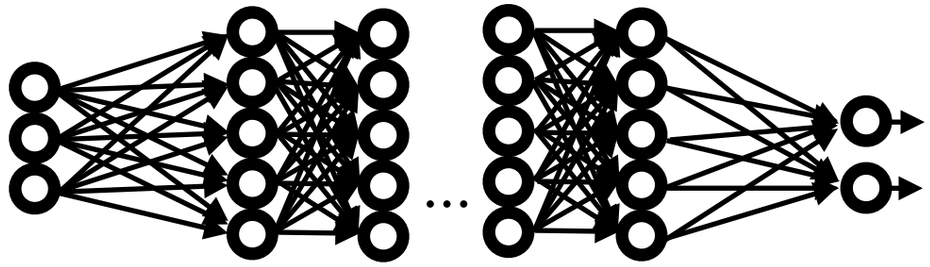
Artificial Intelligence

Taro Sekiyama

National Institute of Informatics (NII)
sekiyama@nii.ac.jp

Review

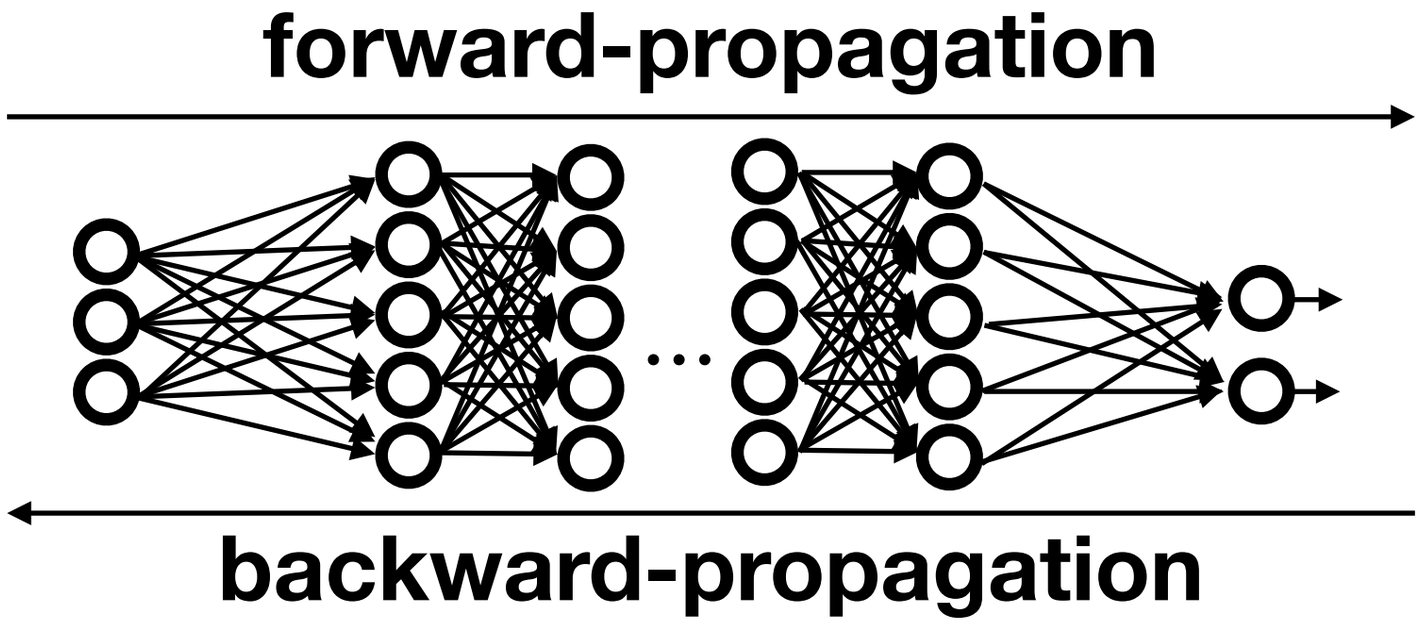
- Neural networks (NNs) are powerful, but difficult-to-amend, ML models
- Simple feedforward NN model



  - Including multiple hidden layers (called ***fully connected***)
  - Neuron: basic computational unit
    - Linear transformation + activation function

Review

- Training of NNs is to optimize parameters of neurons
  - The gradient descent algorithm to optimize is called ***back-propagation***



Review

- The vanishing gradient problem
  - Gradients in deeper NNs tend to approach to zero
- Important to strike a balance between expressivity and trainability

Challenges of NNs

1. Hard to find optimal parameters even by gradient descent
   - The expressivity of FNNs makes their cost functions non-convex
   - Important to design ***features*** that makes optimization easier

2. How do we handle sequential data without the vanishing gradient problem?

This lecture

Introducing two special types of NNs

- Convolutional neural networks (CNNs)
  - Specialized in image processing
  - Good at extracting features from images
- Recurrent neural networks (RNNs)
  - Specialized in sequential data
  - Relaxing the vanishing gradient problem

Success of CNNs

- Image recognition

Images from: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

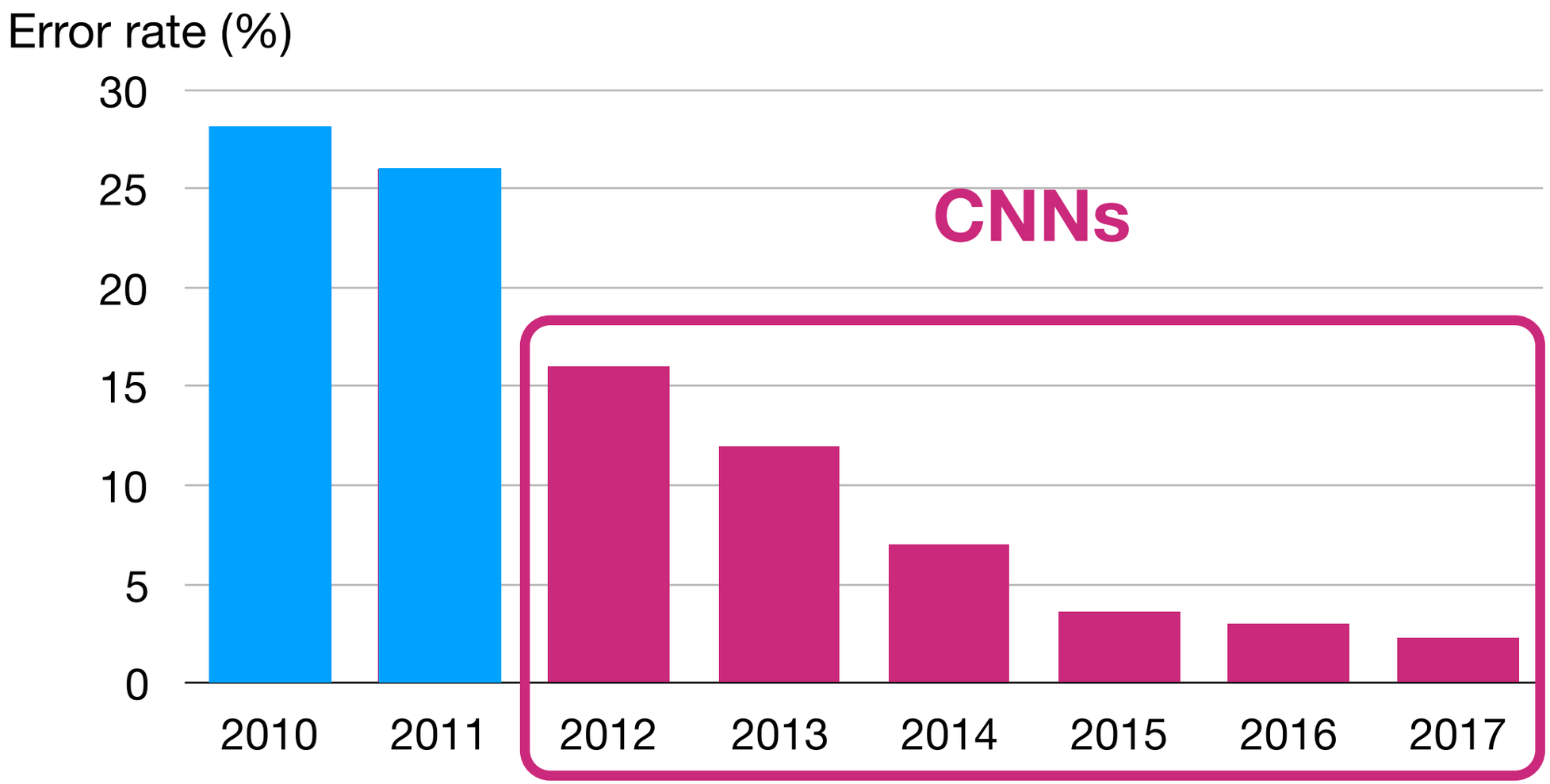
Success of CNNs

- Image recognition

Images from: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Success of CNNs

- CNNs have be winning in ILSVRC (ImageNet Large Scale Visual Recognition Competition) since 2012



Success of CNNs

- Object detection

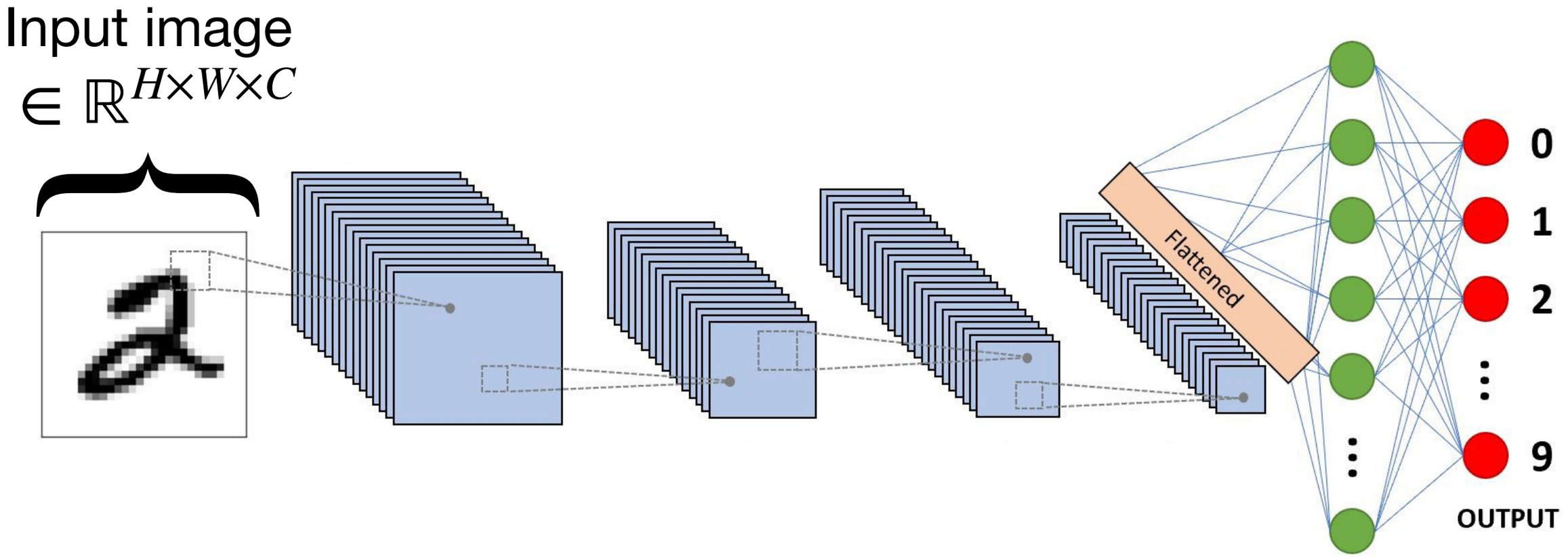
https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606

Success of CNNs

- Image generation/transformation

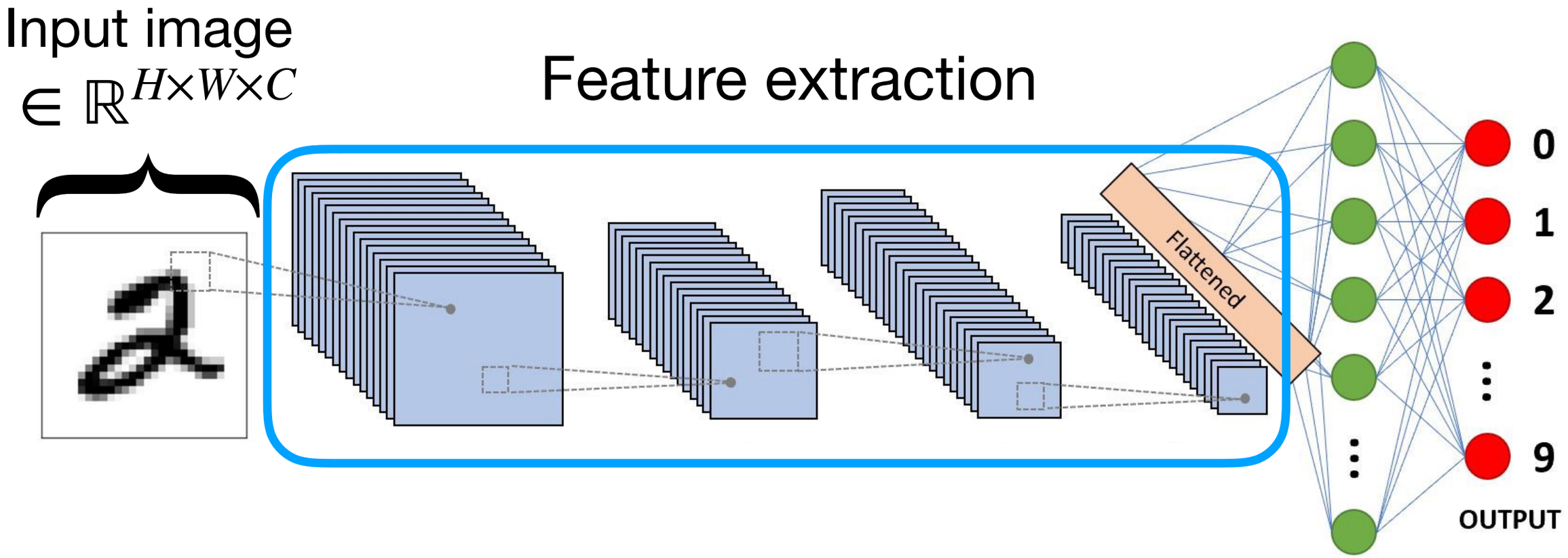
https://twitter.com/pixivsketch/status/1405475308108759042/photo/1

Convolution neural networks



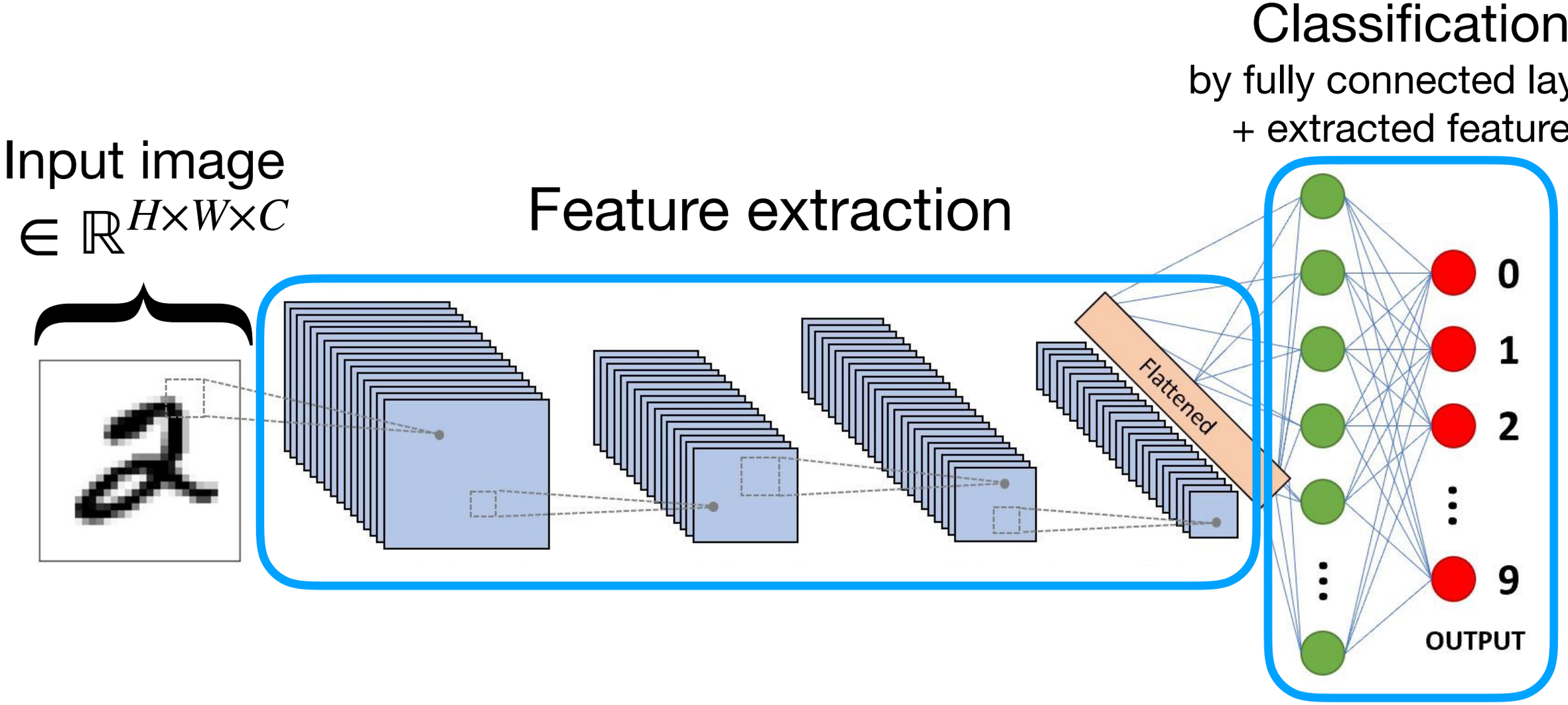
The image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolution neural networks



The image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolution neural networks



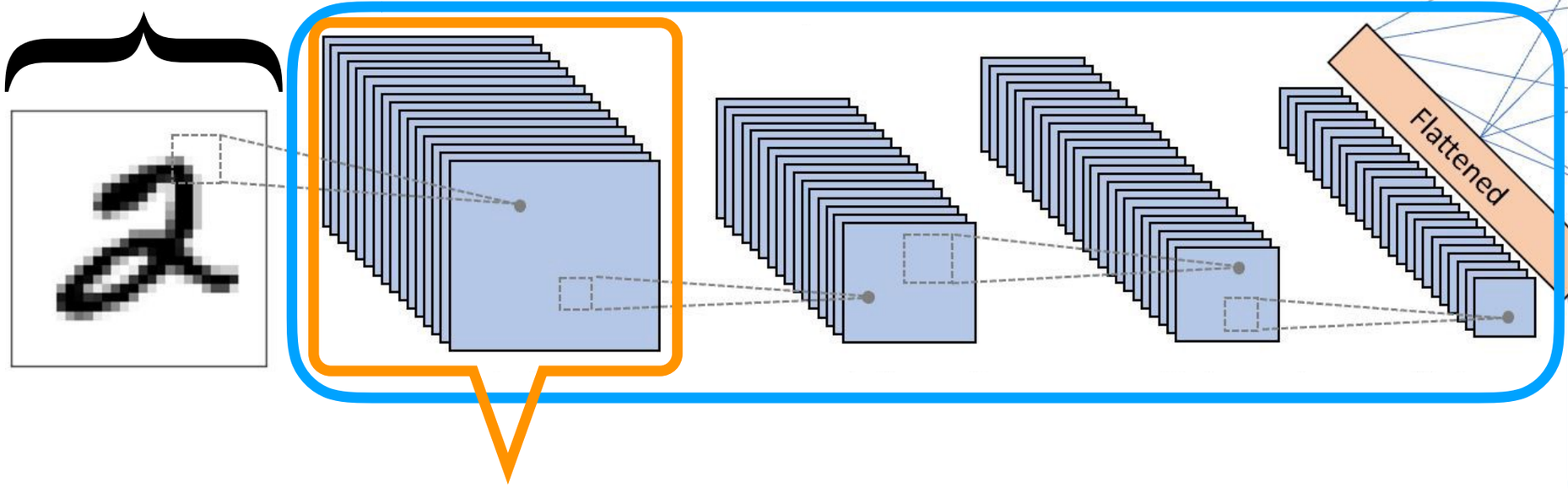
The image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

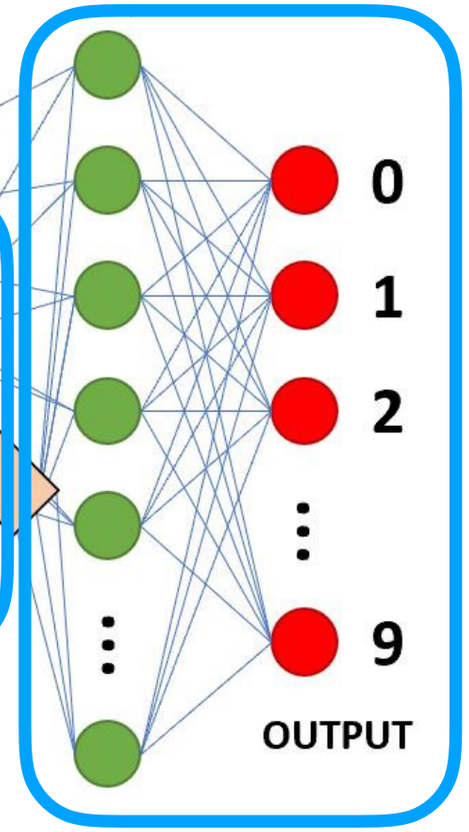
Convolution neural networks

Classification
by fully connected layers
+ extracted features

Input image
$\in \mathbb{R}^{H \times W \times C}$

Feature extraction



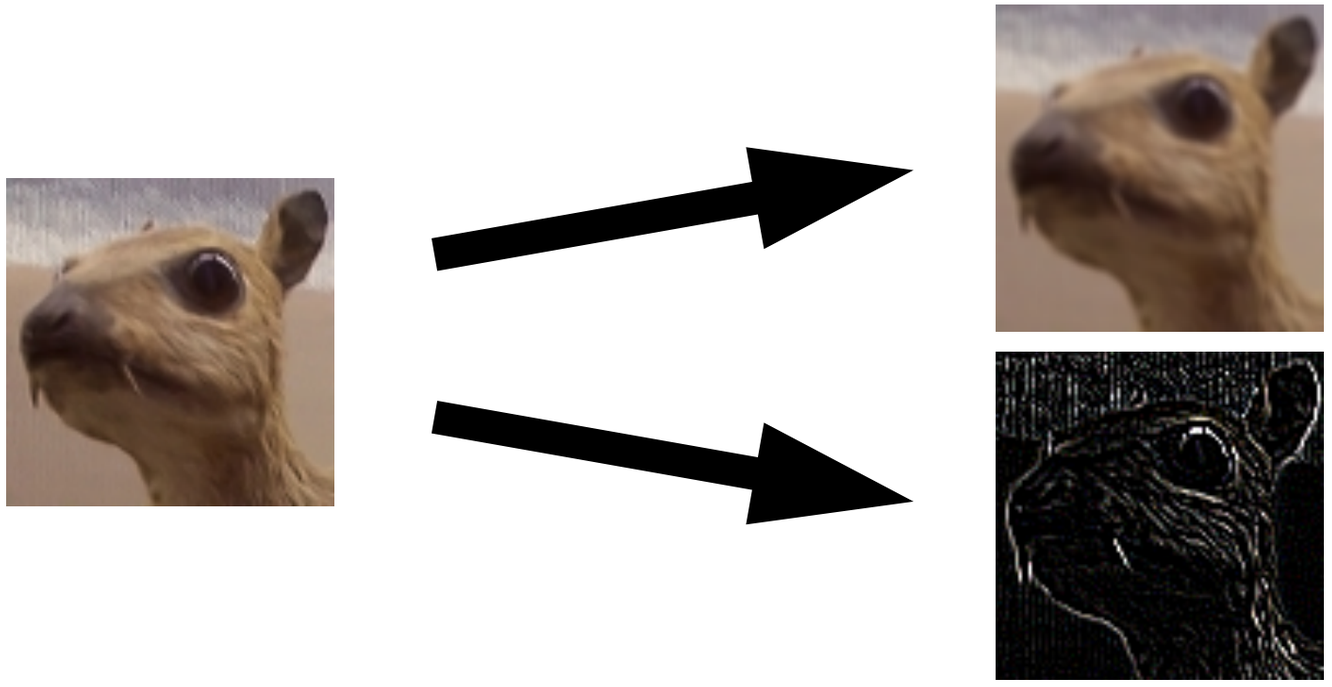


Features extracted
by **a convolutional + pooling layer**

The image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolution

- Image transformation by filtering



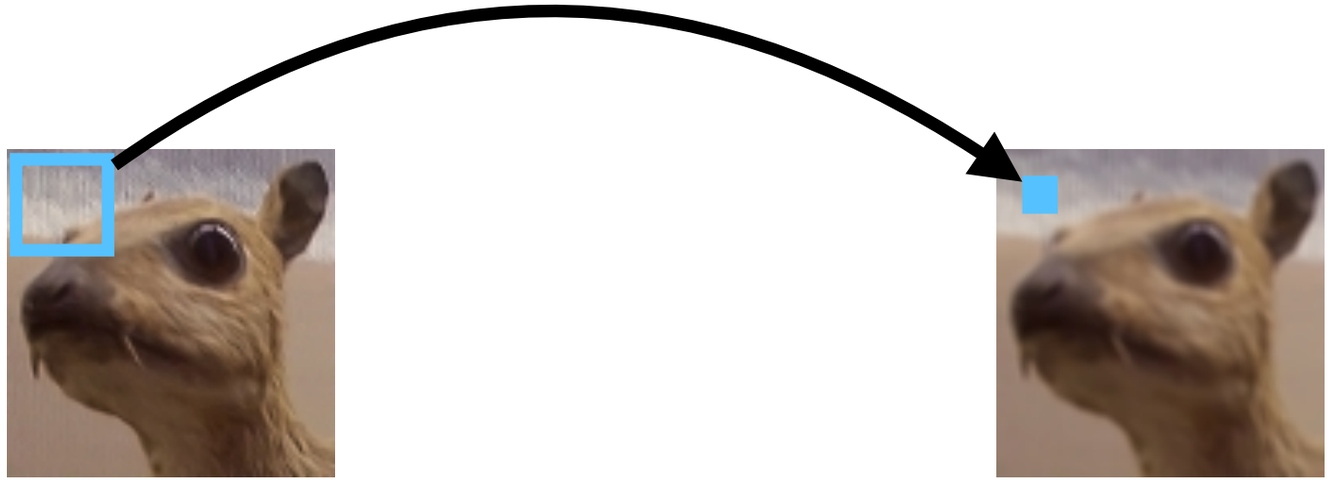
with different parameters

- Convolutional layers learns transformation parameters that produce image features useful for classification

Images from: https://en.wikipedia.org/wiki/Kernel_(image_processing)

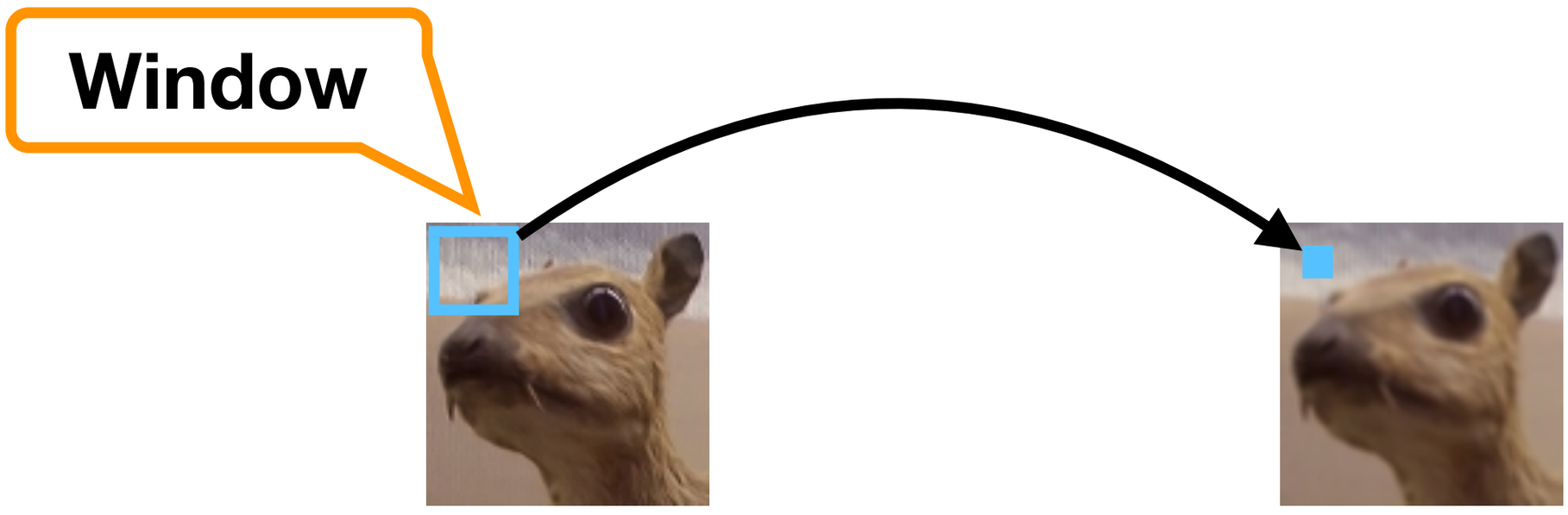
Convolution

- Each pixel of the output image is computed with a small region of the input image



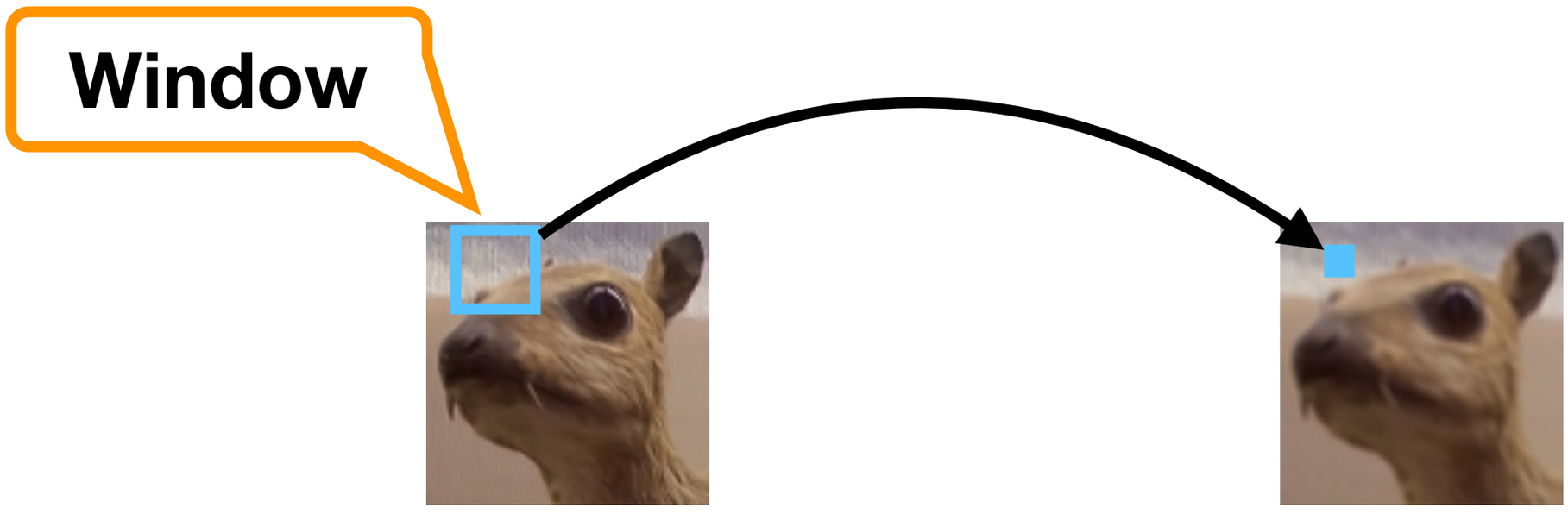
Convolution

- Each pixel of the output image is computed with a small region of the input image



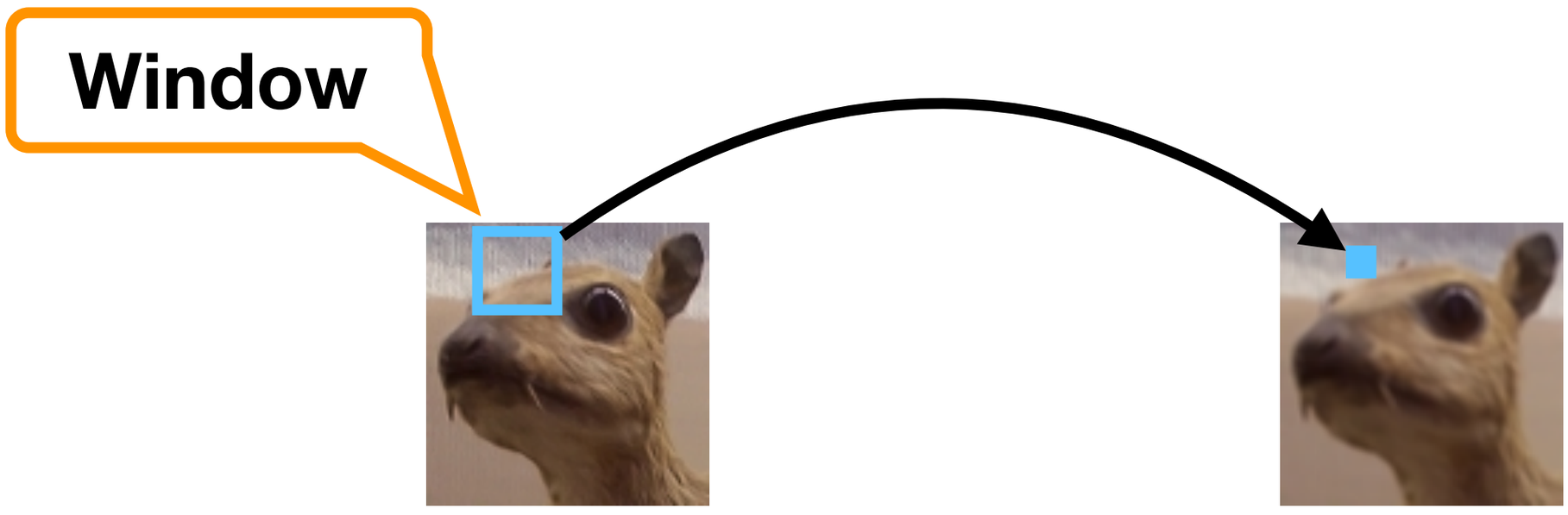
Convolution

- Each pixel of the output image is computed with a small region of the input image



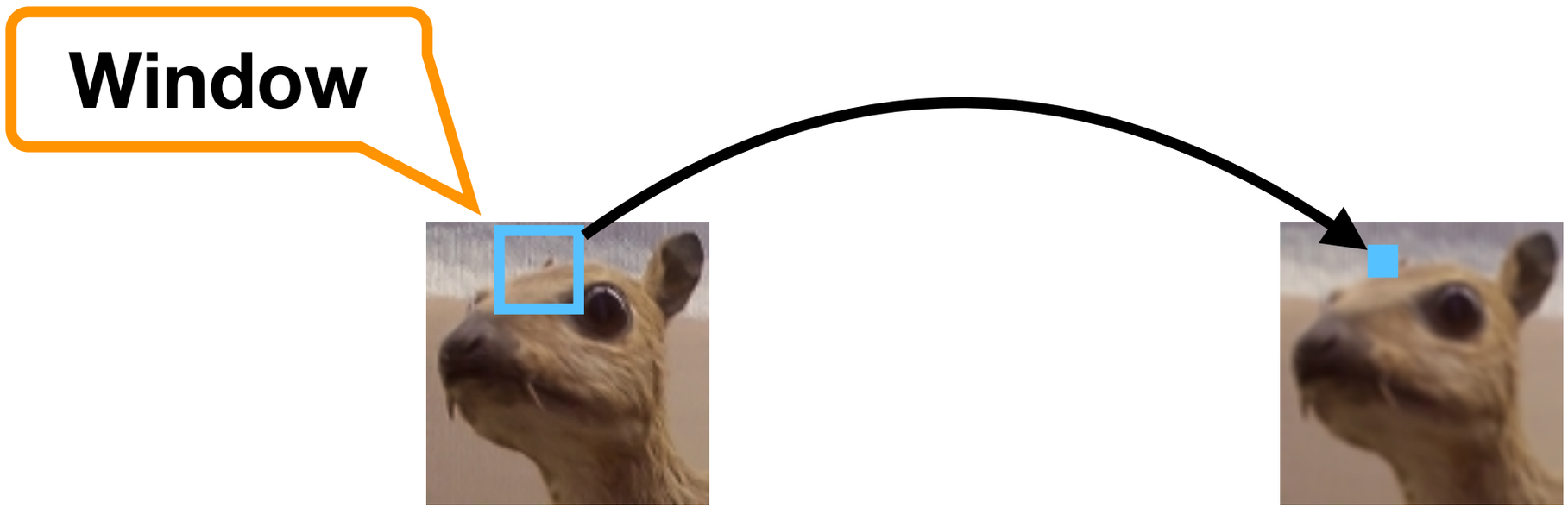
Convolution

- Each pixel of the output image is computed with a small region of the input image



Convolution

- Each pixel of the output image is computed with a small region of the input image



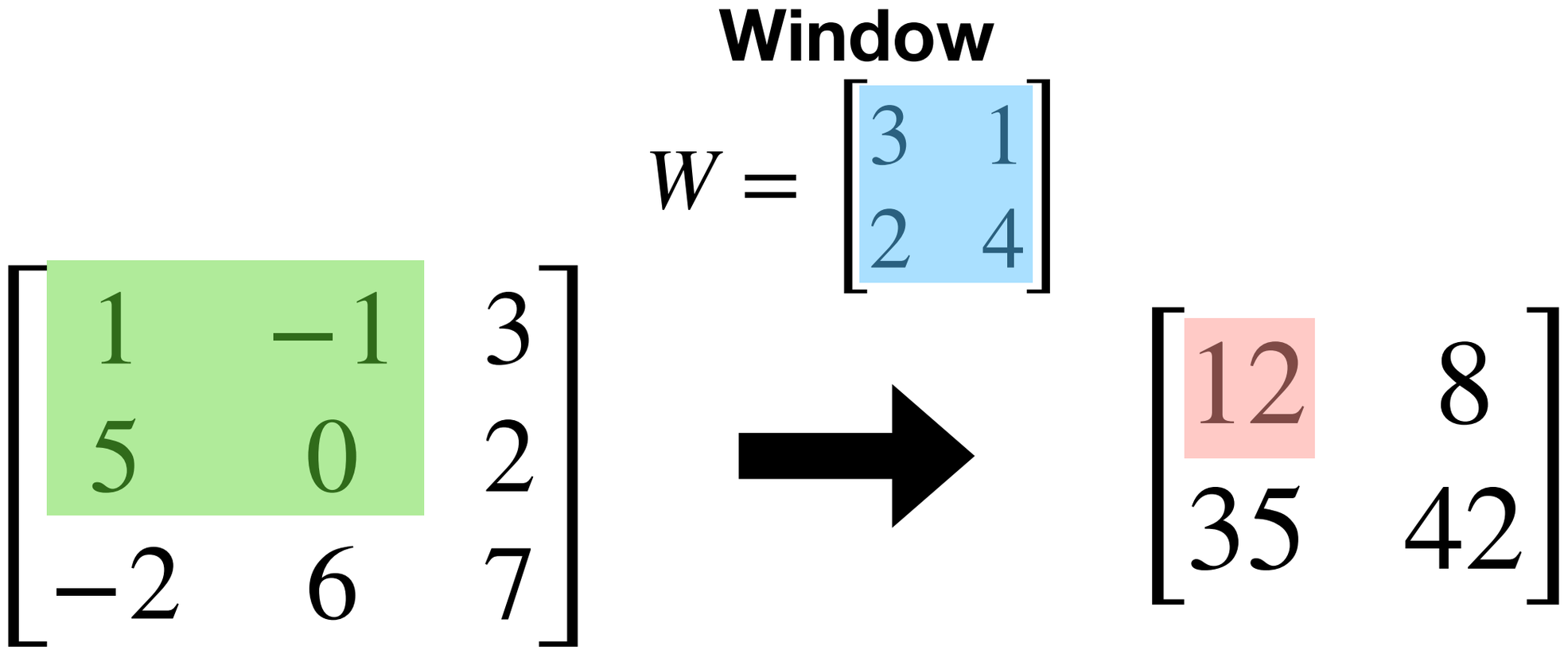
Running example

Window

$$W = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix} \rightarrow \begin{bmatrix} 12 & 8 \\ 35 & 42 \end{bmatrix}$$

Running example



Running exa

Window

$$W = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix} \rightarrow \begin{bmatrix} 12 & 8 \\ 35 & 42 \end{bmatrix}$$

The sum of

values of $\begin{bmatrix} 1 & -1 \\ 5 & 0 \end{bmatrix}$

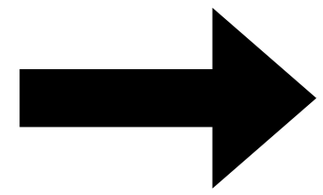
weighted by $\begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$

Running exa

Window

$$W = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$



$$\begin{bmatrix} 12 & 8 \\ 35 & 42 \end{bmatrix}$$

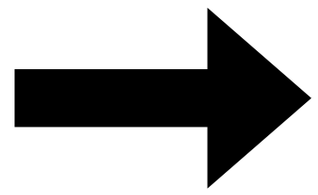
The sum of values of $\begin{bmatrix} -1 & 3 \\ 0 & 2 \end{bmatrix}$ weighted by $\begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$

Running exa

Window

$$W = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$



$$\begin{bmatrix} 12 & 8 \\ 35 & 42 \end{bmatrix}$$

The sum of

values of $\begin{bmatrix} 5 & 0 \\ -2 & 6 \end{bmatrix}$

weighted by $\begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$

Running exa

Window

$$W = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$

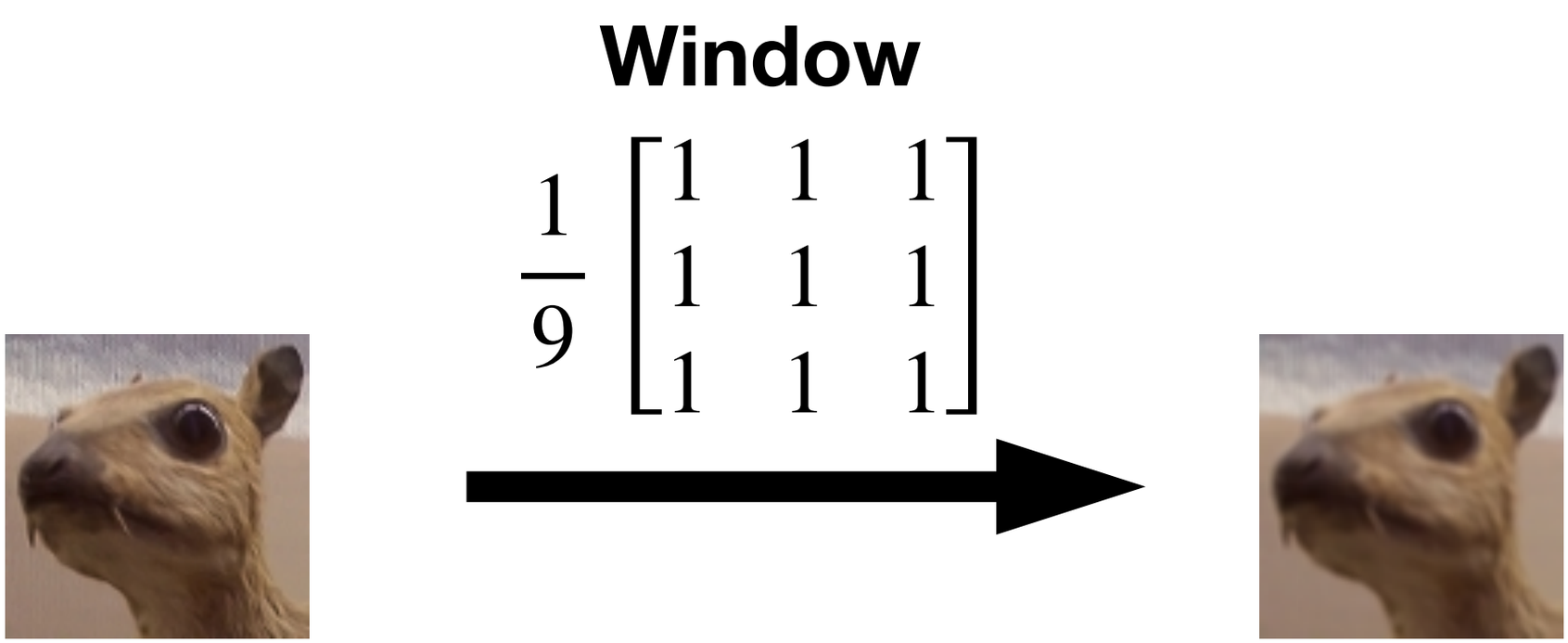
The sum of

values of $\begin{bmatrix} 0 & 2 \\ 6 & 7 \end{bmatrix}$

weighted by $\begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}$

$$\begin{bmatrix} 12 & 8 \\ 35 & 42 \end{bmatrix}$$

Example



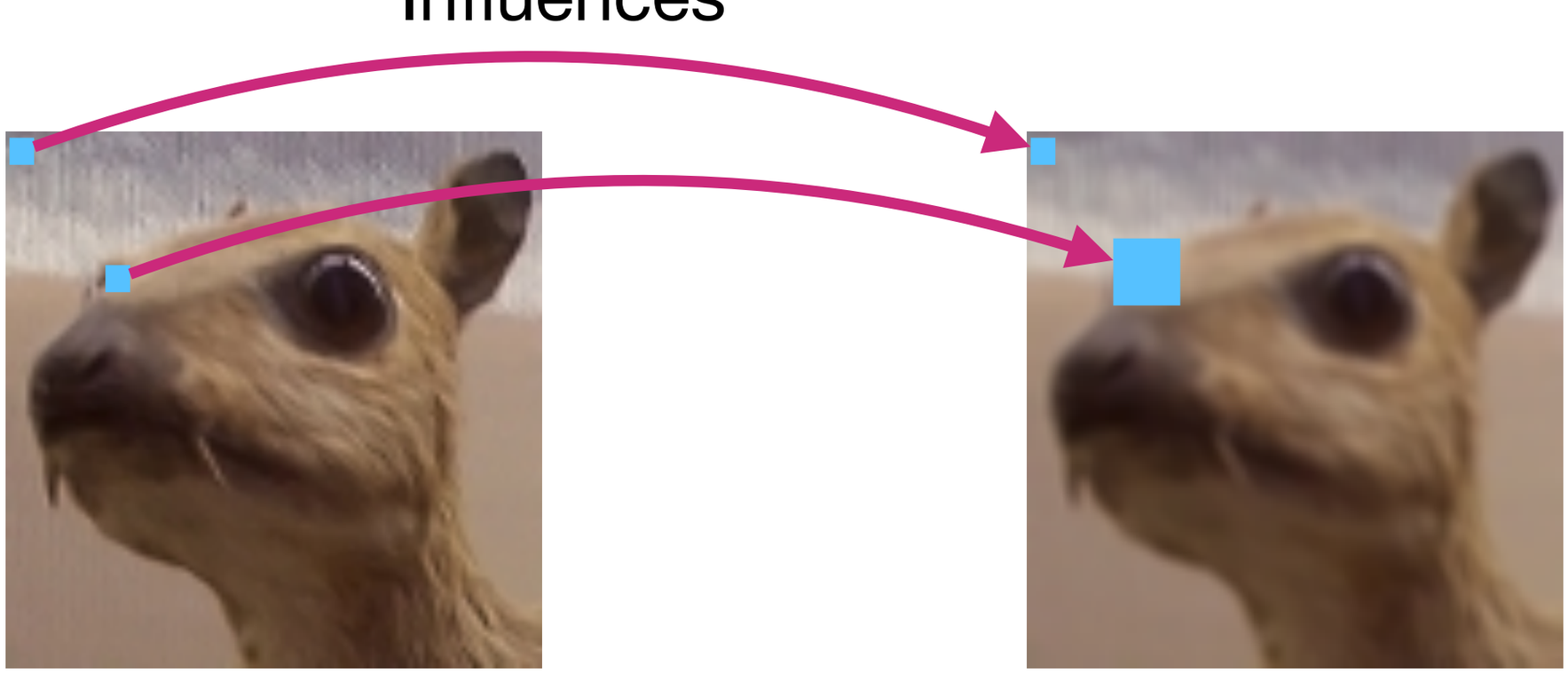
Convolution

- Input
  - Features I : $H \times W \times C$ array
    - H : image height, W : image width, C : the # of features
  - Window K : $h \times w \times c$ array
- Output
  - $(H - h + 1) \times (W - w + 1) \times (C - c + 1)$ array A s.t.

$$A_{i,j,k} = \sum_{i'=1}^{h} \sum_{j'=1}^{w} \sum_{k'=1}^{c} I_{i+i',j+j',k+k'} \; K_{i',j',k'}$$

Problem

- The pixels in the corners of an image have minor contributions in the output



Padding

- Making matrices larger by padding with zeros

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

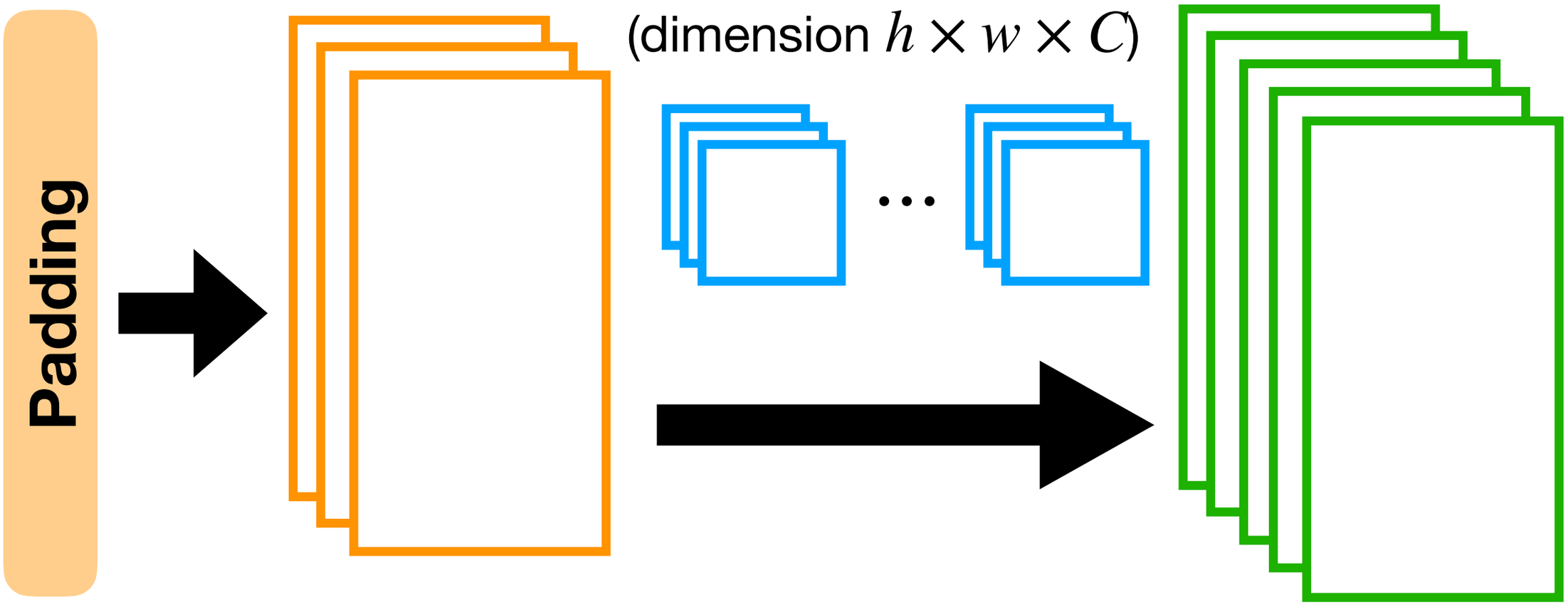
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & x_{11} & x_{12} & x_{13} & 0 \\ 0 & x_{21} & x_{22} & x_{23} & 0 \\ 0 & x_{31} & x_{32} & x_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Padding

- Making matrices larger by padding with zeros

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & x_{11} & x_{12} & x_{13} & 0 \\ 0 & x_{21} & x_{22} & x_{23} & 0 \\ 0 & x_{31} & x_{32} & x_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

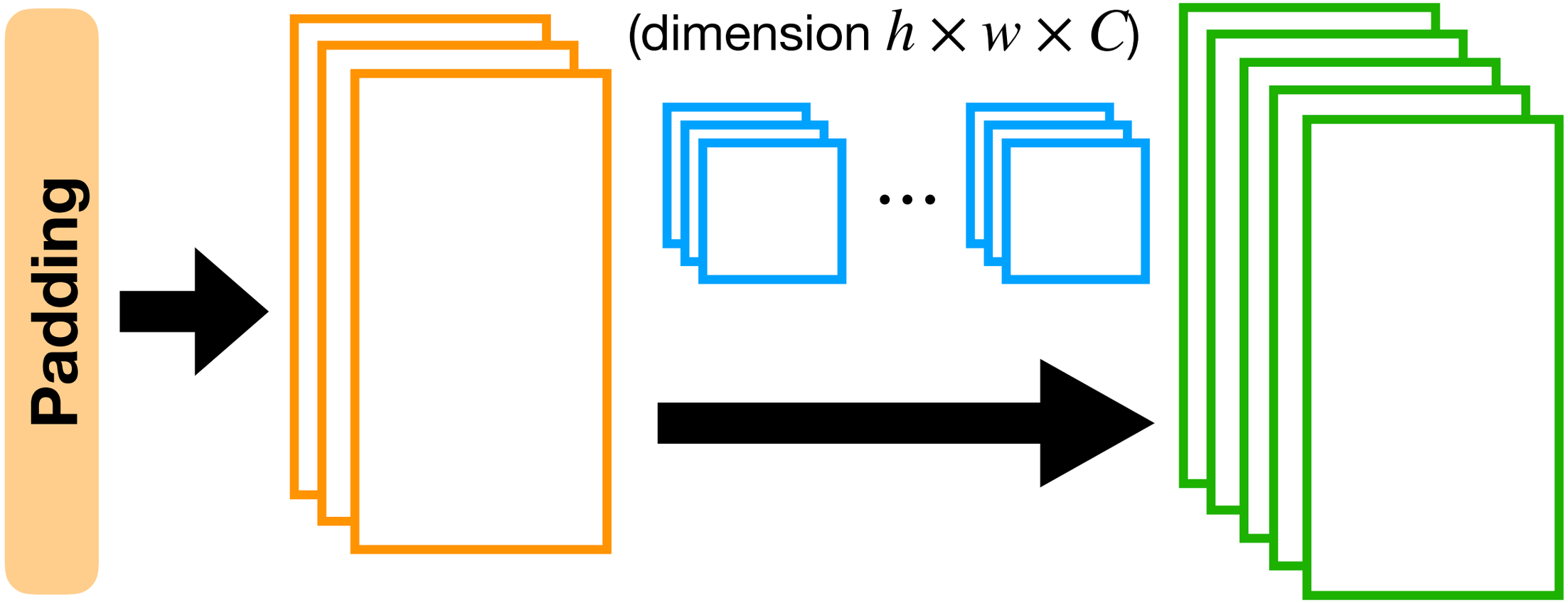
Convolutional layers



Convo

Entries of kernel arrays are
l*earnable weights* optimized by training

C' **kernel arrays**
(dimension $h \times w \times C$)



Input
$H \times W \times C$ array

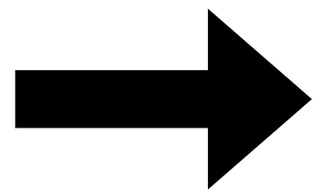
Output
$(H - h + 1) \times (W - w + 1) \times C'$
array

Pooling

- A technique to reduce the dimensions of features

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$

2×2 filter



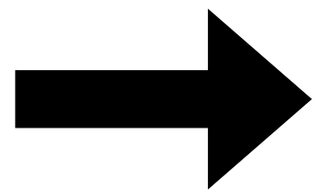
$$\begin{bmatrix} 5 & 3 \\ 6 & 7 \end{bmatrix}$$

Pooling

- A technique to reduce the dimensions of features

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$

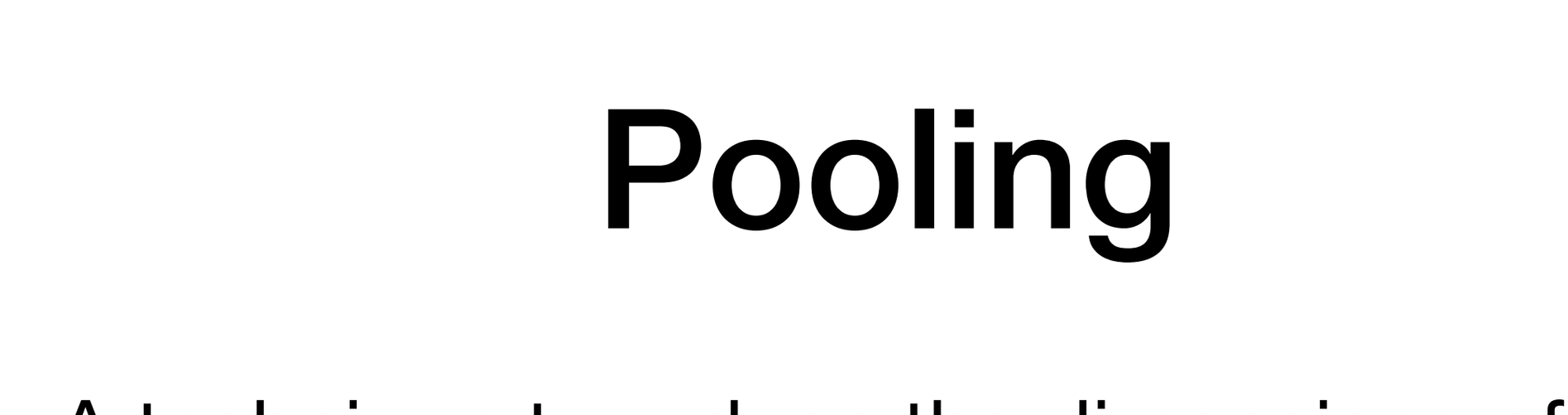
2×2 filter



$$\begin{bmatrix} 5 & 3 \\ 6 & 7 \end{bmatrix}$$

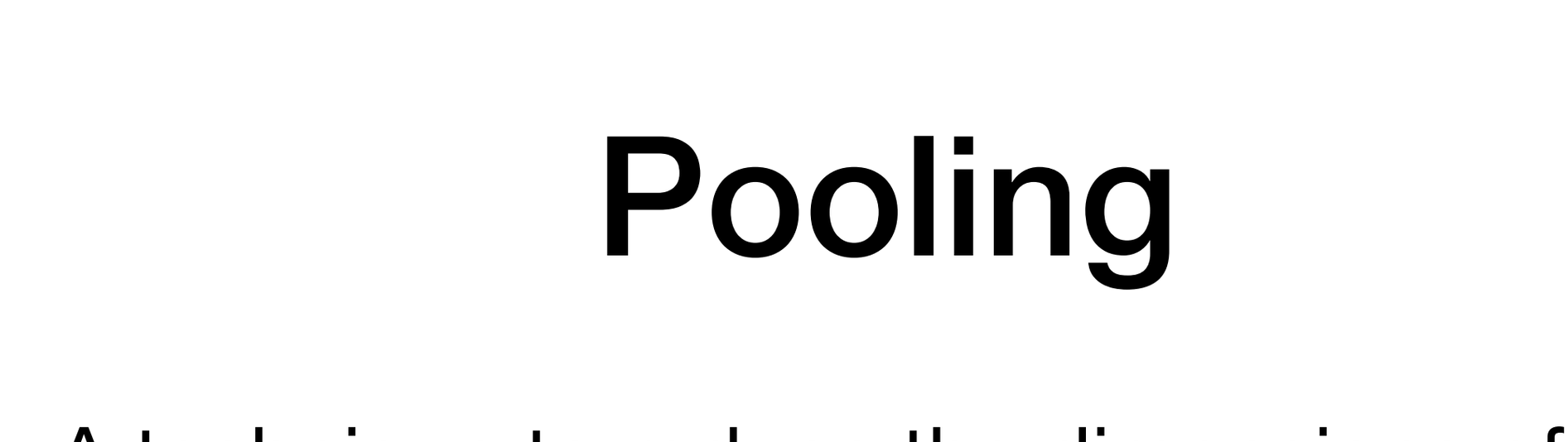
Pooling

- A technique to reduce the dimensions of features



Pooling

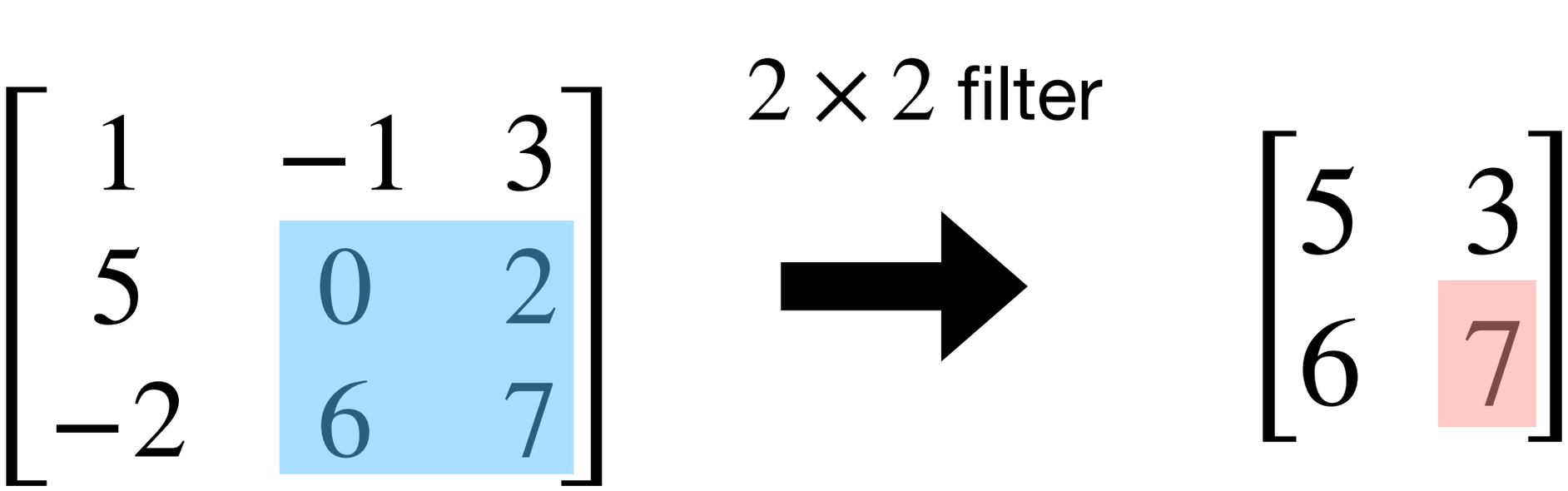
- A technique to reduce the dimensions of features



Pooling

- A technique to reduce the dimensions of features

Max pooling



Pooling

- A technique to reduce the dimensions of features

Average pooling

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix} \xrightarrow{2 \times 2 \text{ filter}} \begin{bmatrix} 1.25 & 1 \\ 2.25 & 3.75 \end{bmatrix}$$

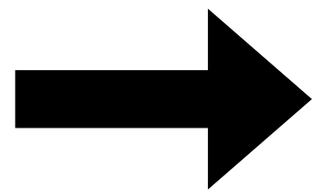
Pooling

- A technique to reduce the dimensions of features

Average pooling

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$

2×2 filter



$$\begin{bmatrix} 1.25 & 1 \\ 2.25 & 3.75 \end{bmatrix}$$

Pooling

- A technique to reduce the dimensions of features

Average pooling

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$

2×2 filter

$$\rightarrow \quad \begin{bmatrix} 1.25 & 1 \\ 2.25 & 3.75 \end{bmatrix}$$

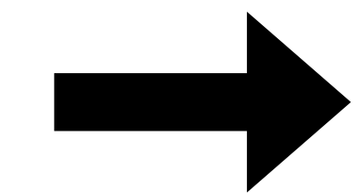
Pooling

- A technique to reduce the dimensions of features

Average pooling

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$

2×2 filter



$$\begin{bmatrix} 1.25 & 1 \\ 2.25 & 3.75 \end{bmatrix}$$

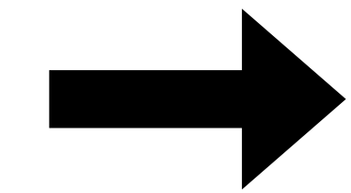
Pooling

- A technique to reduce the dimensions of features

Average pooling

$$\begin{bmatrix} 1 & -1 & 3 \\ 5 & 0 & 2 \\ -2 & 6 & 7 \end{bmatrix}$$

2×2 filter



$$\begin{bmatrix} 1.25 & 1 \\ 2.25 & 3.75 \end{bmatrix}$$

Why pooling?

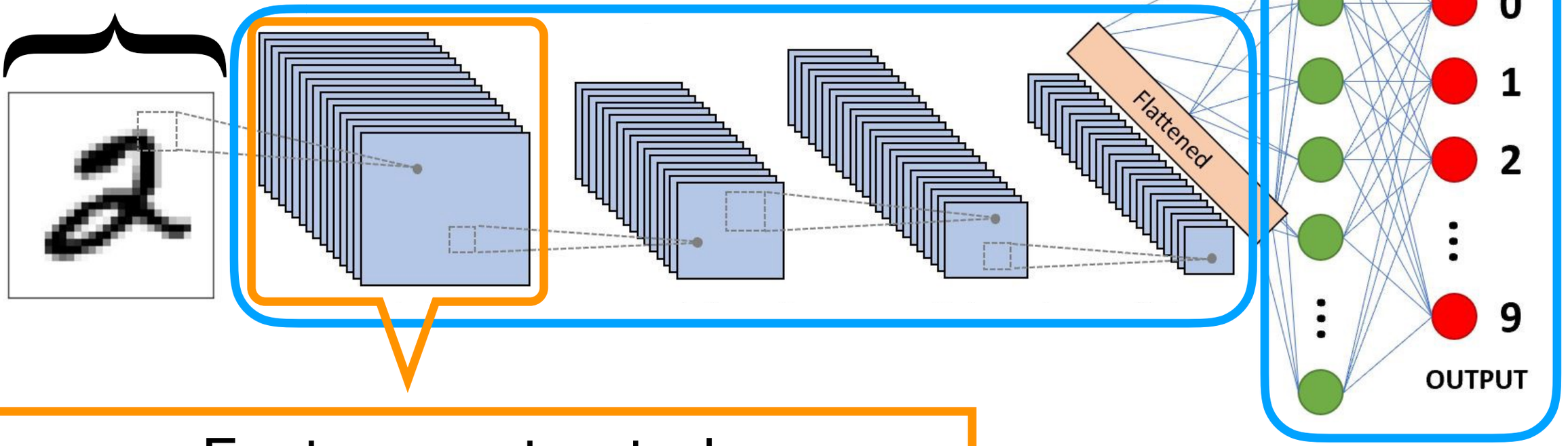
Pooling layers contribute to:

- Speed up
- Simplifying models
  - Simpler models are more unlikely to be overfitted

Convolution neural networks

Classification
by fully connected layers
+ extracted features

Input image
$\in \mathbb{R}^{H \times W \times C}$



Features extracted
by **a convolutional + pooling layer**

The image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Other topics

- Dropout
- Residual blocks
- Batch normalization

This lecture

Introducing two special types of NNs

- Convolutional neural networks (CNNs)
  - Specialized in image processing
  - Involving layers to extract features from images
- **Recurrent neural networks (RNNs)**
  - Specialized in sequential data
  - Relaxing the vanishing gradient problem

Task examples with sequential data

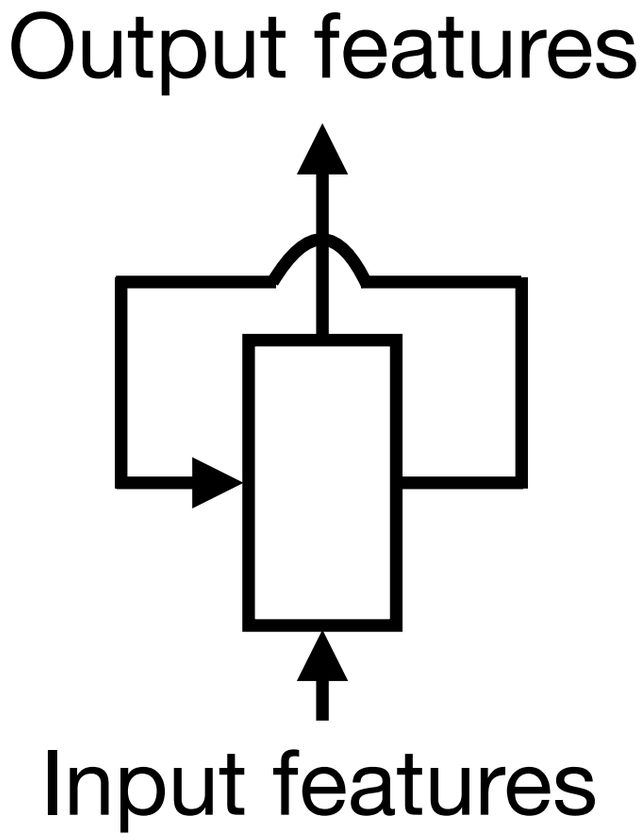
- Natural language processing
  - Sentences are sequences of words
- Audio signal processing
  - Audio waveform is a sequence of signals (in a digital format)
- Anomaly detection

Challenge

- Sequential data have variable lengths
- Feedforward NNs cannot be used because they suppose the size of data to be fixed

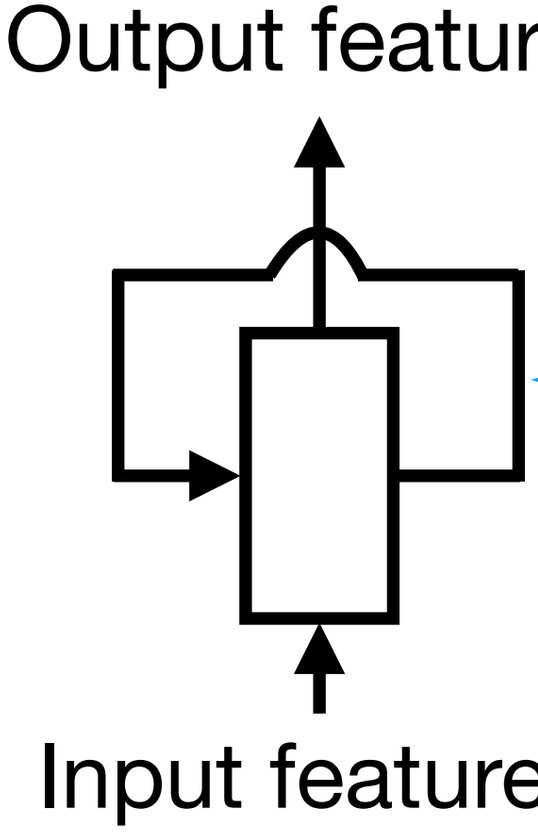
Recurrent neural networks

- ***Expandable*** according to input data, involving ***cyclic*** modules



Recurrent neural networks

- ***Expandable*** according to input data, involving ***cyclic*** modules

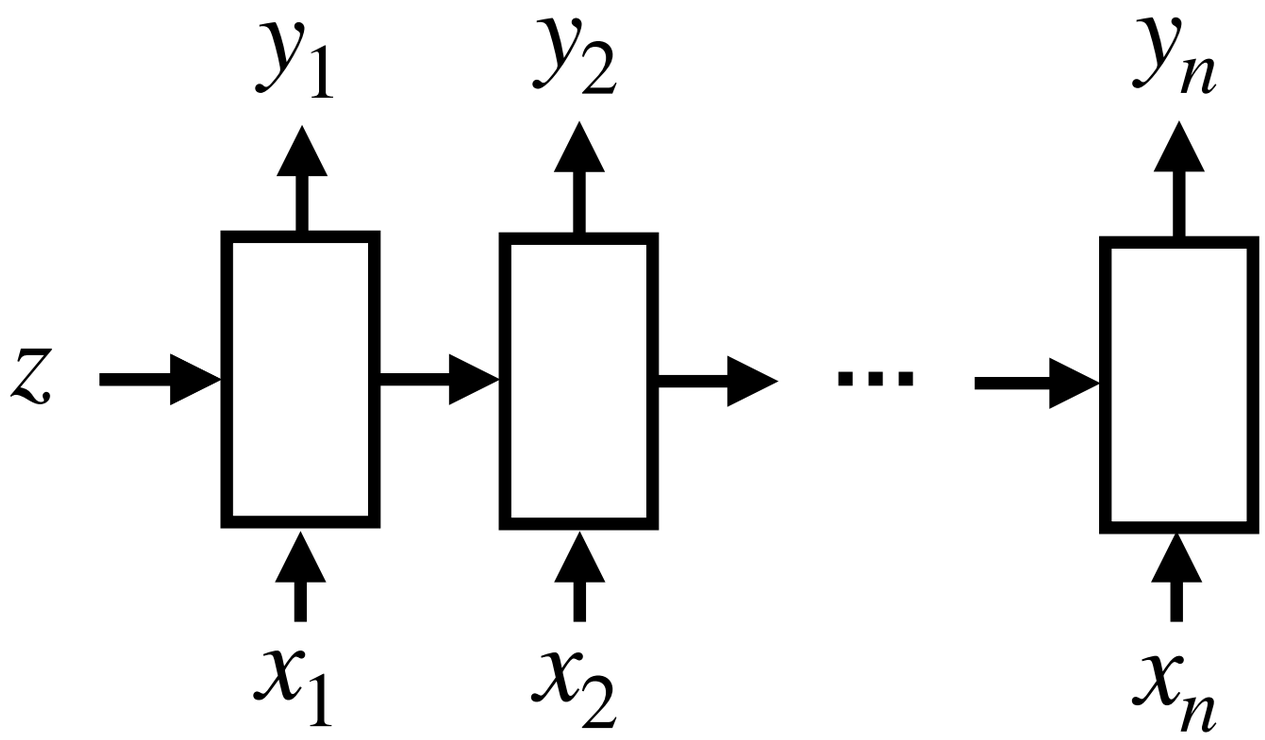


State features,
containing information propagated
from the current input to the next

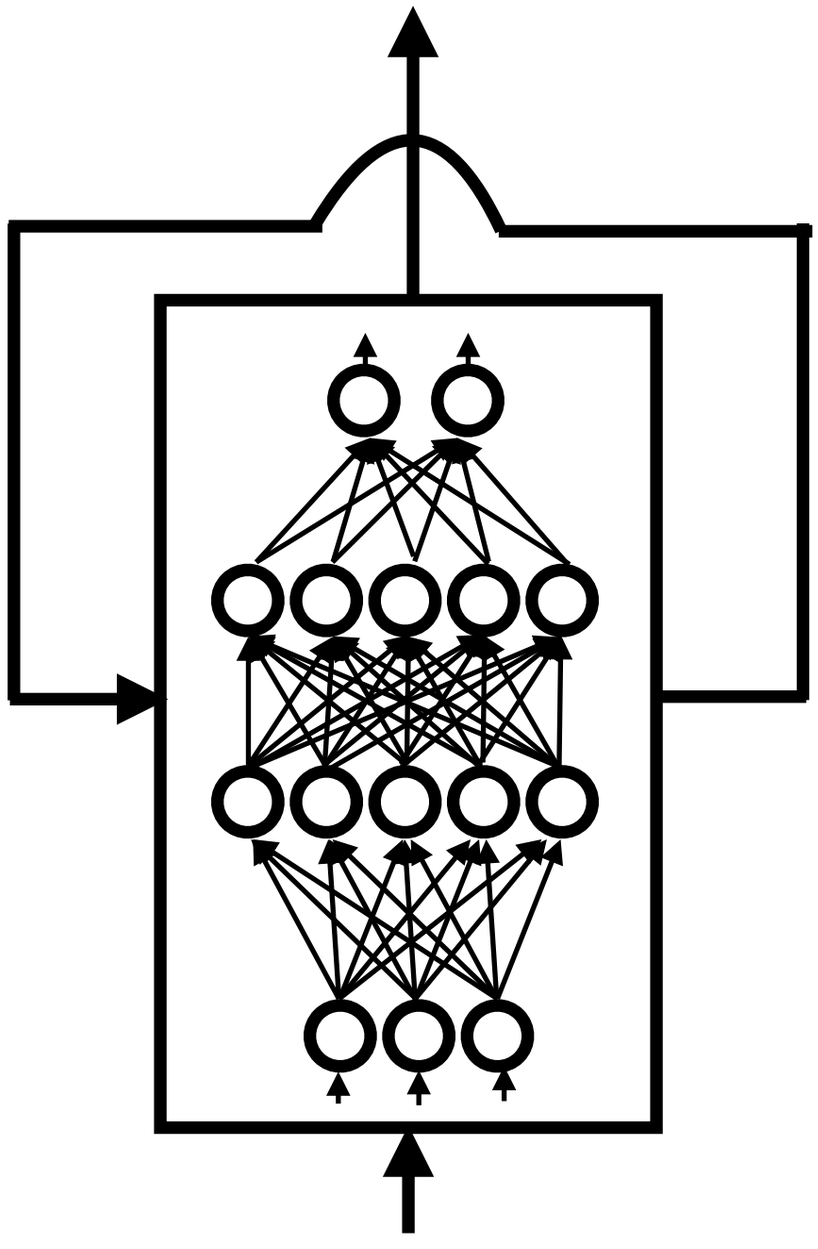
Recurrent neural networks

Given:

- Initial state feature $z \in \mathbb{R}^l$
- Input sequence $x_1, \cdots, x_n \in \mathbb{R}^m$

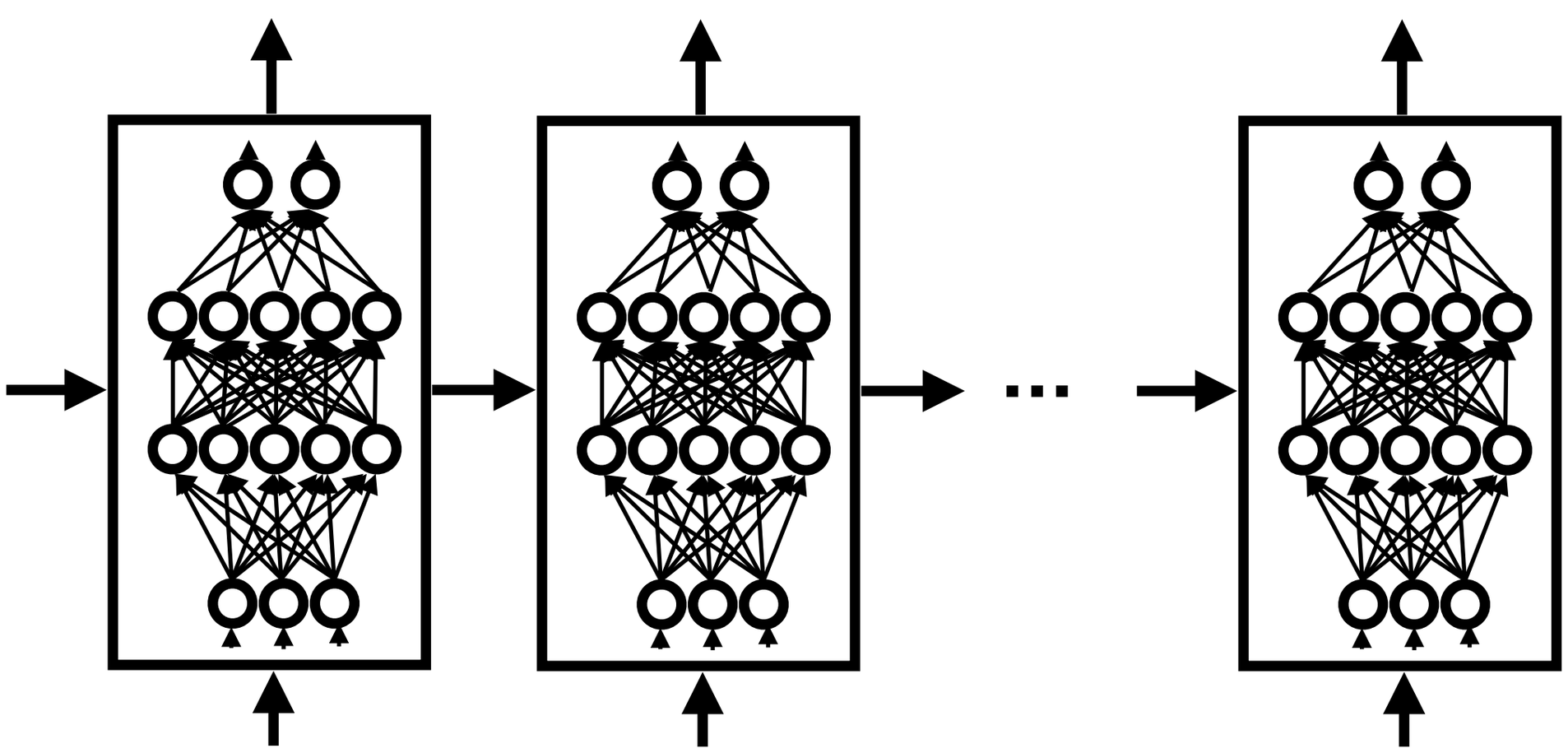


A simple implementation



Problem

- Expanded NNs are very deep, so they suffer from the vanishing gradient problem



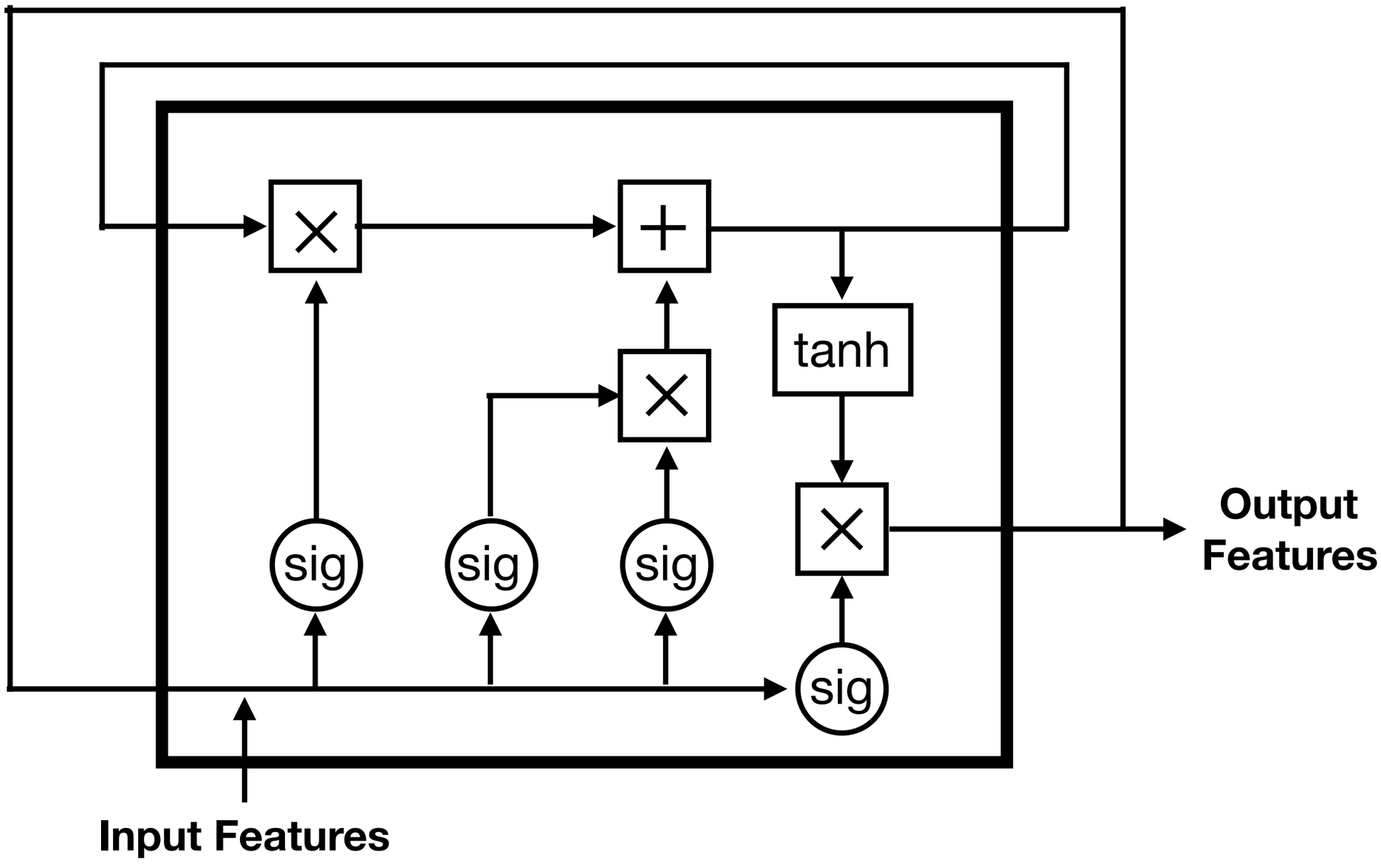
Long-short term memories (LSTMs)

f

Point-wise application of function f

f

Multiple neurons with activation f



Notice

- Next week is the turn of programming!
- Exercises will be assinged on Course N@vi