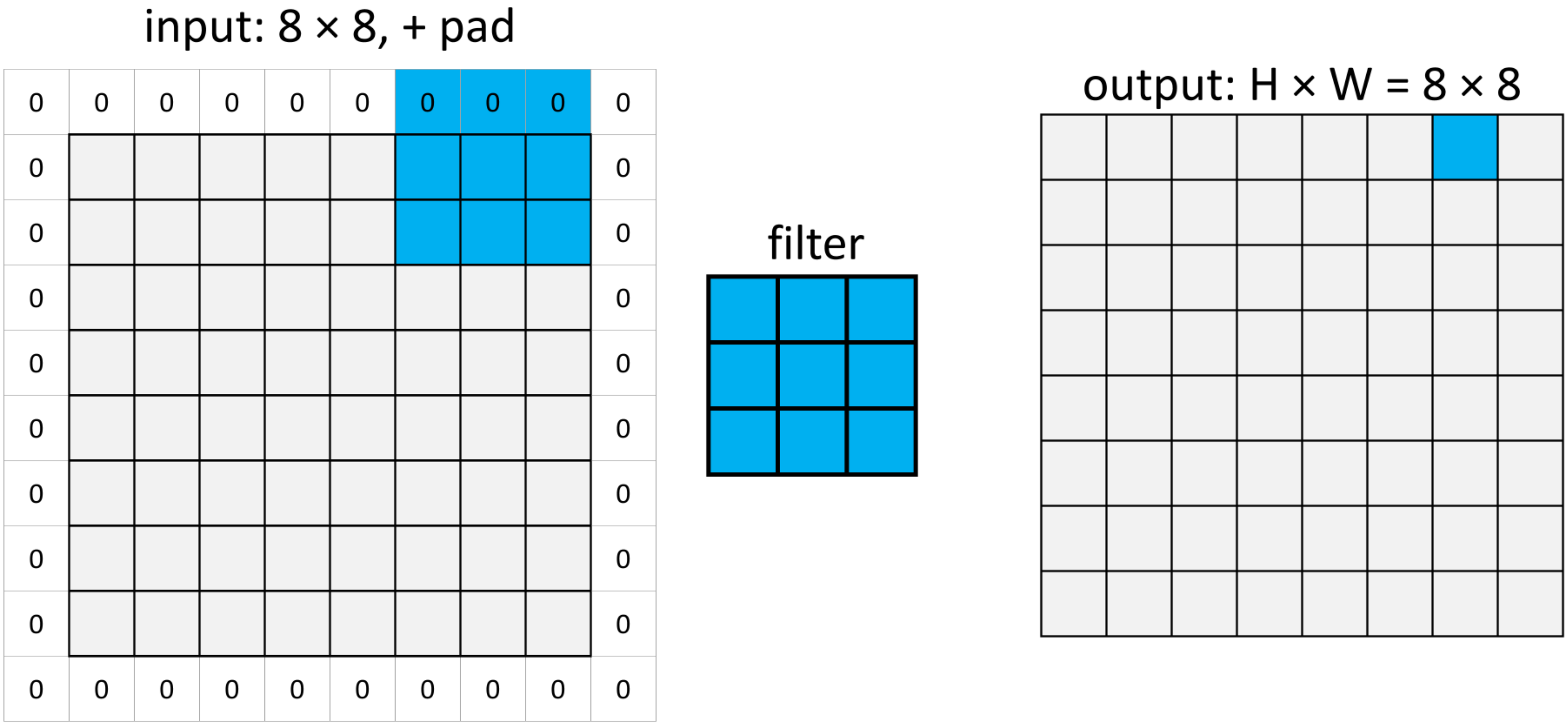
# Convolution: padding



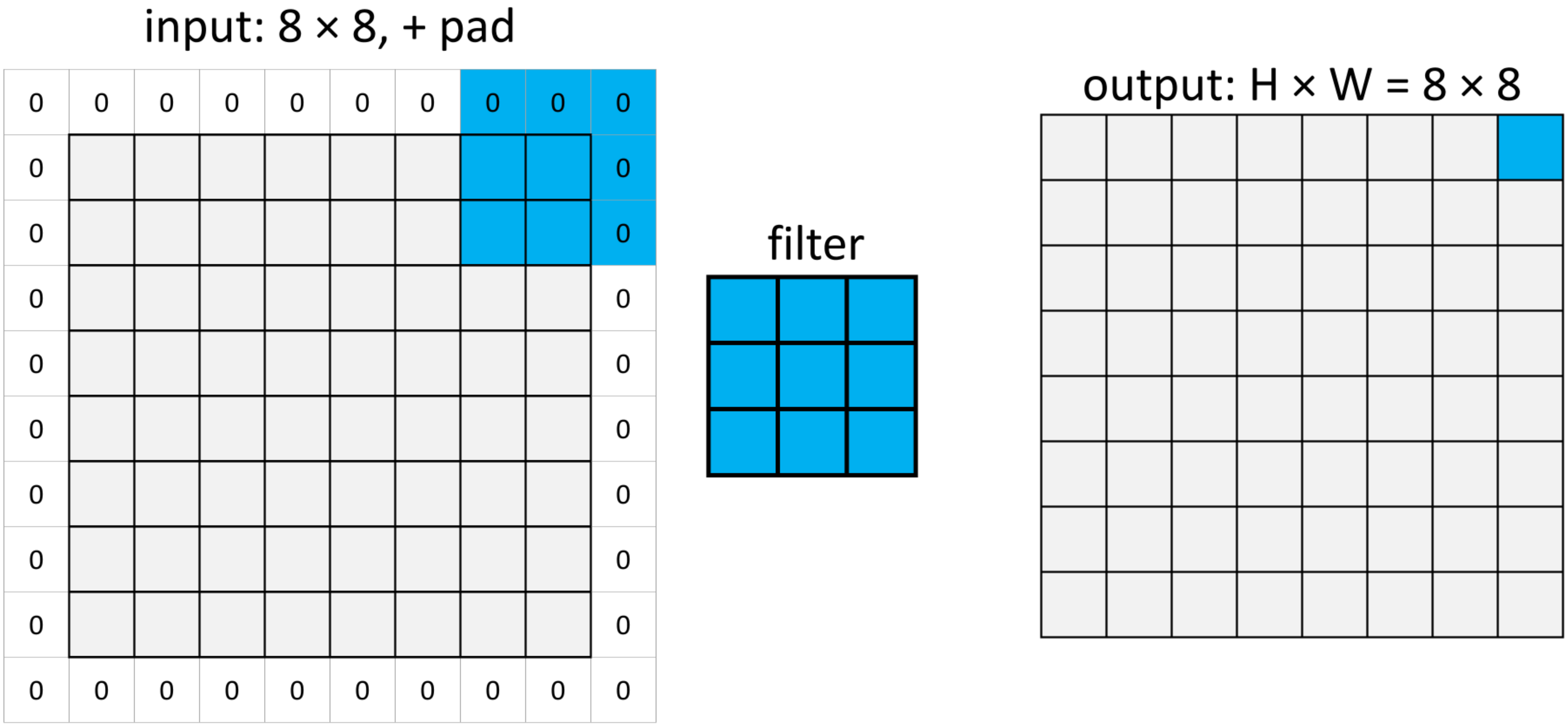
# Convolution: padding



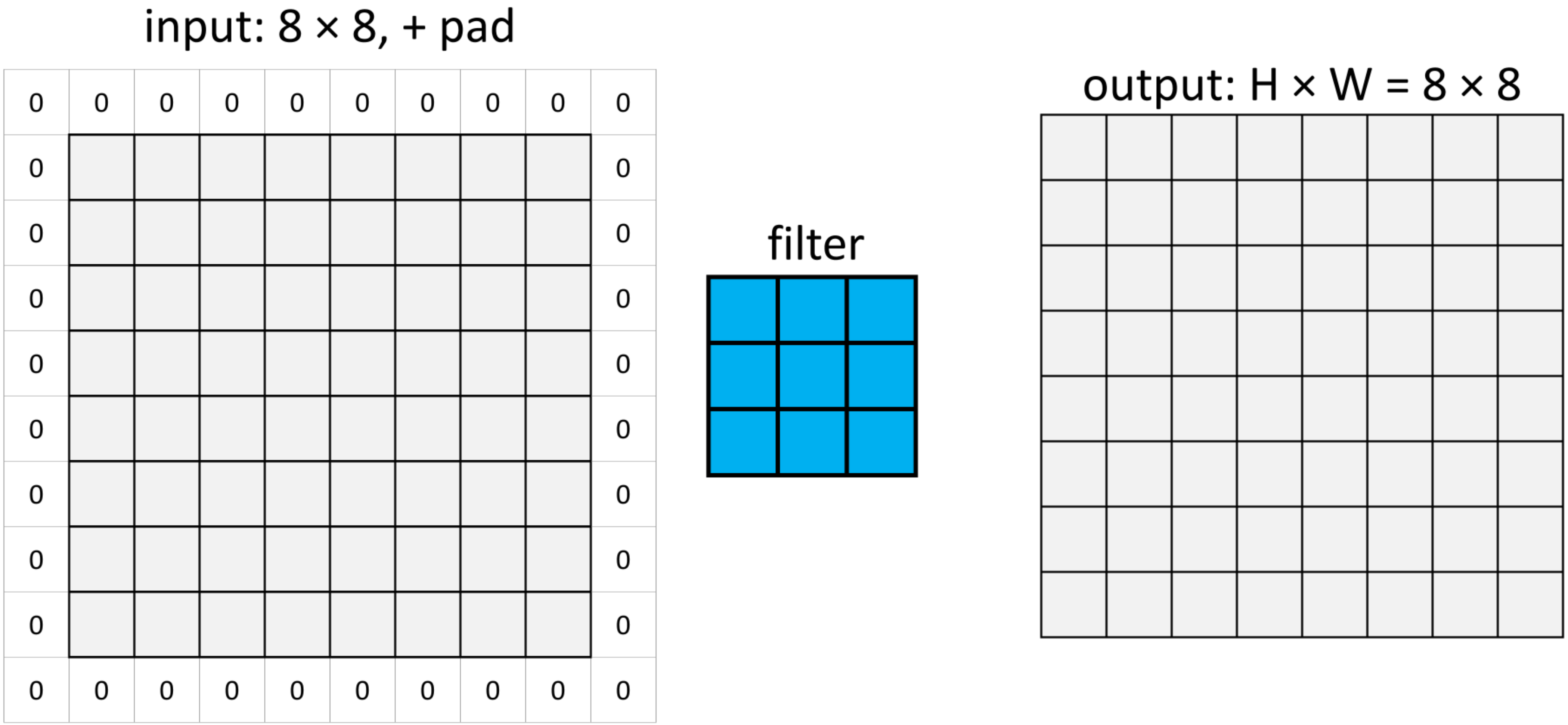
# Convolution: padding



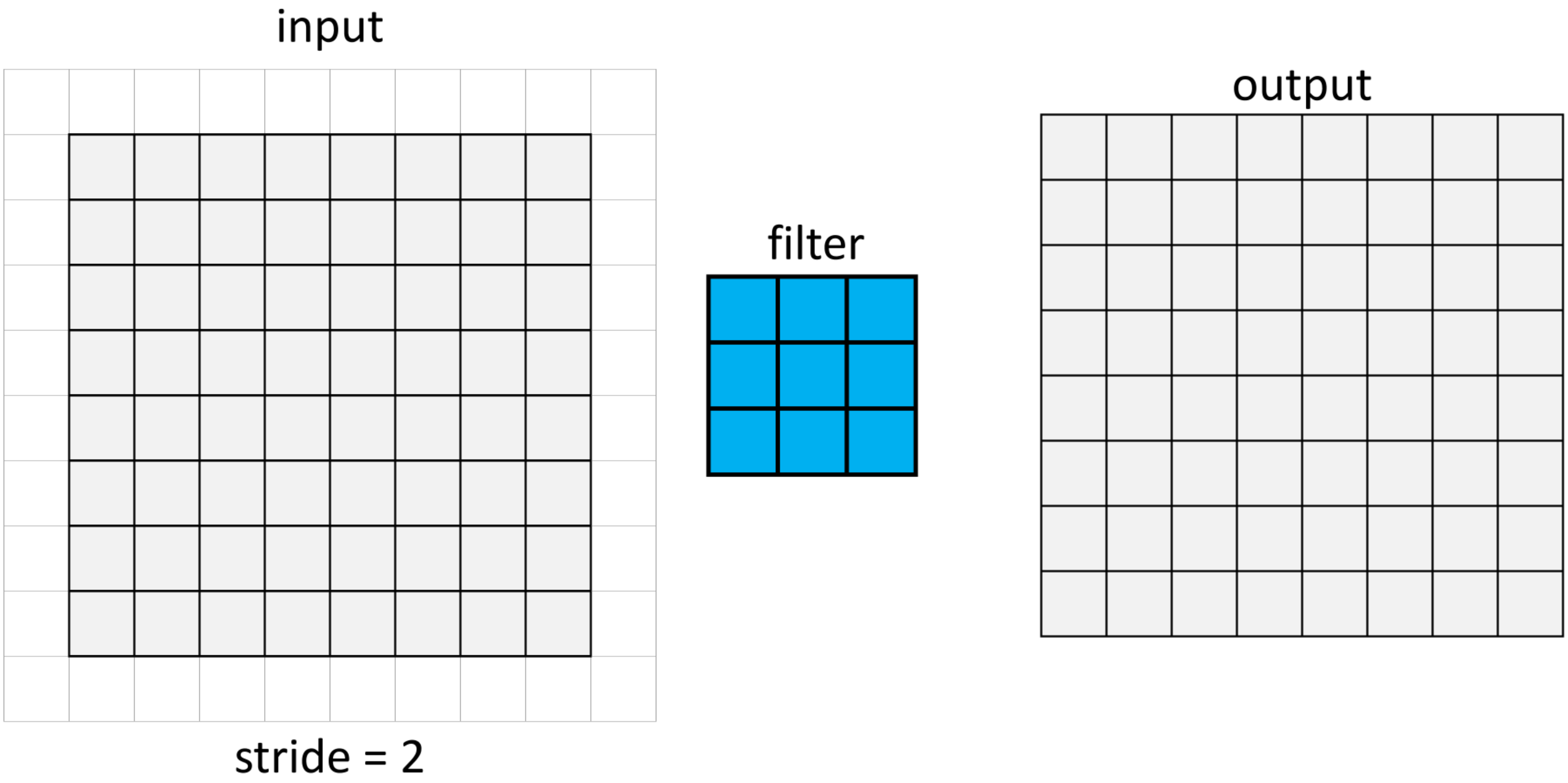
# Convolution: padding



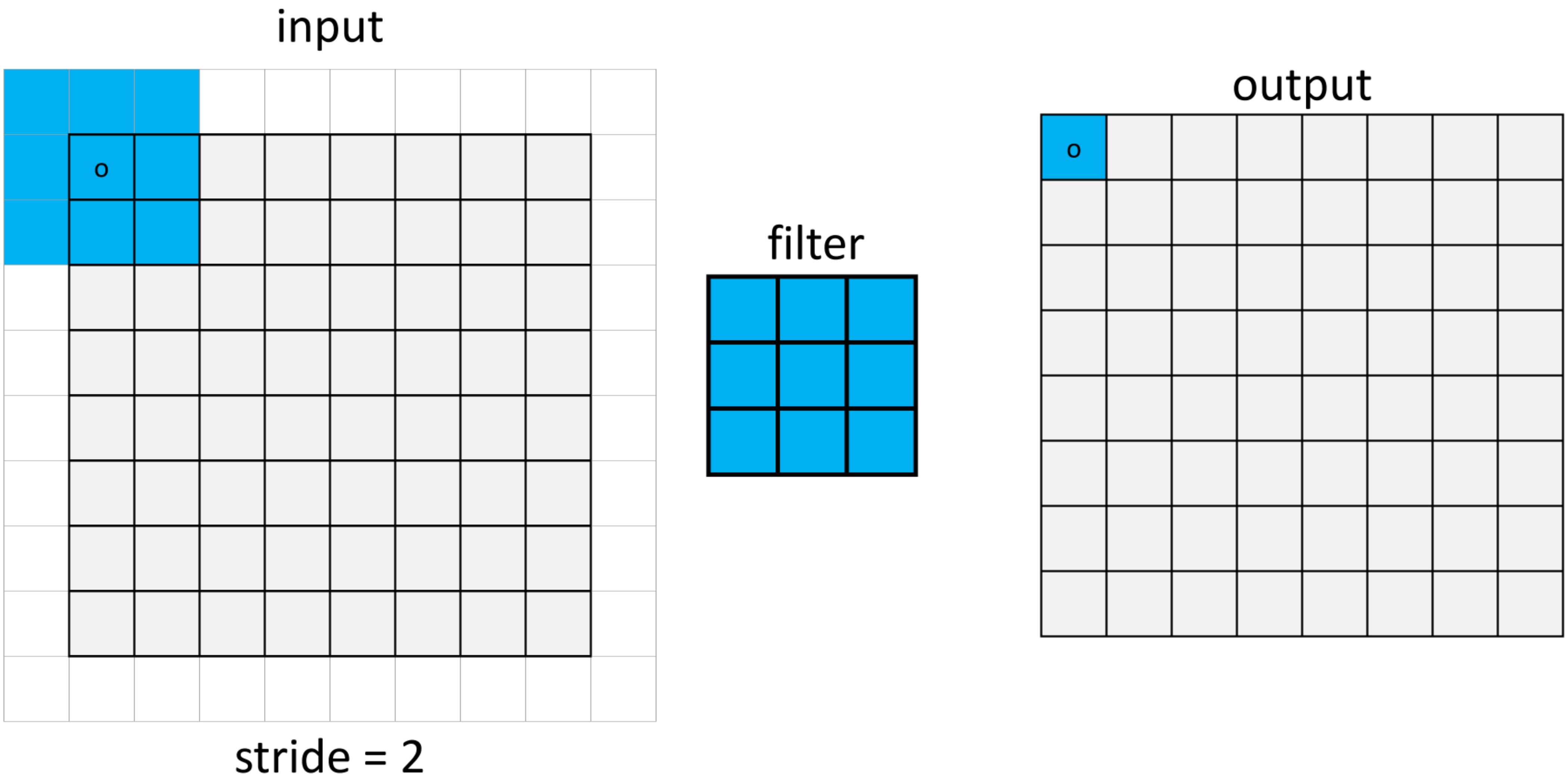
# Convolution: padding



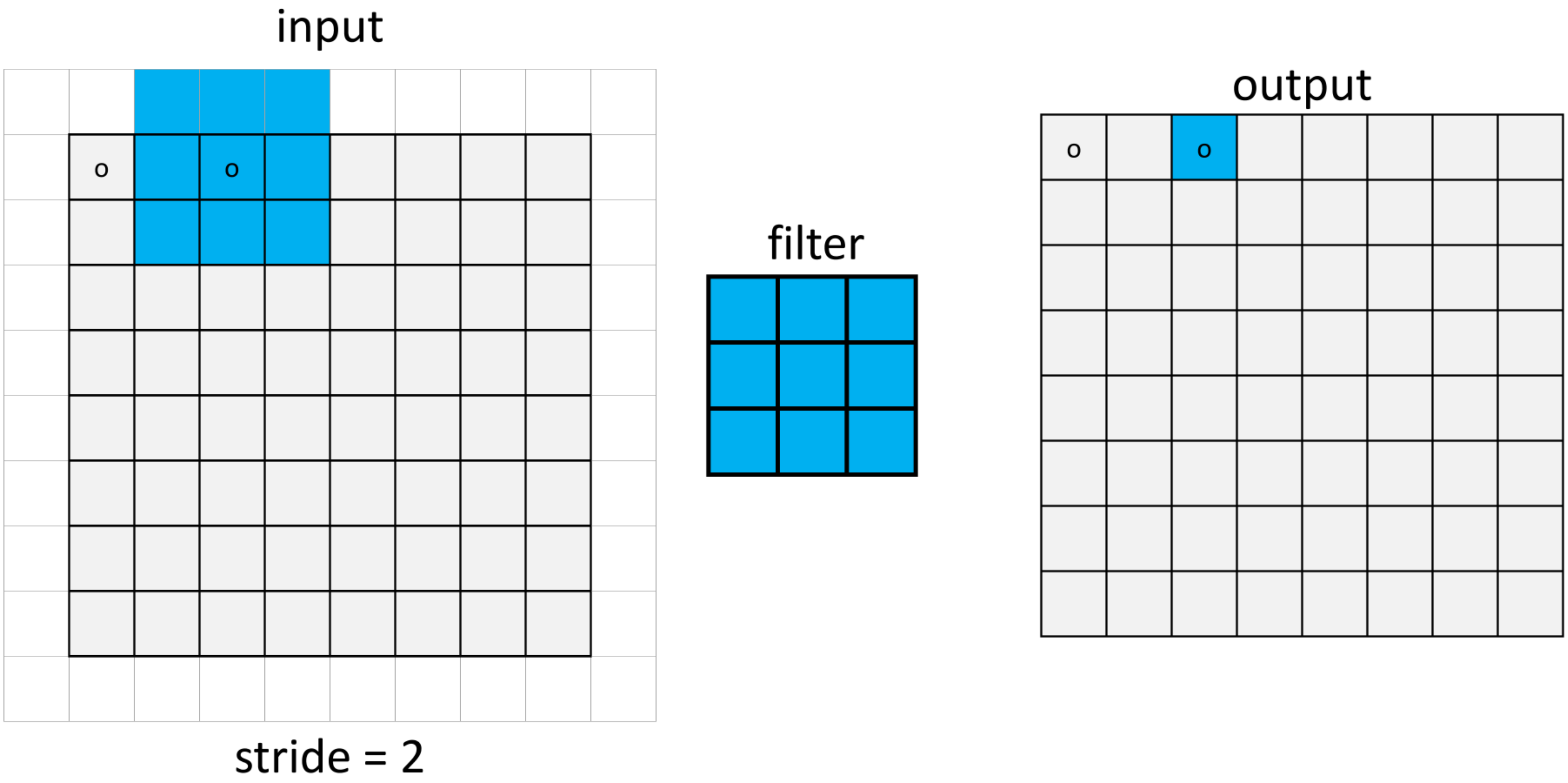
# Convolution: stride



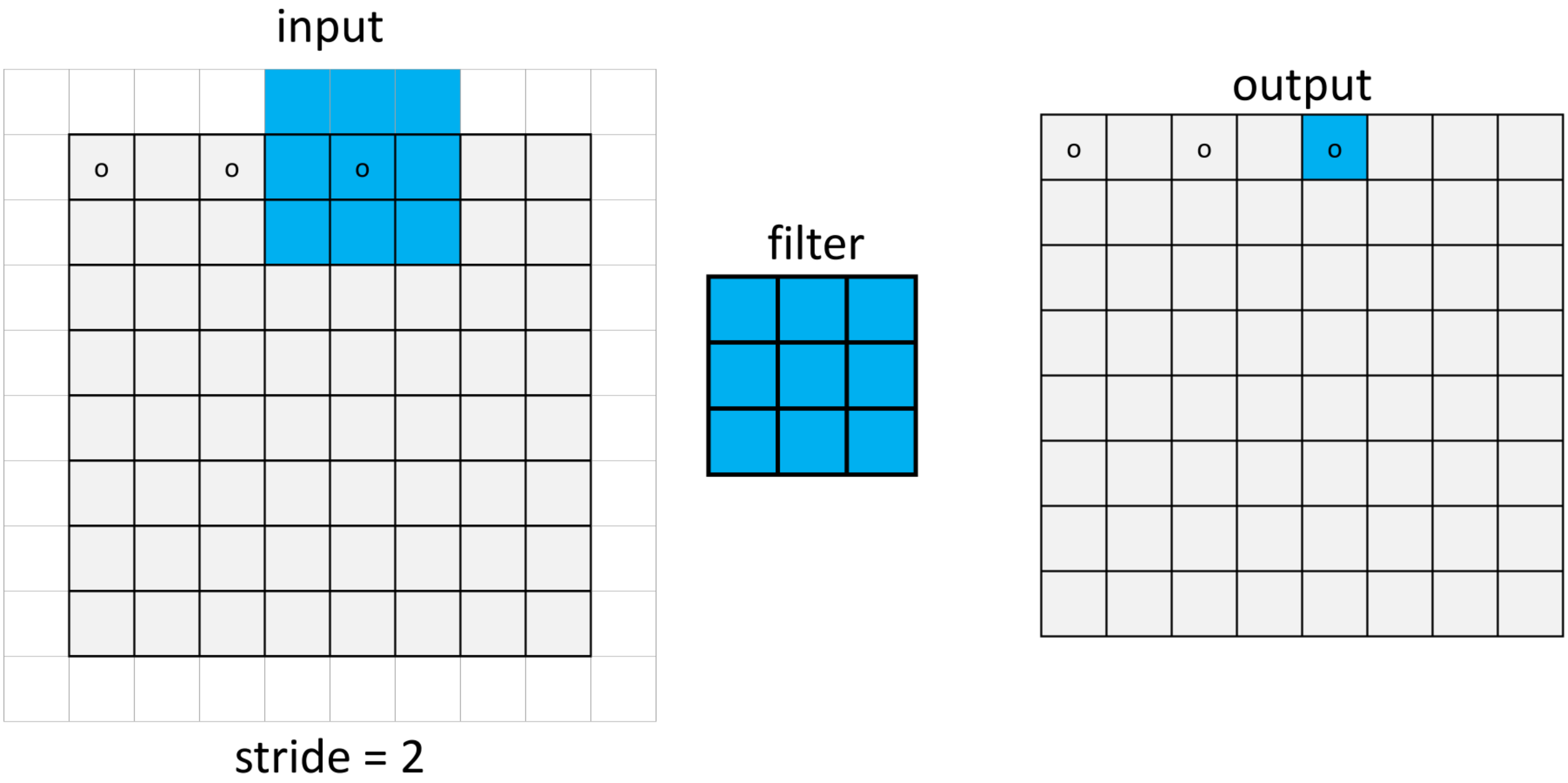
# Convolution: stride



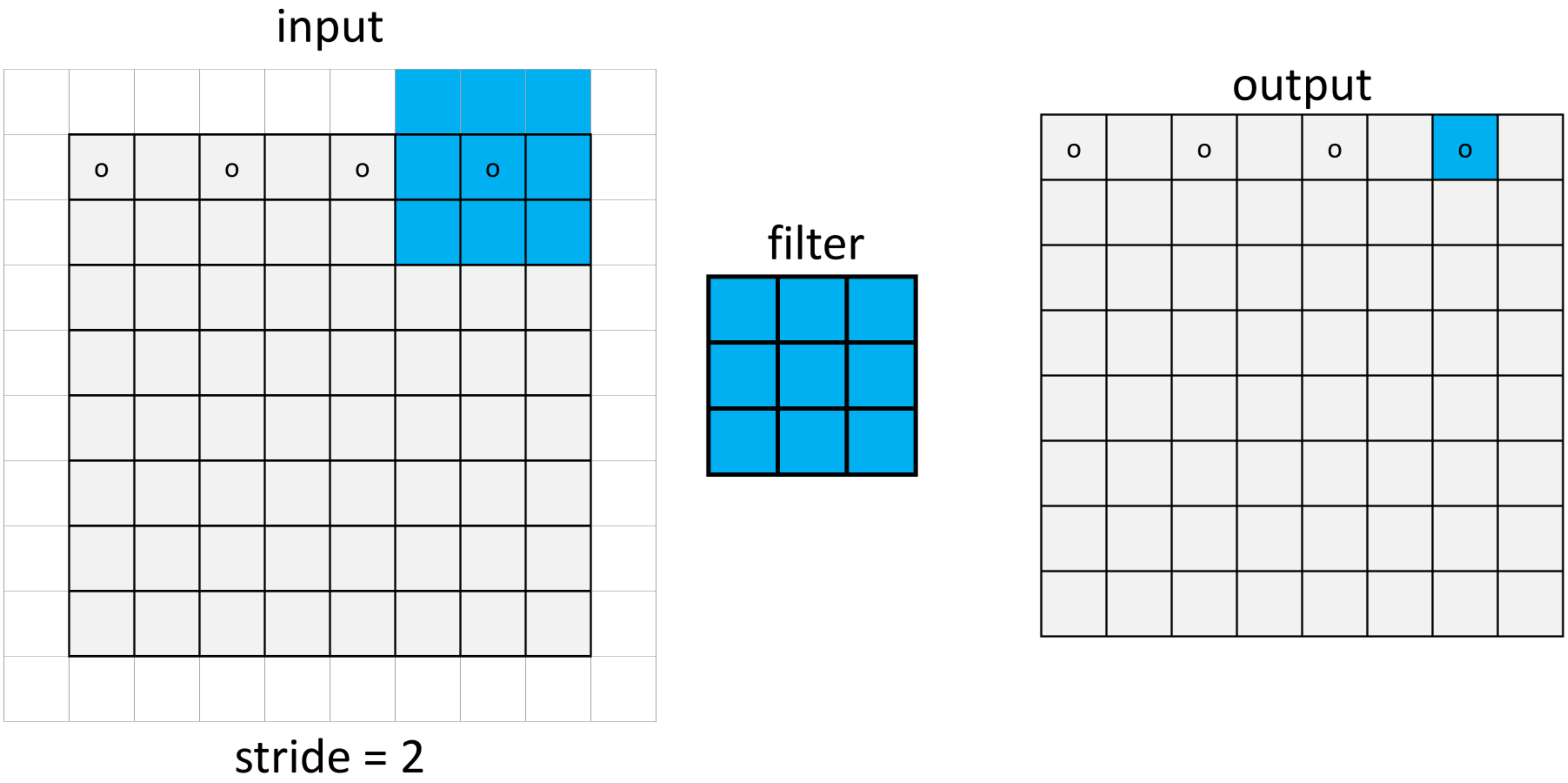
# Convolution: stride



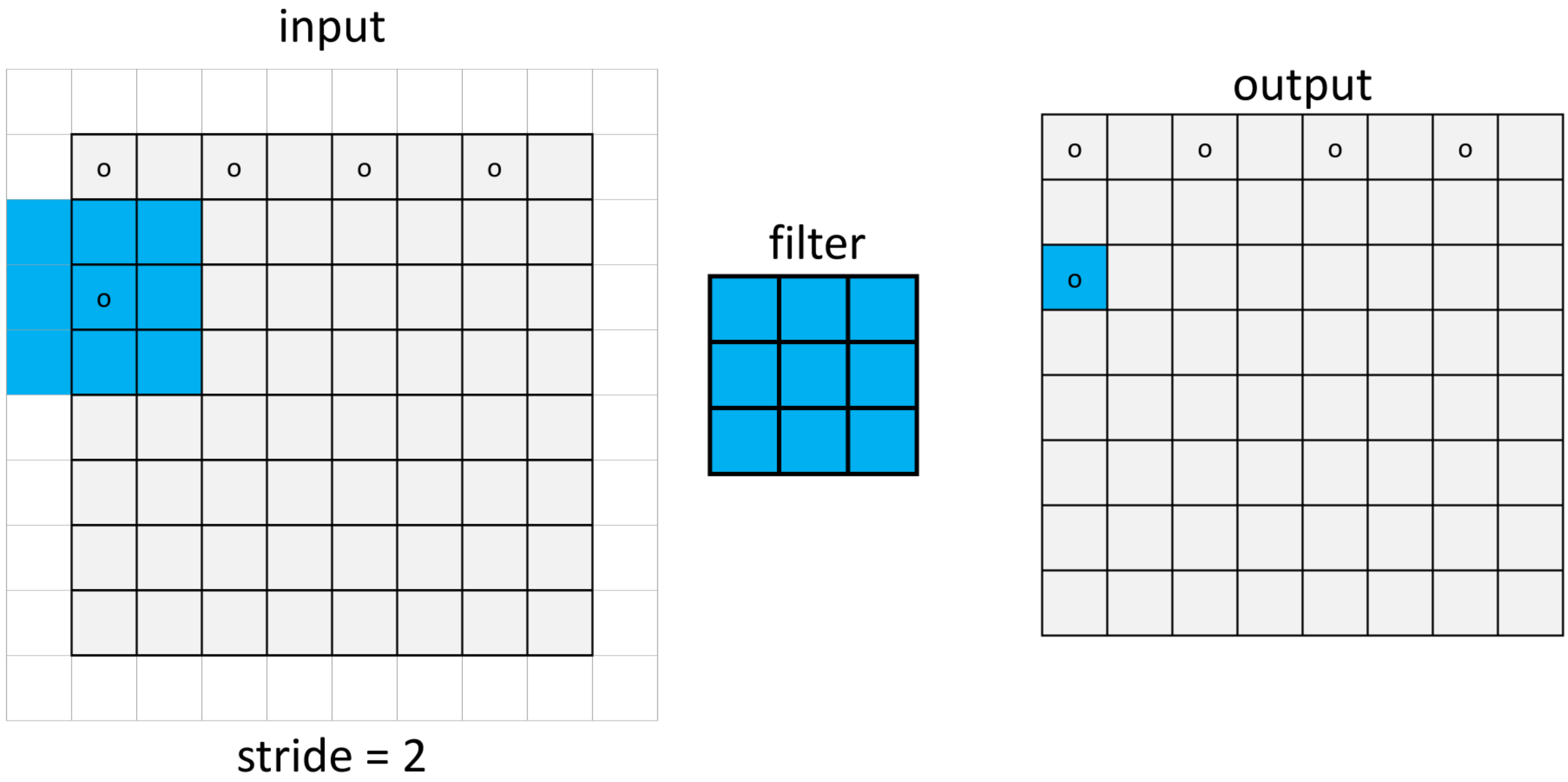
# Convolution: stride



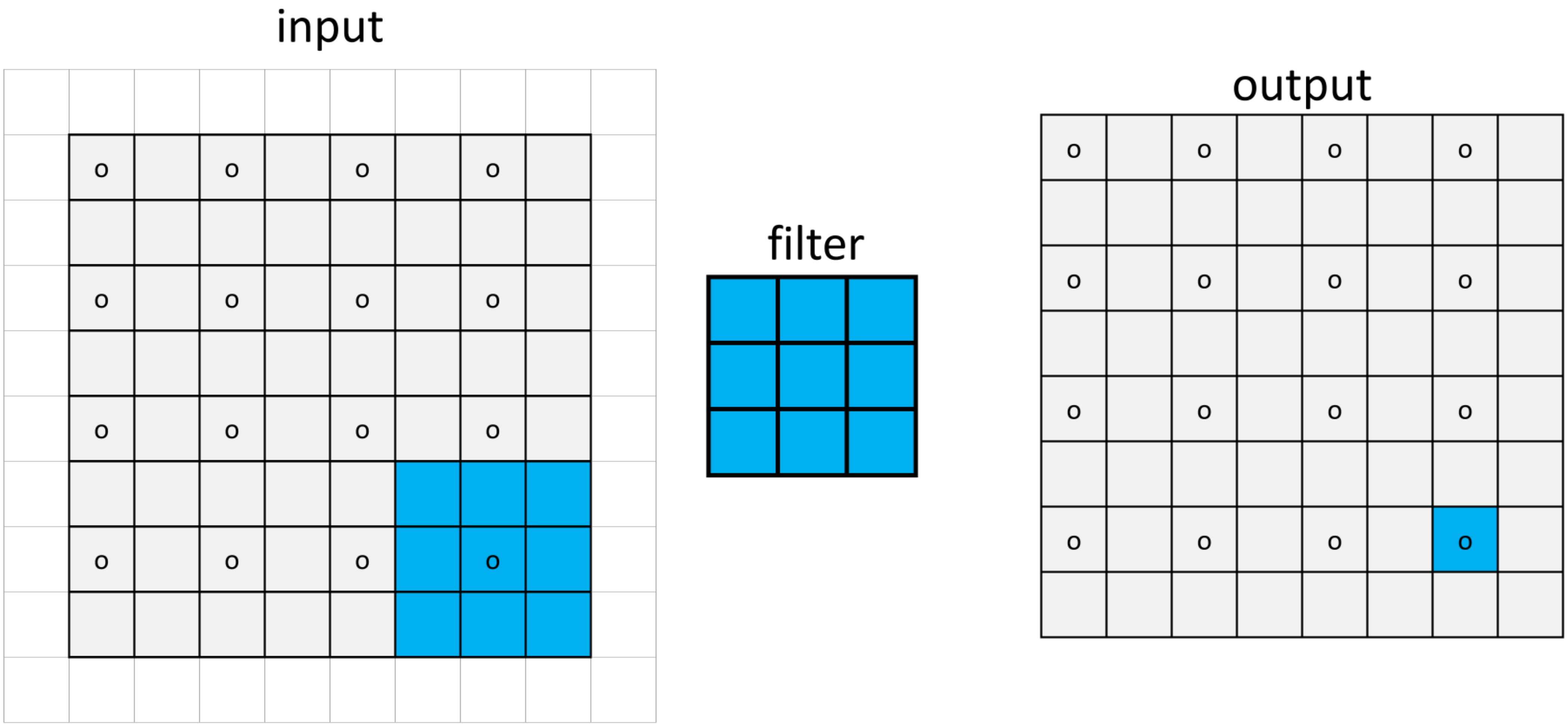
# Convolution: stride



# Convolution: stride

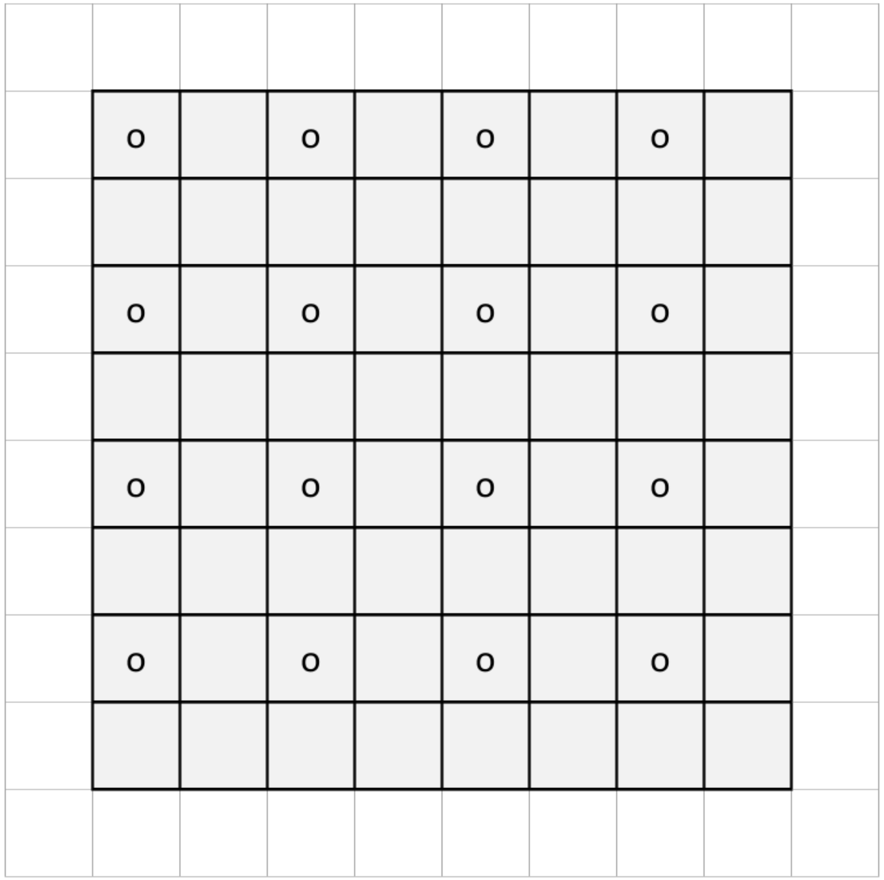


# Convolution: stride



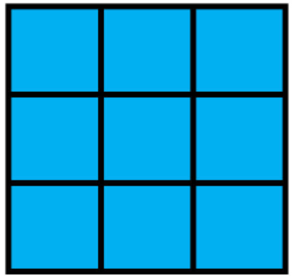
# Convolution: stride

input: $H \times W = 8 \times 8$

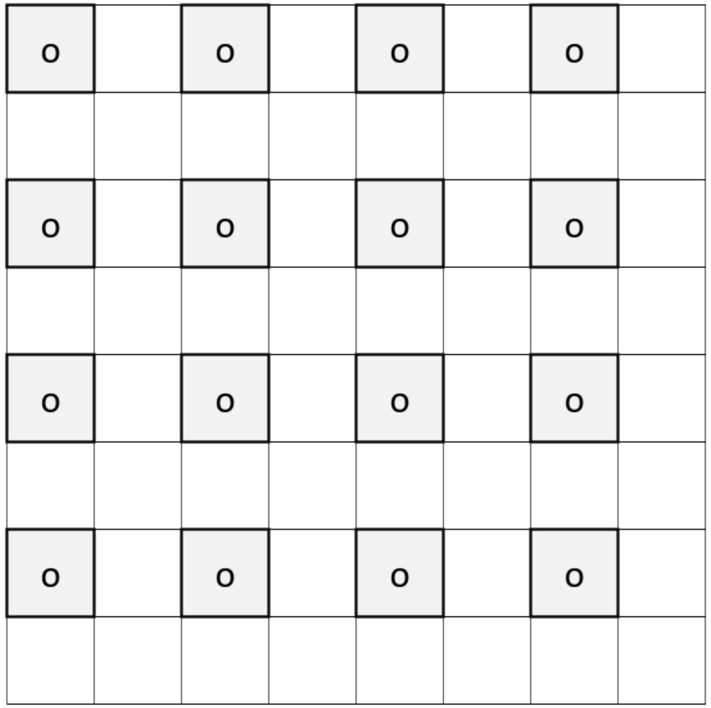


stride = 2

filter

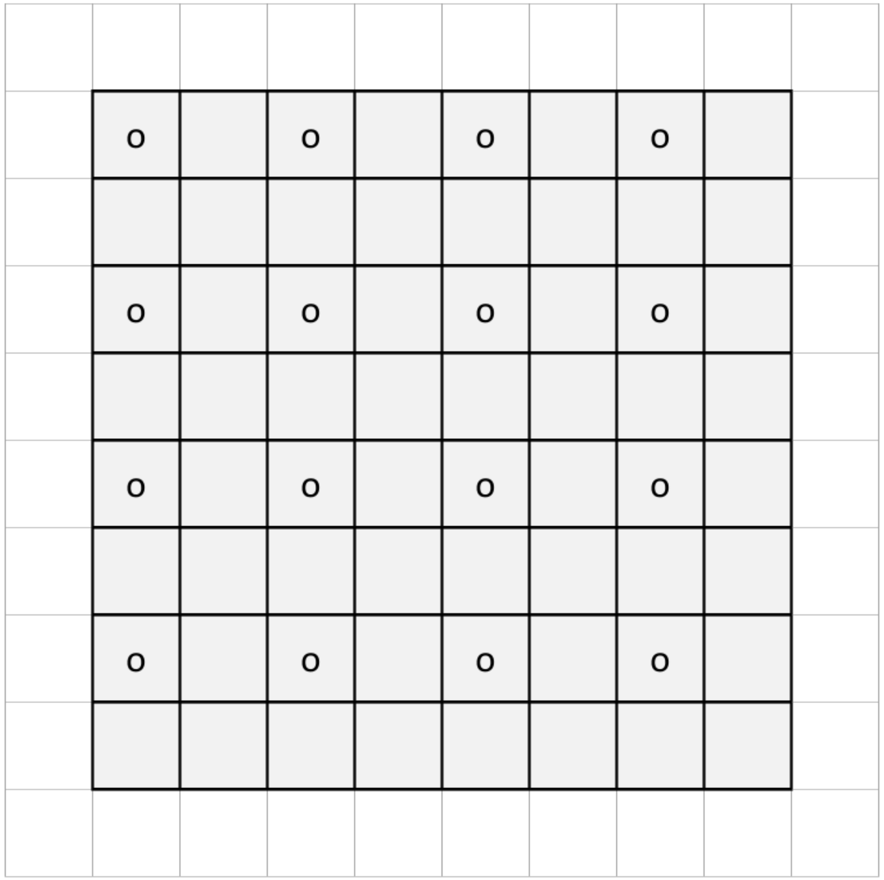


output: $H \times W = \mathbf{4 \times 4}$



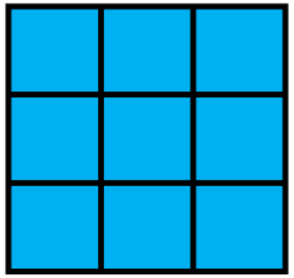
# Convolution: stride

input: $H \times W = 8 \times 8$

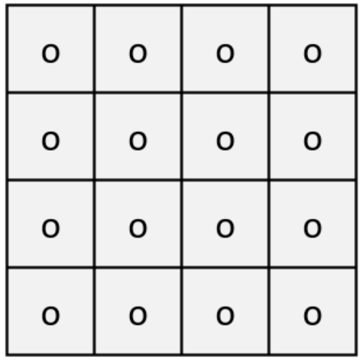



stride = 2

filter



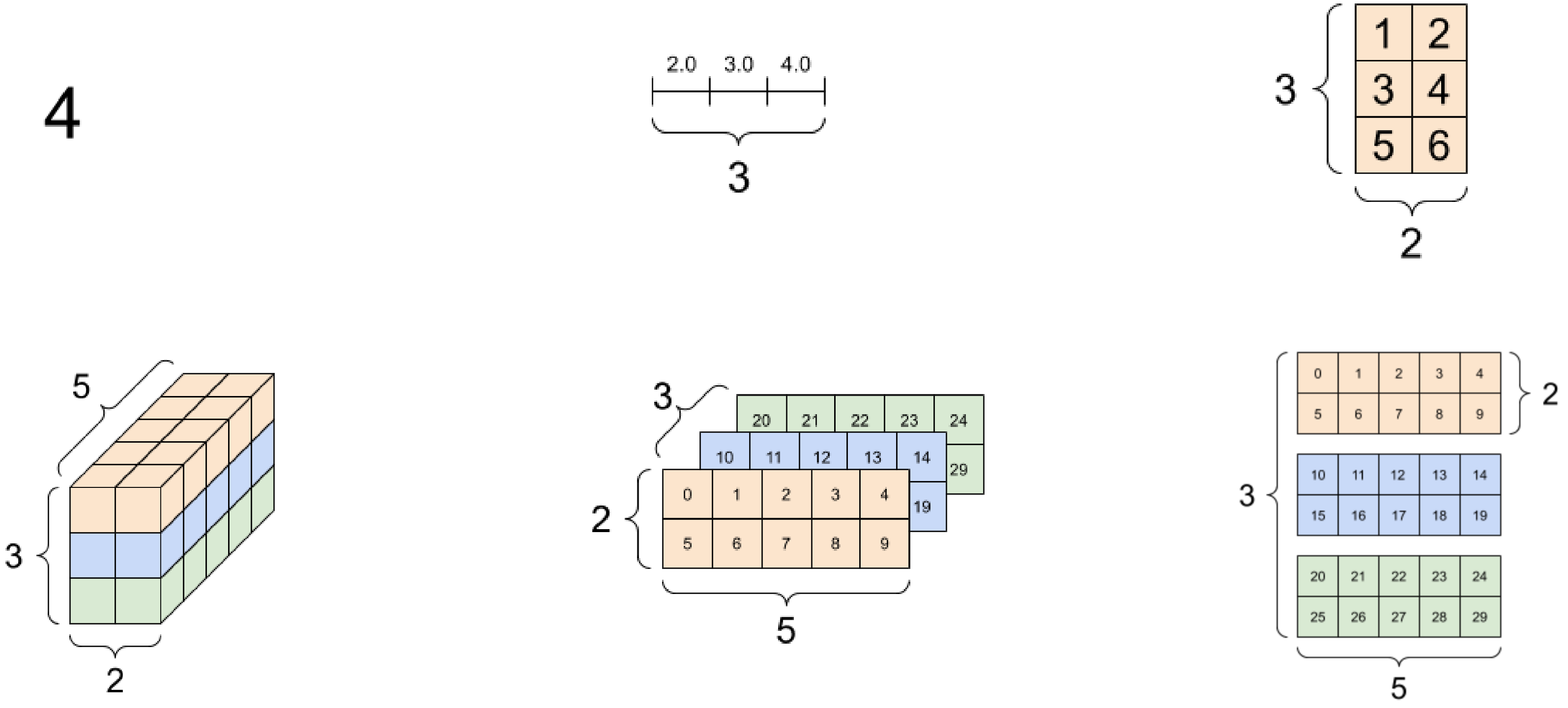
output: $H \times W = \mathbf{4 \times 4}$



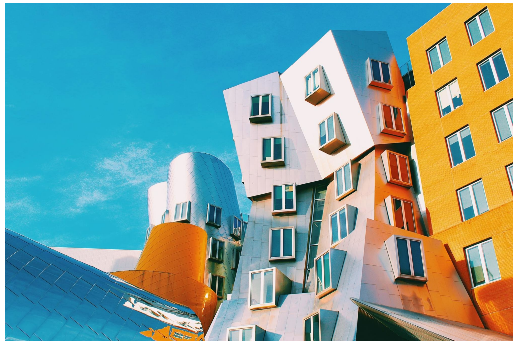
# Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary

# A tender intro to tensor:




[image credit: tensorflow]

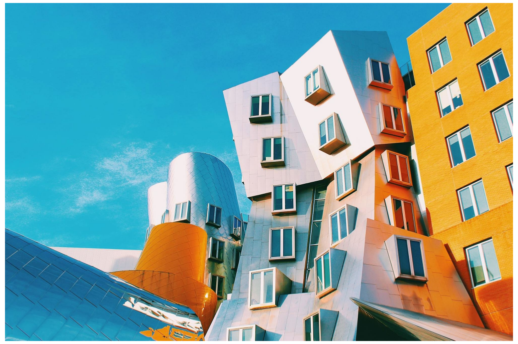


blue

green

red

[Photo by Zayn Shah on Unsplash]




[Photo by Zayn Shah on Unsplash]

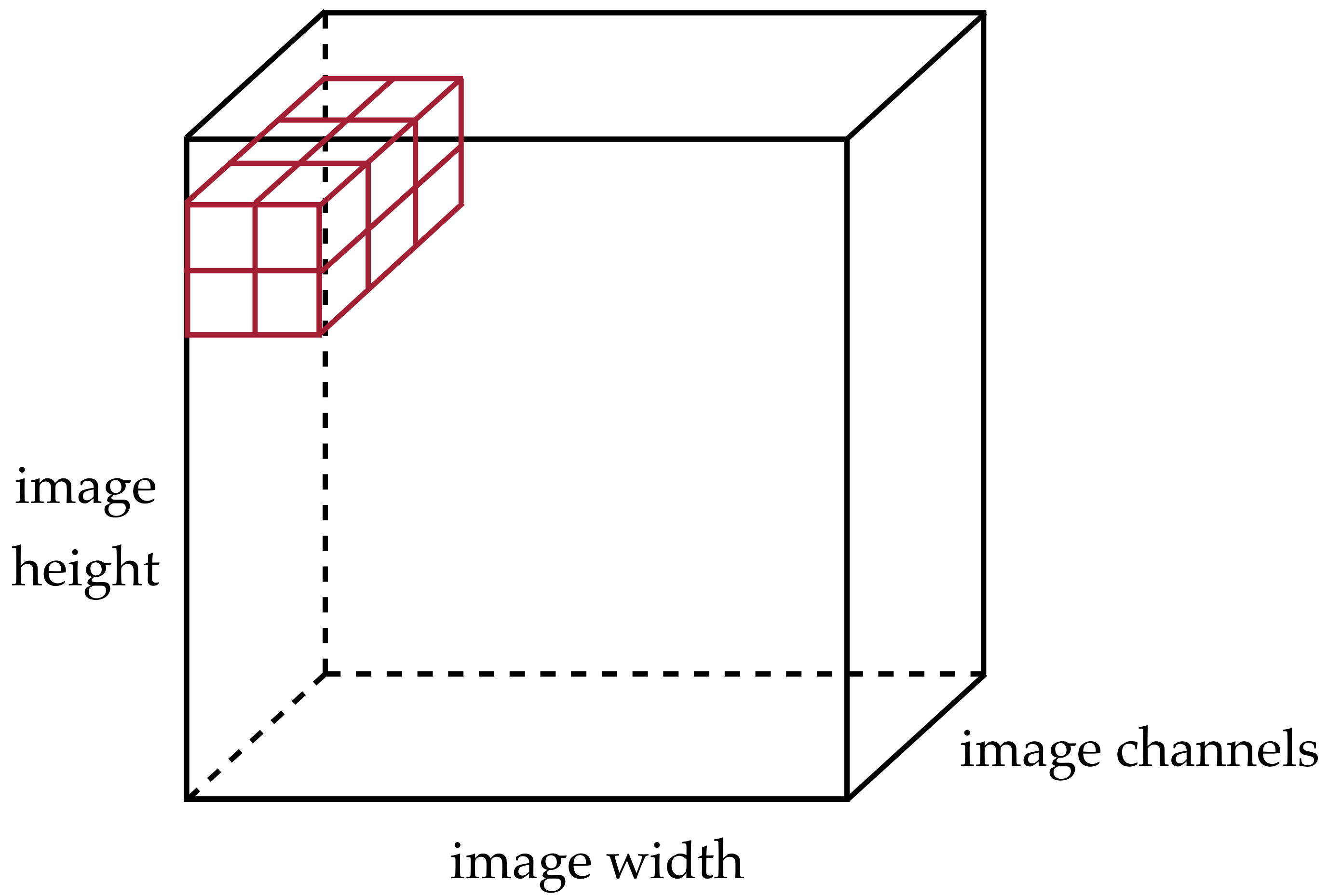
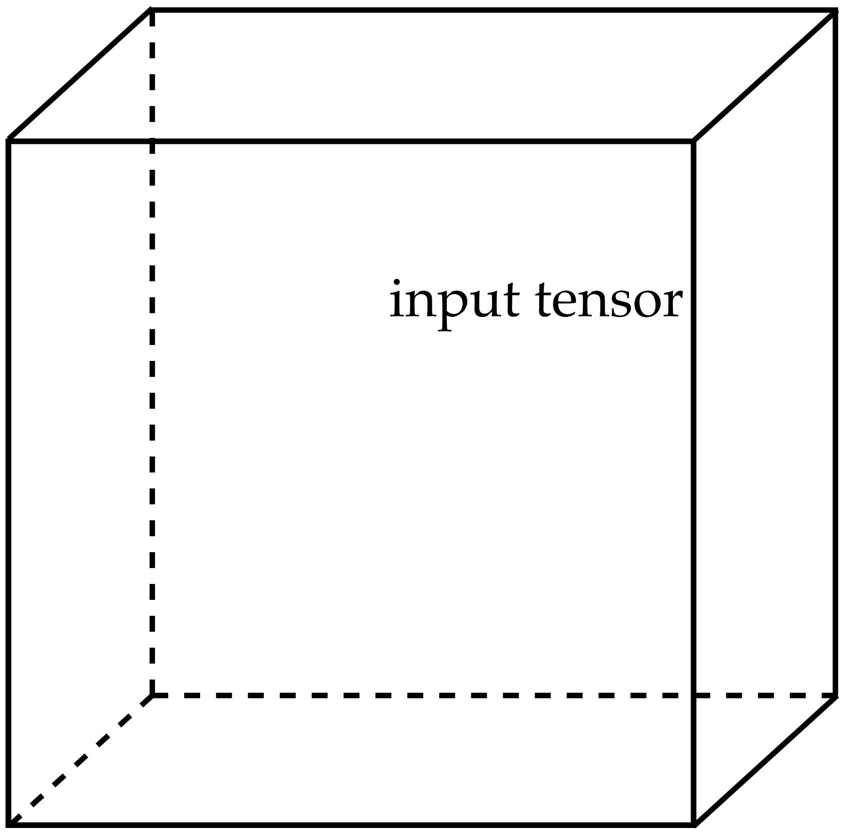
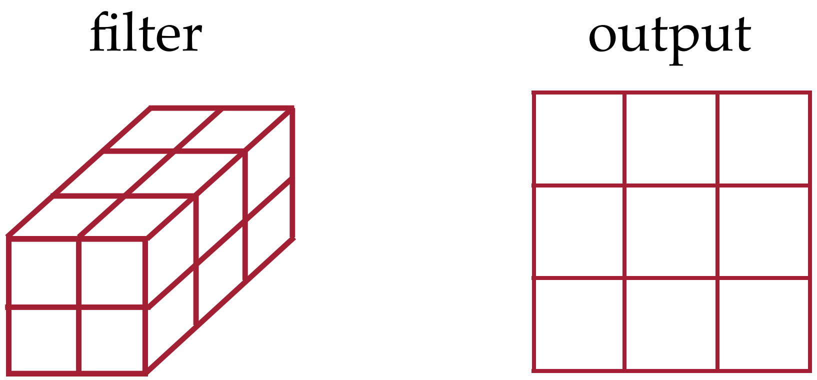
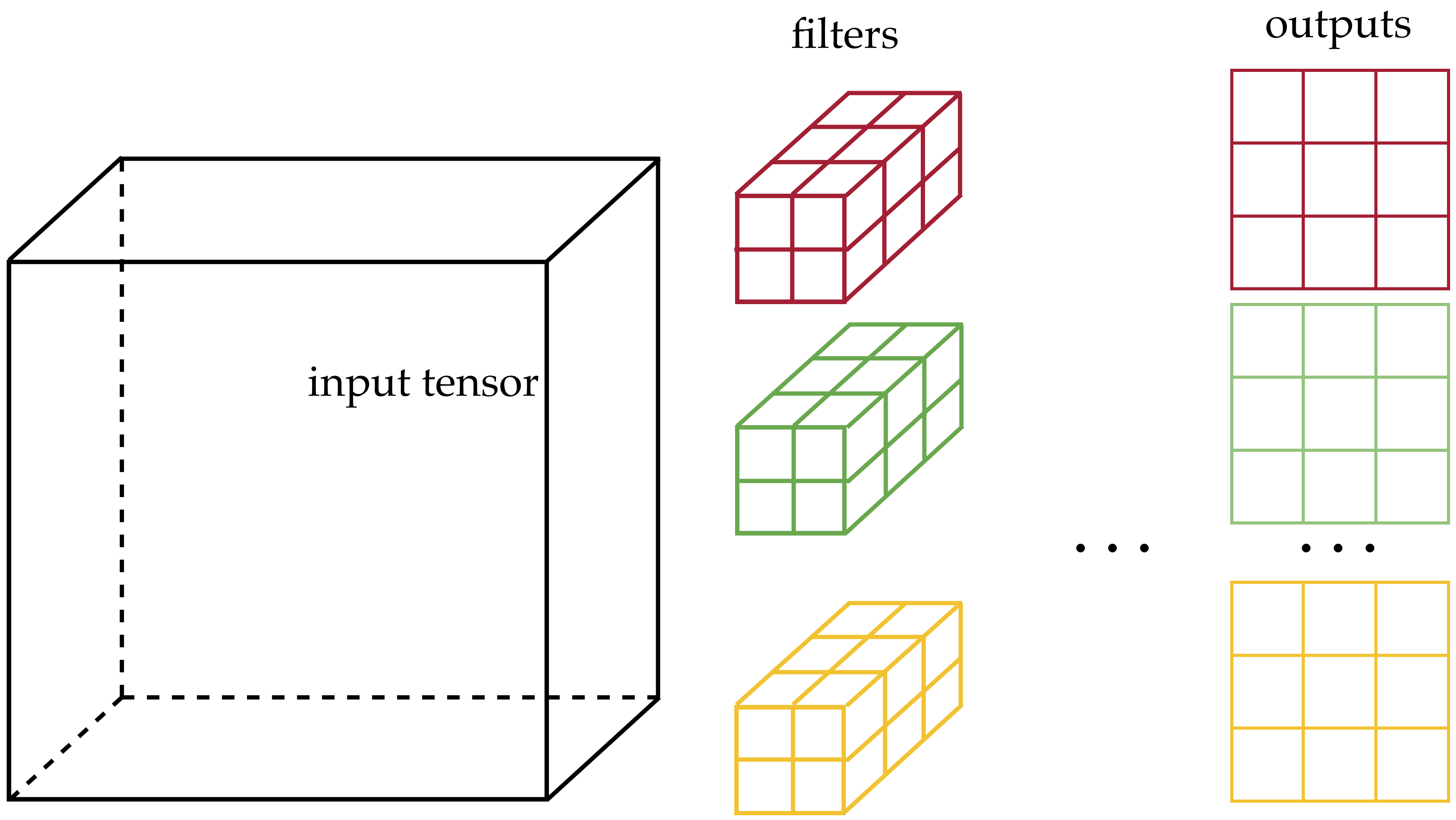


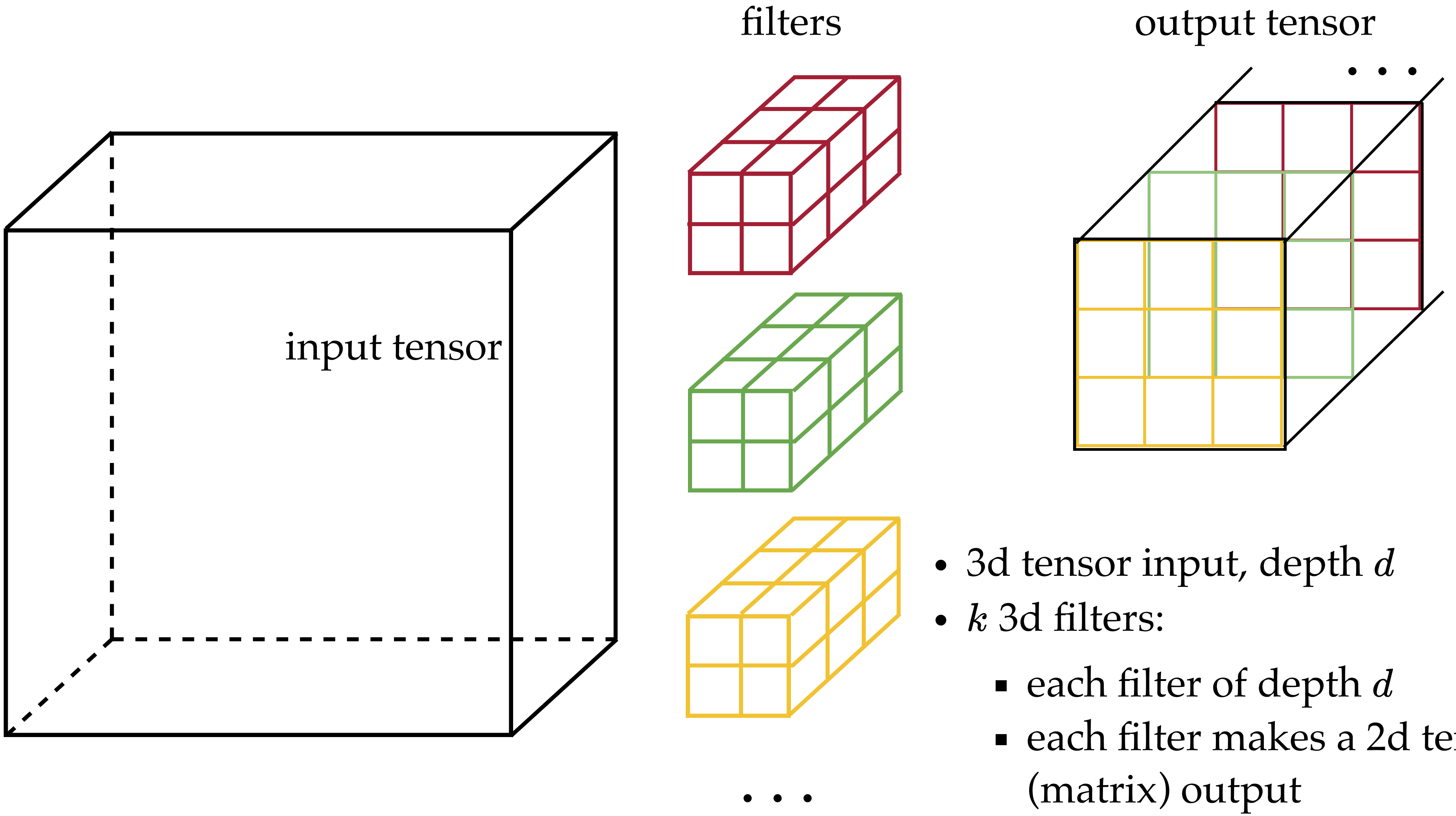
image
height
image channels
image width



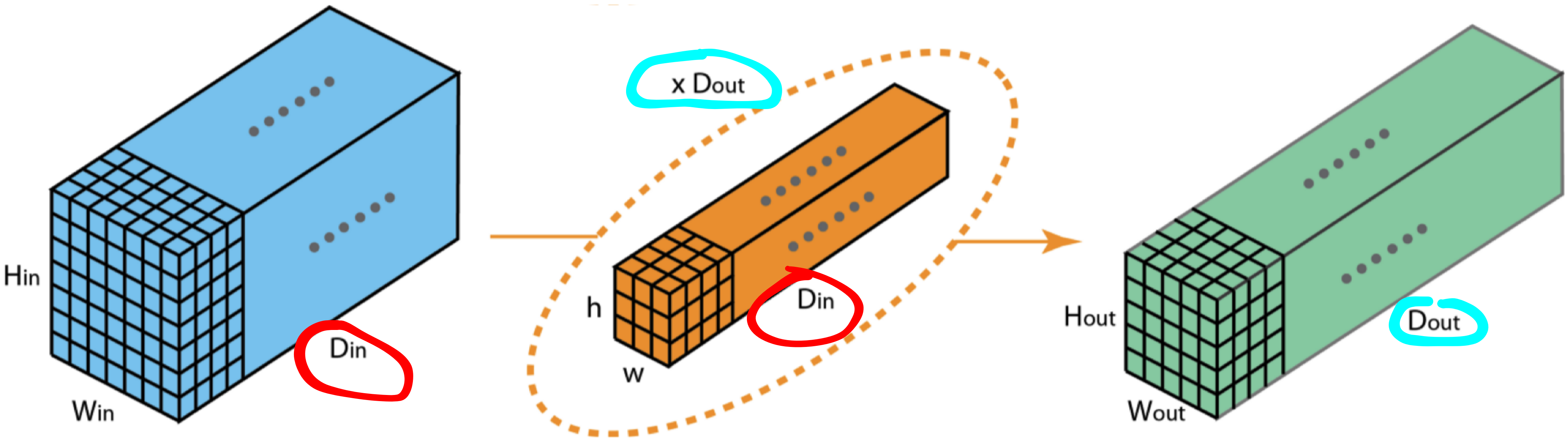




- 3d tensor input, depth $d$
- 3d tensor filter, depth $d$
- 2d tensor (matrix) output


filters
outputs
input tensor



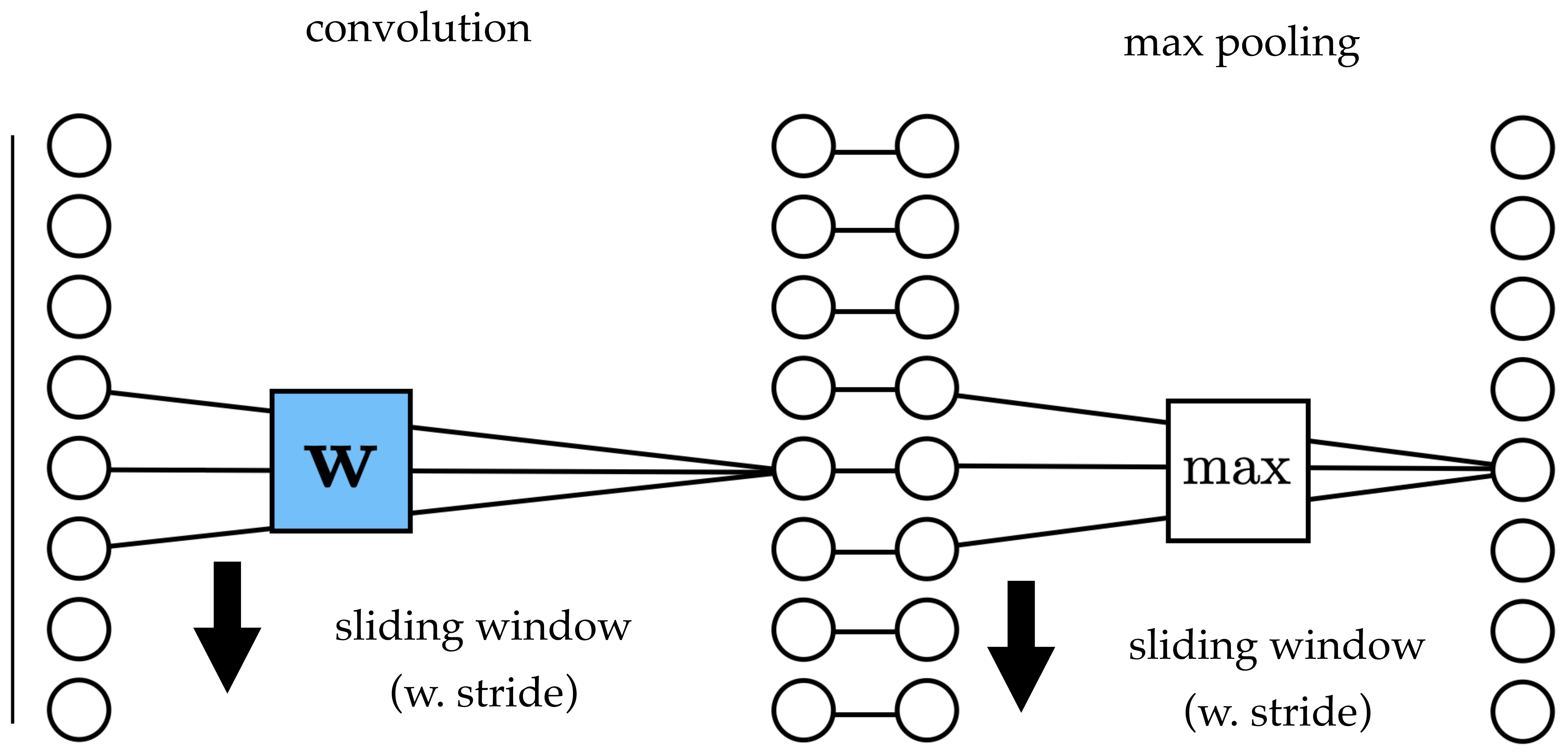

- 3d tensor input, depth $d$
- $k$ 3d filters:
  - each filter of depth $d$
  - each filter makes a 2d tensor (matrix) output
- total output 3d tensor, depth $k$



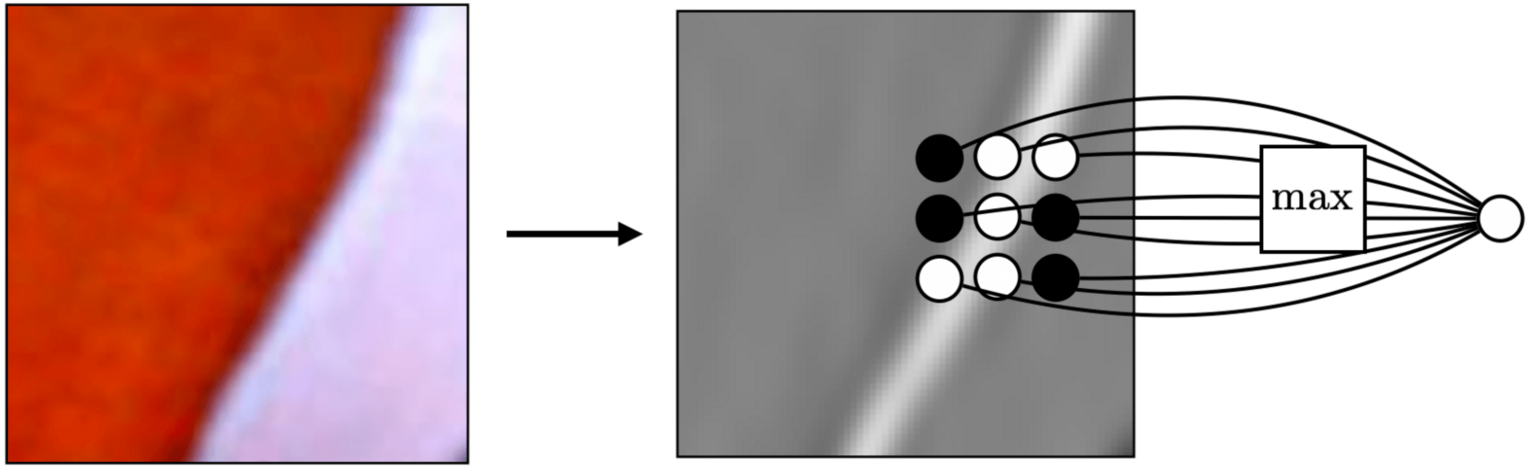

[image credit: medium]

# Outline

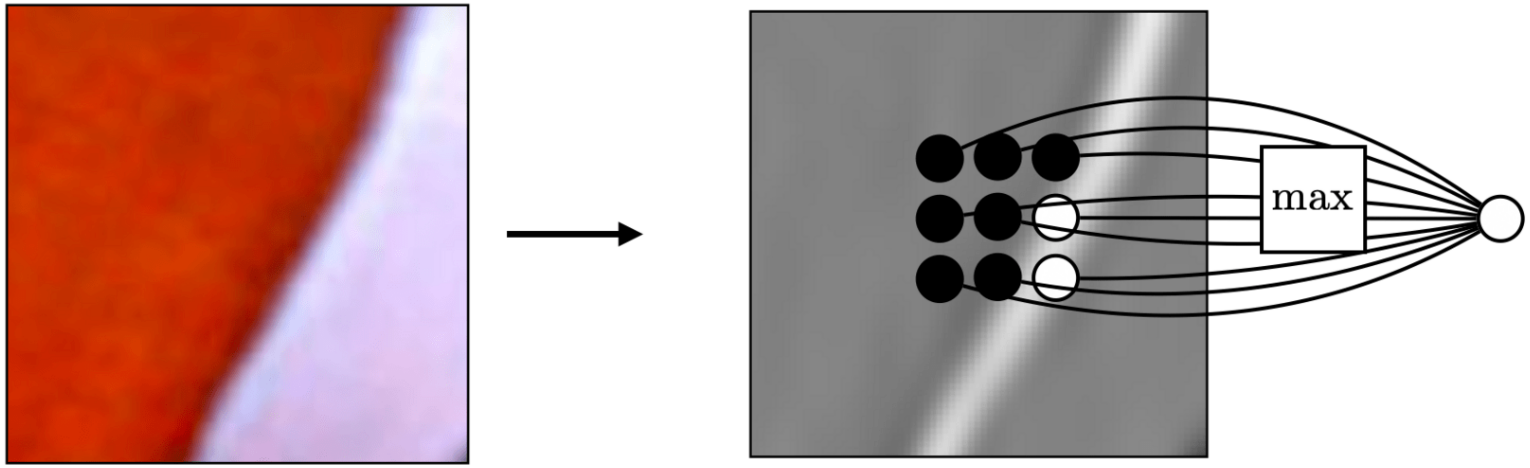
- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary


convolution
max pooling
w
max
sliding window
(w. stride)
sliding window
(w. stride)

Pooling across spatial locations achieves stability w.r.t. small translations:

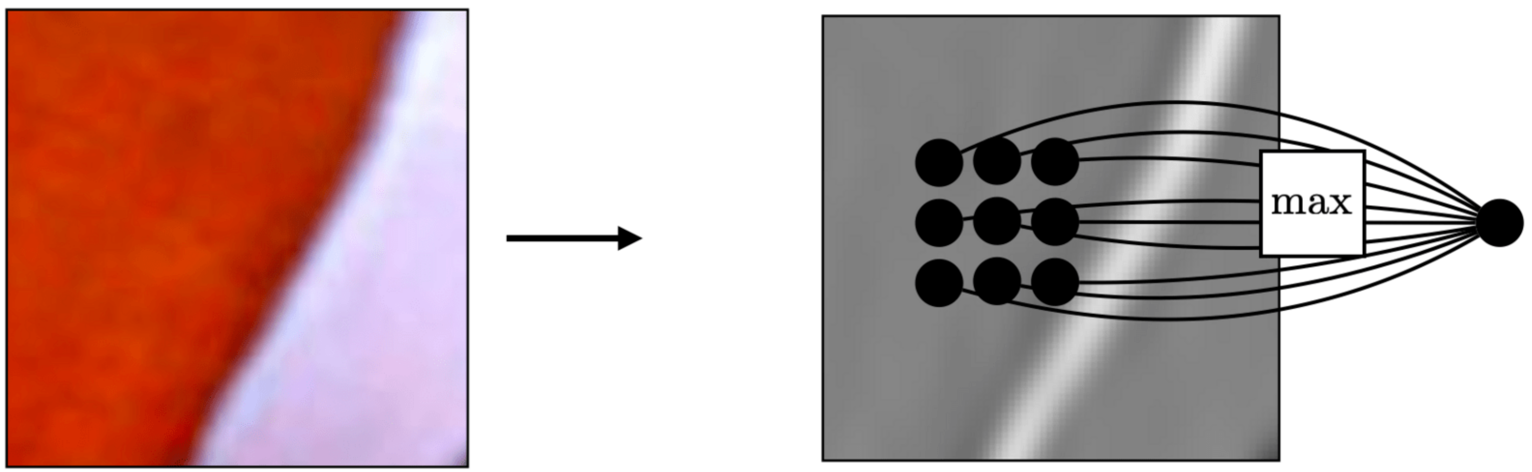


Pooling across spatial locations achieves stability w.r.t. small translations:



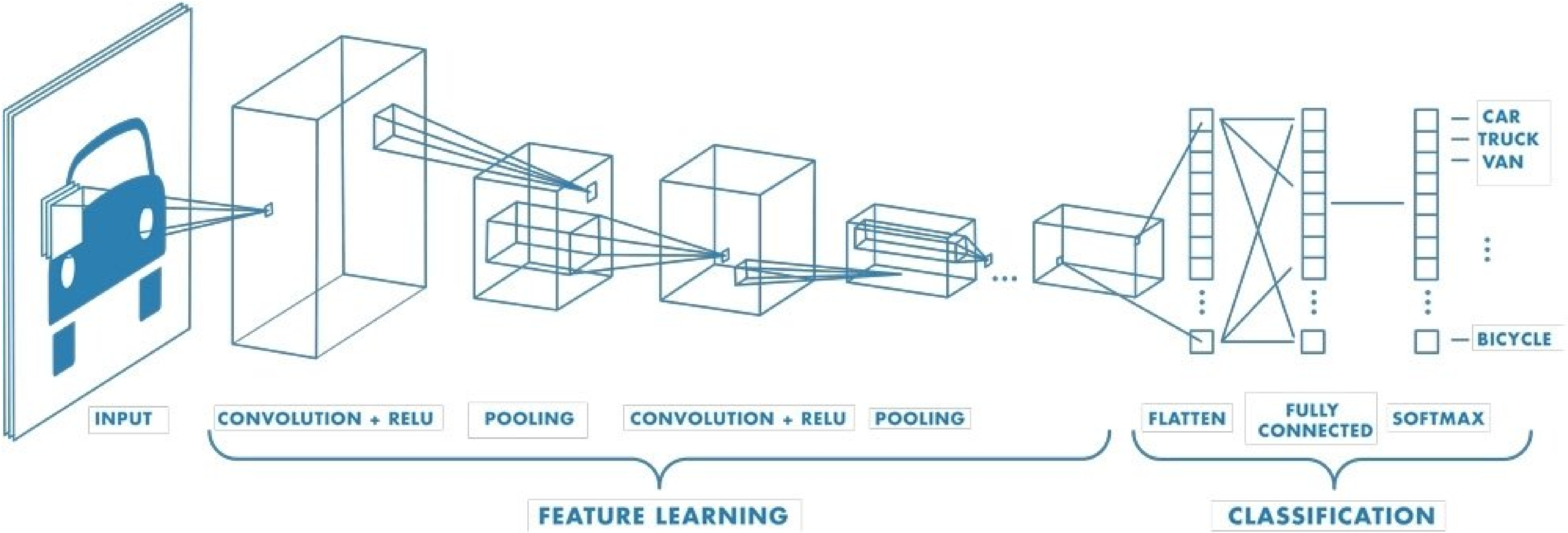

large response regardless of exact position of edge

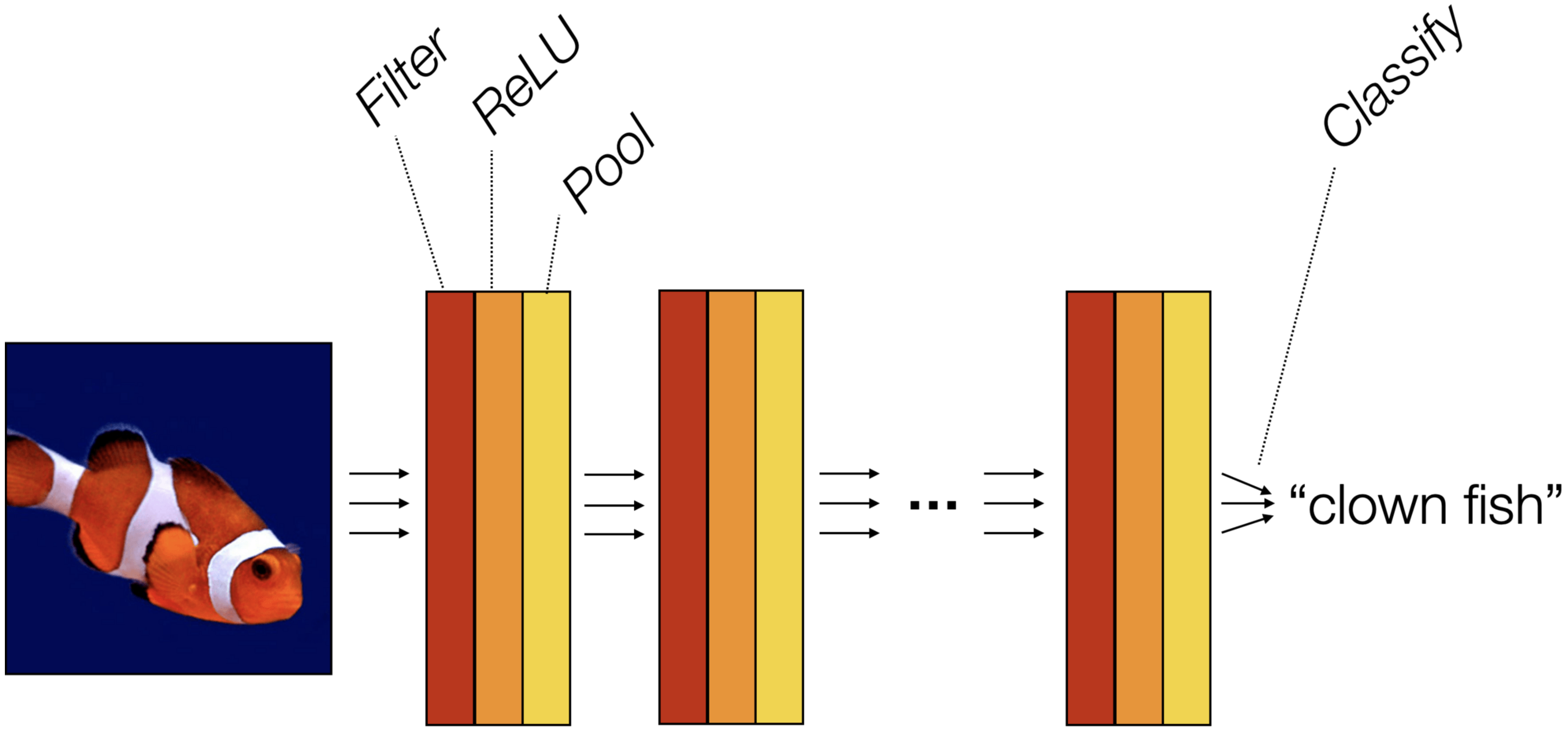
Pooling across spatial locations achieves stability w.r.t. small translations:



# Outline

- Recap (fully-connected net)
- Motivation and big picture ideas of CNN
- Convolution operation
  - 1d and 2d convolution mechanics
  - interpretation:
    - local connectivity
    - weight sharing
  - 3d tensors
- Max pooling
  - Larger window
- Typical architecture and summary


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


Filter
ReLU
Pool
Classify
...
“clown fish”

We'd love it for you to share some lecture feedback.

# Thanks!