Deep Learning:

Convolutional Neural Networks

Robert Legenstein

Institute of Machine Learning and Neural Computation

Robert.legenstein@tugraz.at

Deep Learning VO - WS 25/26

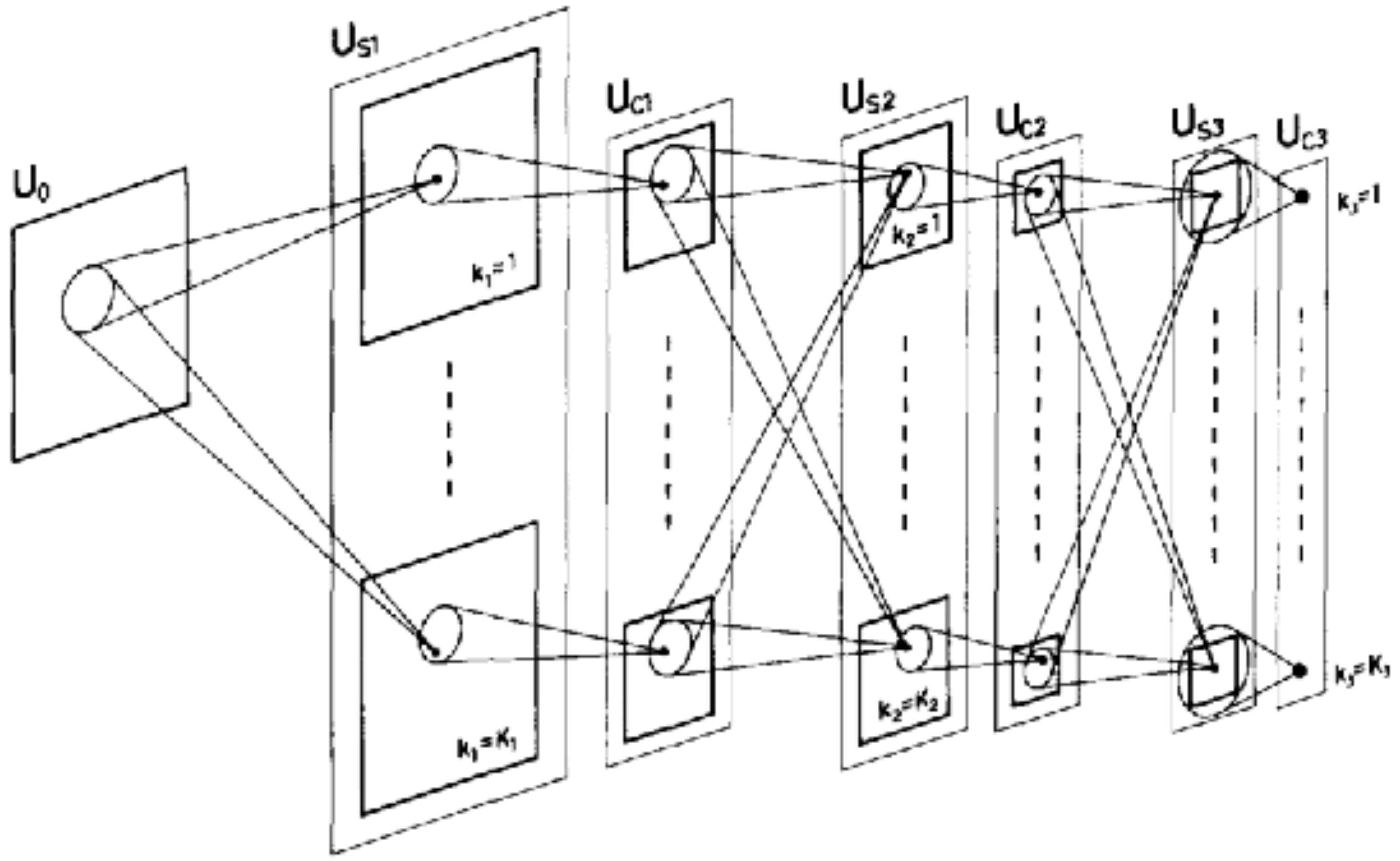
Lecture 7

Today

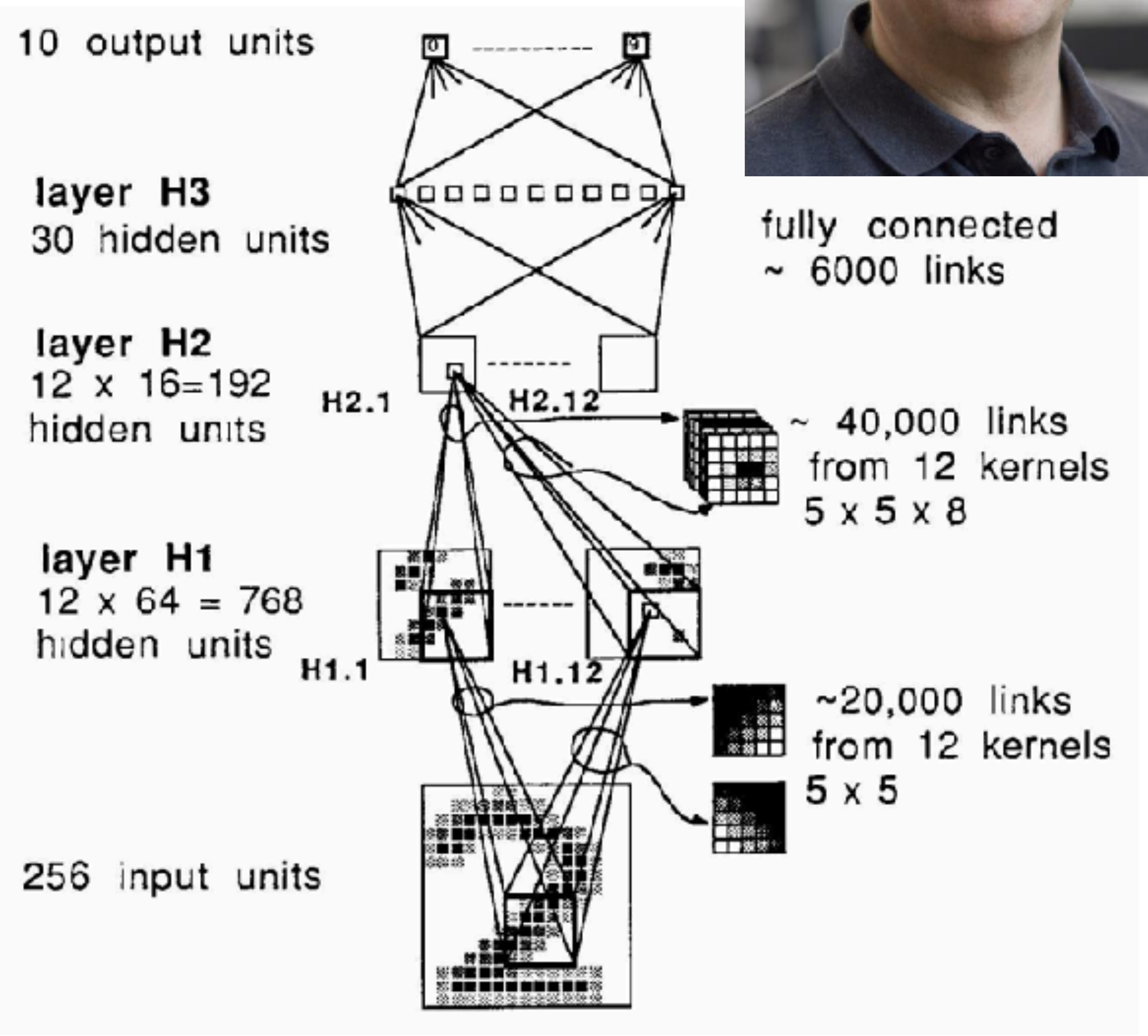
- Convolutional Neural Networks
  - Background
  - CNN Architecture
  - Going Deeper with CNNs
  - Residual Networks (ResNets) & Extensions

Convolutional Neural Networks

- Convolutional neural networks (CNNs) were introduced already in 1989 by Yann LeCun.
- Prior work by Kunihiko Fukushima: "Neocognitron"
- Inspired by the architecture of the visual system of vertebrates.



[K. Fukushima (1980): "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybernetics*.]



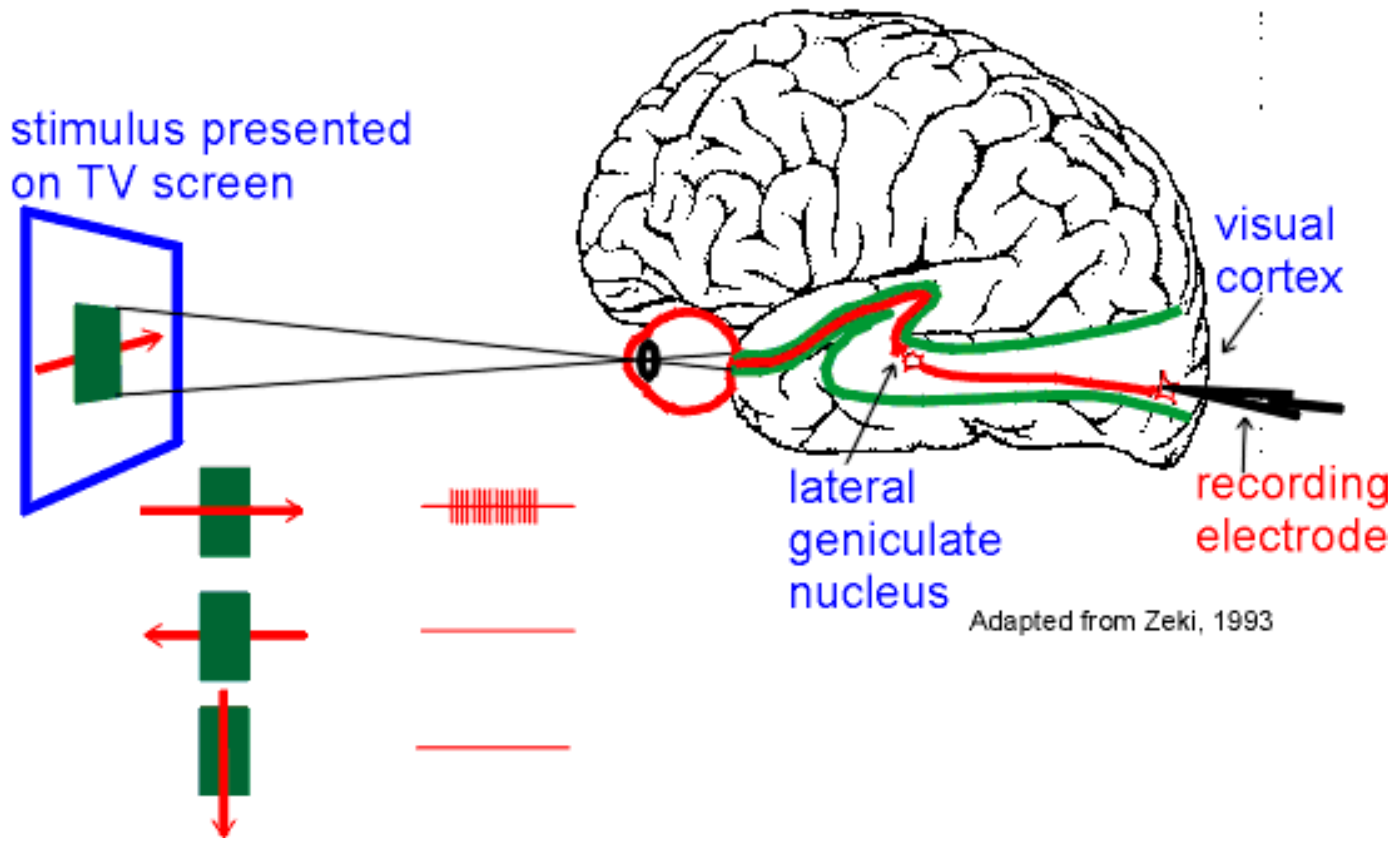
[Y. LeCun, et al. (1989): "Backpropagation applied to handwritten zip code recognition", *Neural Computation*.]

[Y. LeCun (1989): "Generalization and network design strategies", TechRep CRG-TR-89-4 Univ. of Toronto.]

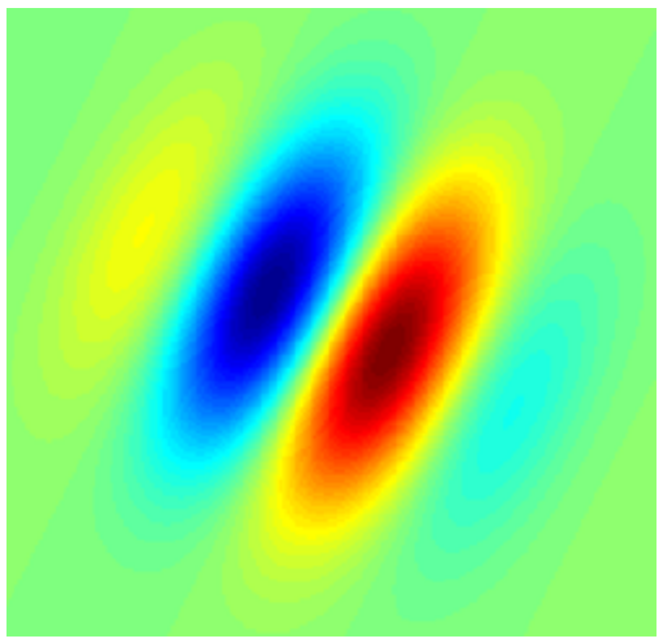
Background on Vision

Hubel and Wiesel (1950s & 60s):

- Neurons in cat and monkey visual cortexes respond to small regions of the visual fields.
- Neighboring cells have similar and overlapping receptive fields.



- Simple cells: Output is maximized by straight edges having particular orientations.
- Complex cells: Output is insensitive to the exact position of the edges in the field.



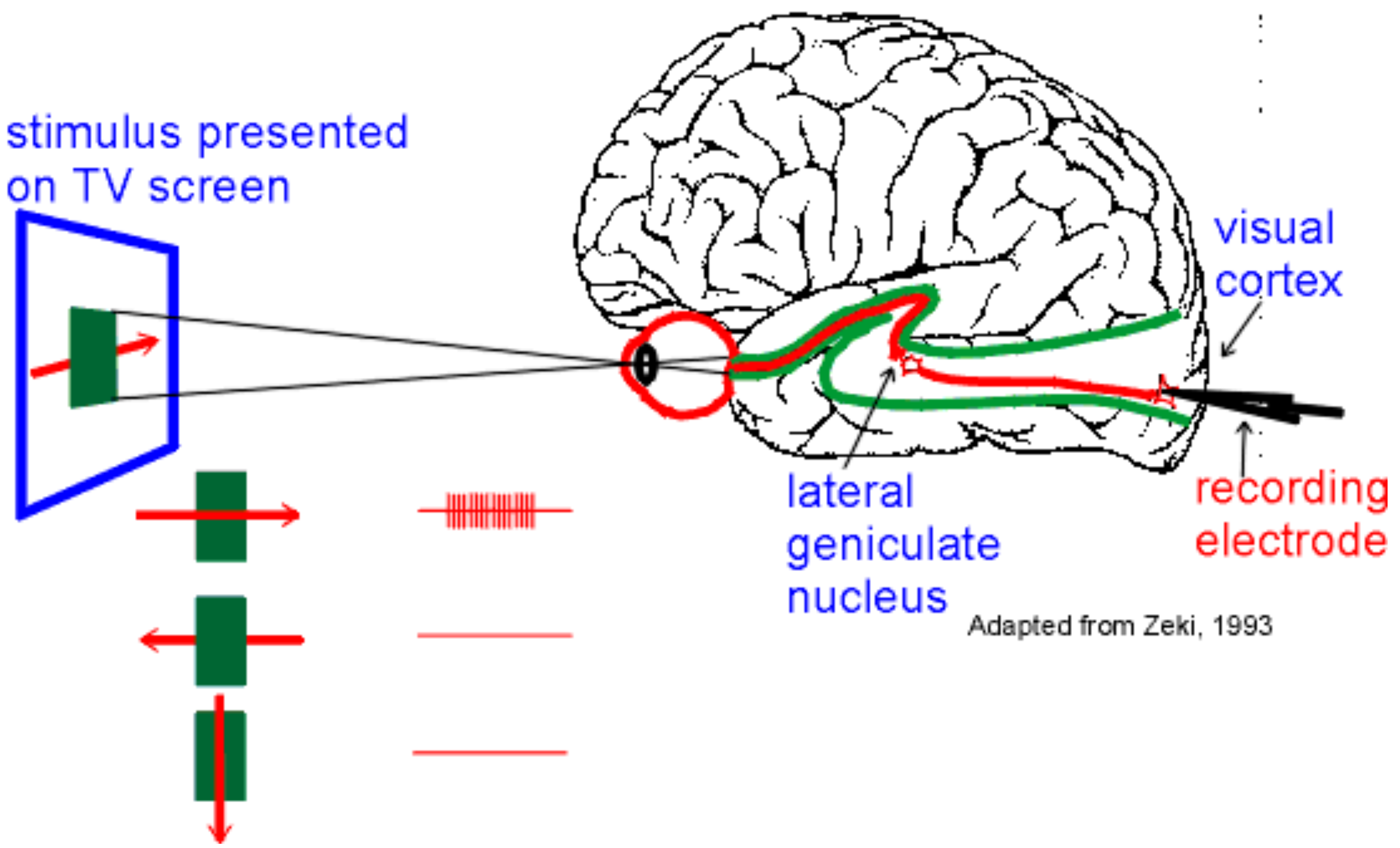
a Gabor filter

[David H. Hubel and Torsten N. Wiesel (2004): "Brain and visual perception: the story of a 25-year collaboration"]

Background on Vision

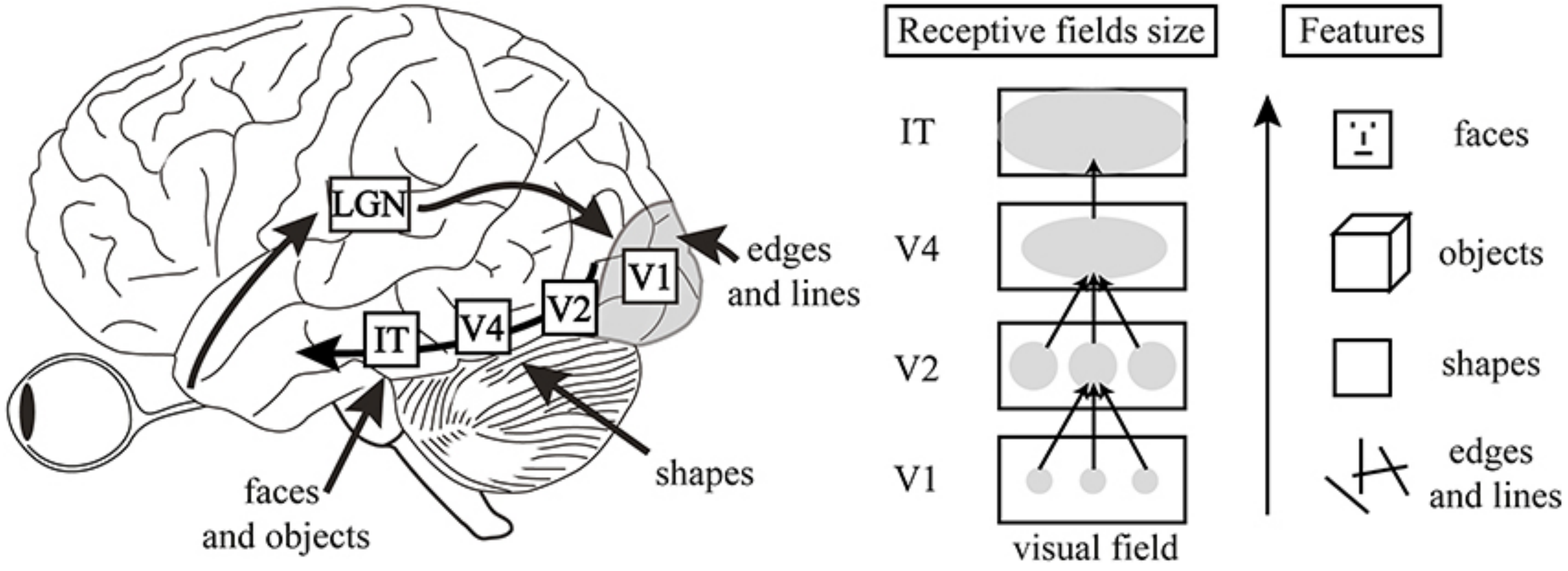
Hubel and Wiesel (1950s & 60s):

- Neurons in cat and monkey visual cortexes respond to small regions of the visual fields.
- Neighboring cells have similar and overlapping receptive fields.



The visual system:

- Hierarchical organization of areas with increasing receptive field sizes and increasingly complex features.



[Michael H. Herzog and Aaron M. Clarke. (2014): "Why vision is not both hierarchical and feedforward"]

Convolutional Networks

Induced invariances by network structure (Inductive Bias).

Convolutional neural networks (CNNs) are:

- Feed-forward neural networks.
- Typically applied on inputs with local structure (images, video, audio).
- Trained end-to-end with stochastic gradient descent (or a variants like Adam).

Statistical insights for image classification problems:

- Images have local features.
    - Close pixels are more strongly correlated than distant ones.
- Local features can appear at different locations.
    - e.g., edges, or objects at different positions.
- Output should be invariant to local distortions or position shifts.

Today

- [ ] Convolutional Neural Networks
  - [x] Background
  - [ ] CNN Architecture
  - [ ] Going Deeper with CNNs
  - [ ] Residual Networks (ResNets) & Extensions

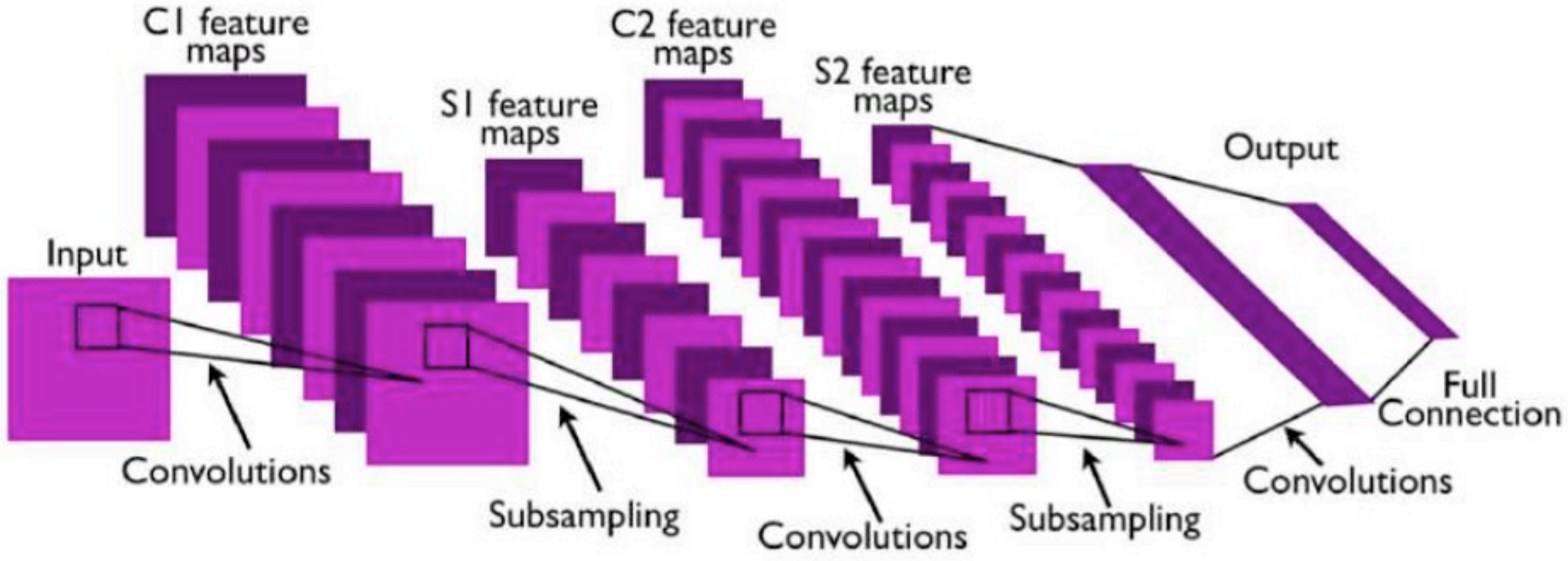
CNN Architecture

(1) Convolutional feature maps:

- Extract features at all possible locations.

(2) Pooling (subsampling):

- Invariance to local distortions.
- Reduce feature map size.



LeNet-5 Architecture (1995)

[LeCun et al. (1998): "Gradient-based learning applied to document recognition”]

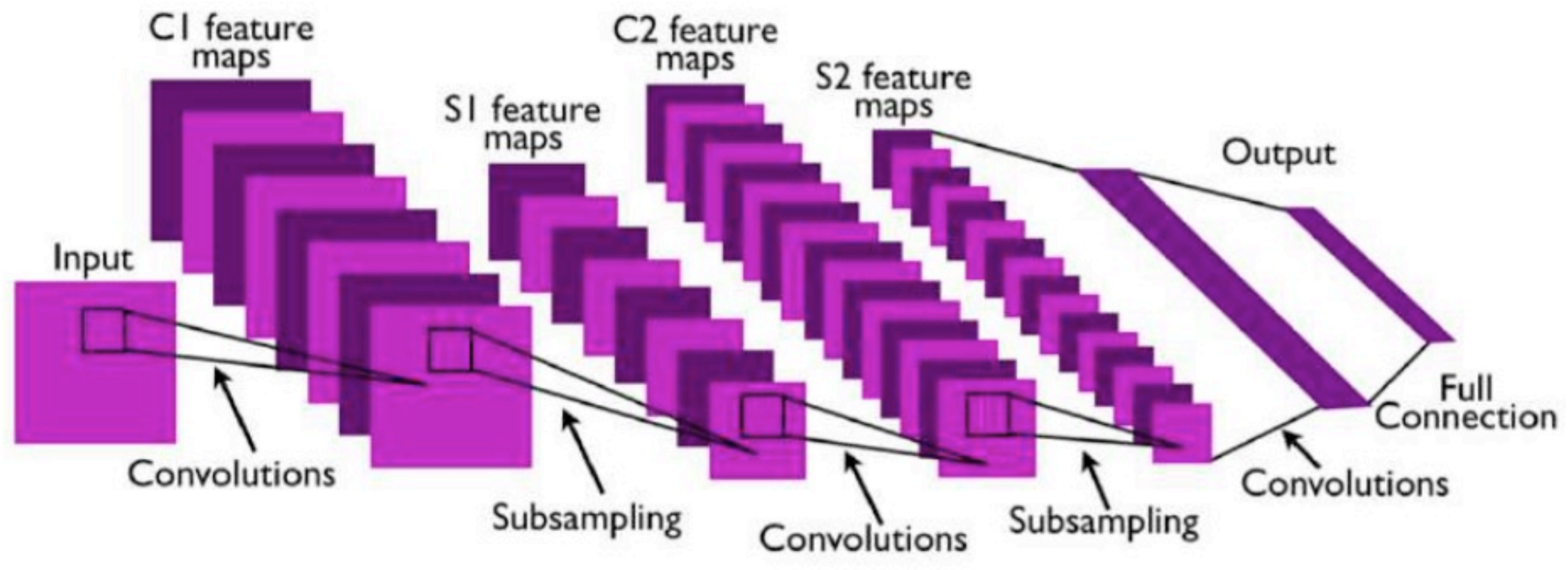
CNN Architecture

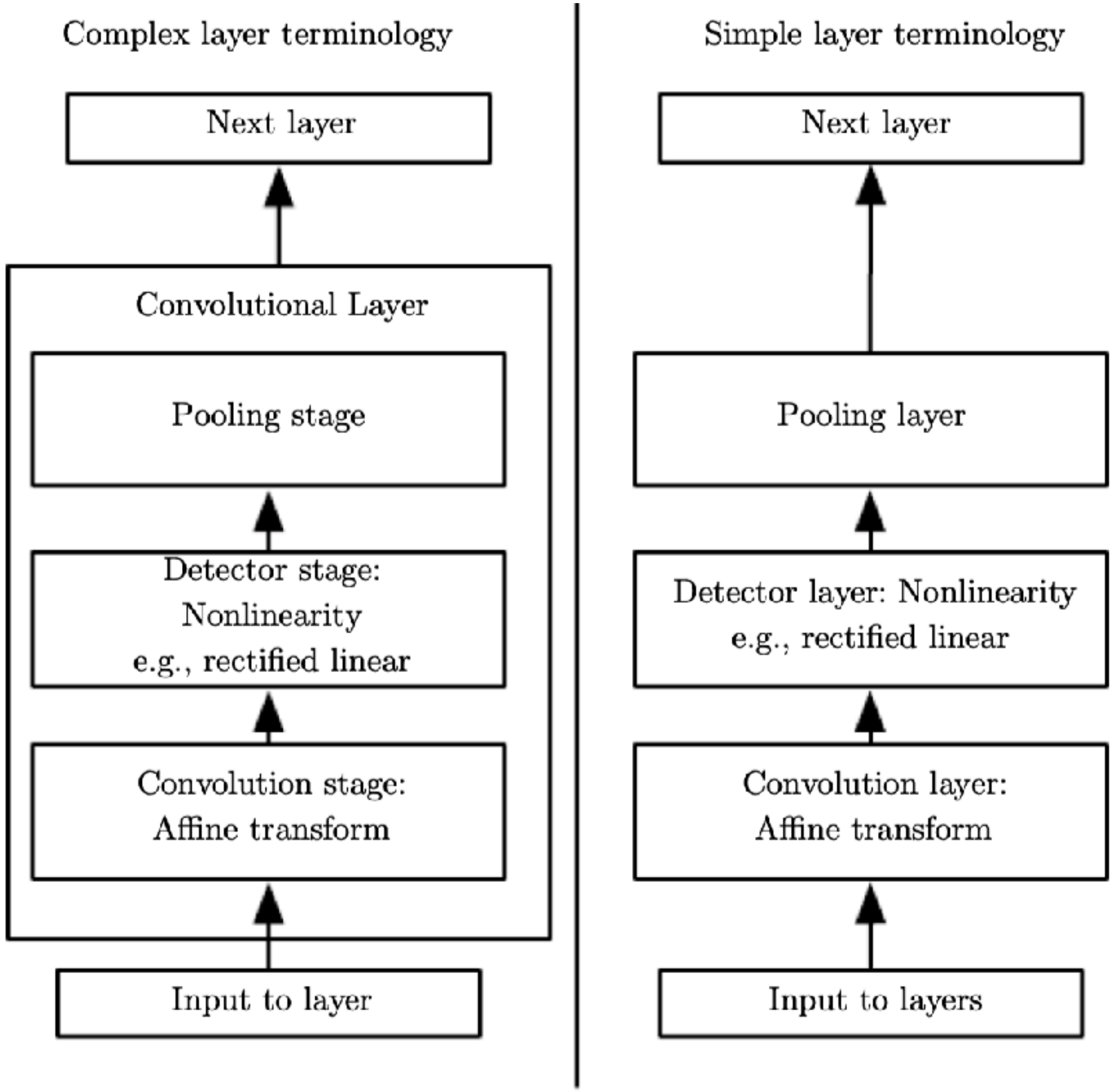
(1) Convolutional feature maps:

- Extract features at all possible locations.

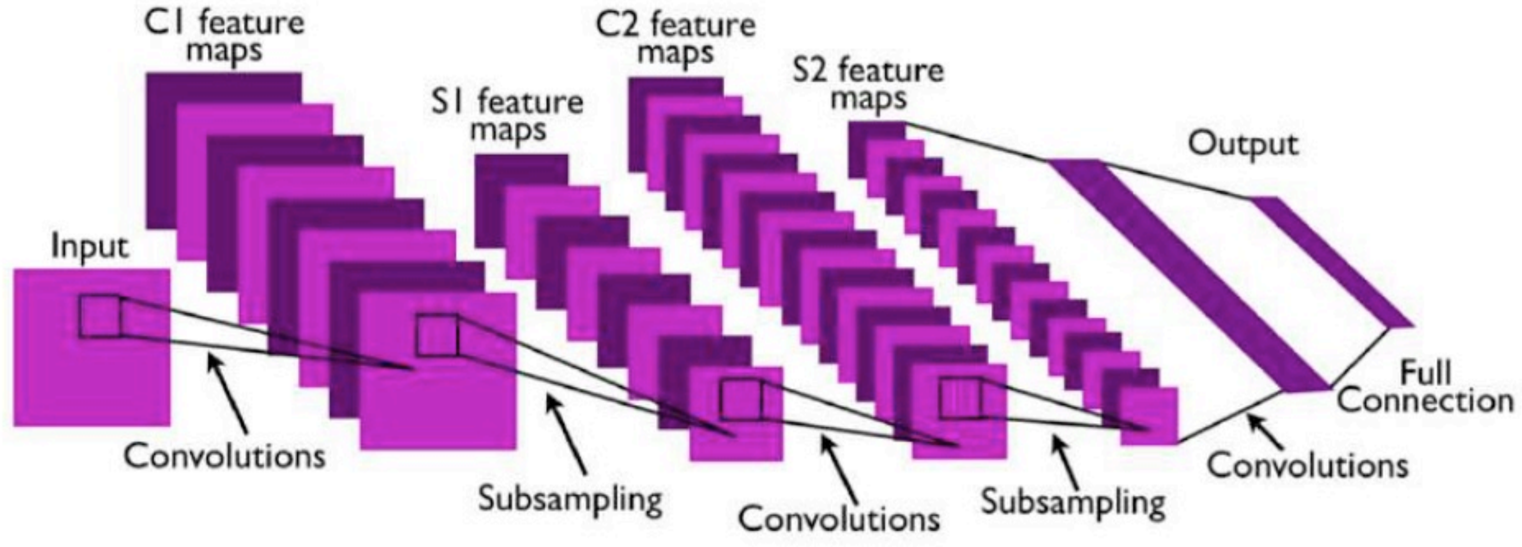
(2) Pooling (subsampling):

- Invariance to local distortions.
- Reduce feature map size.

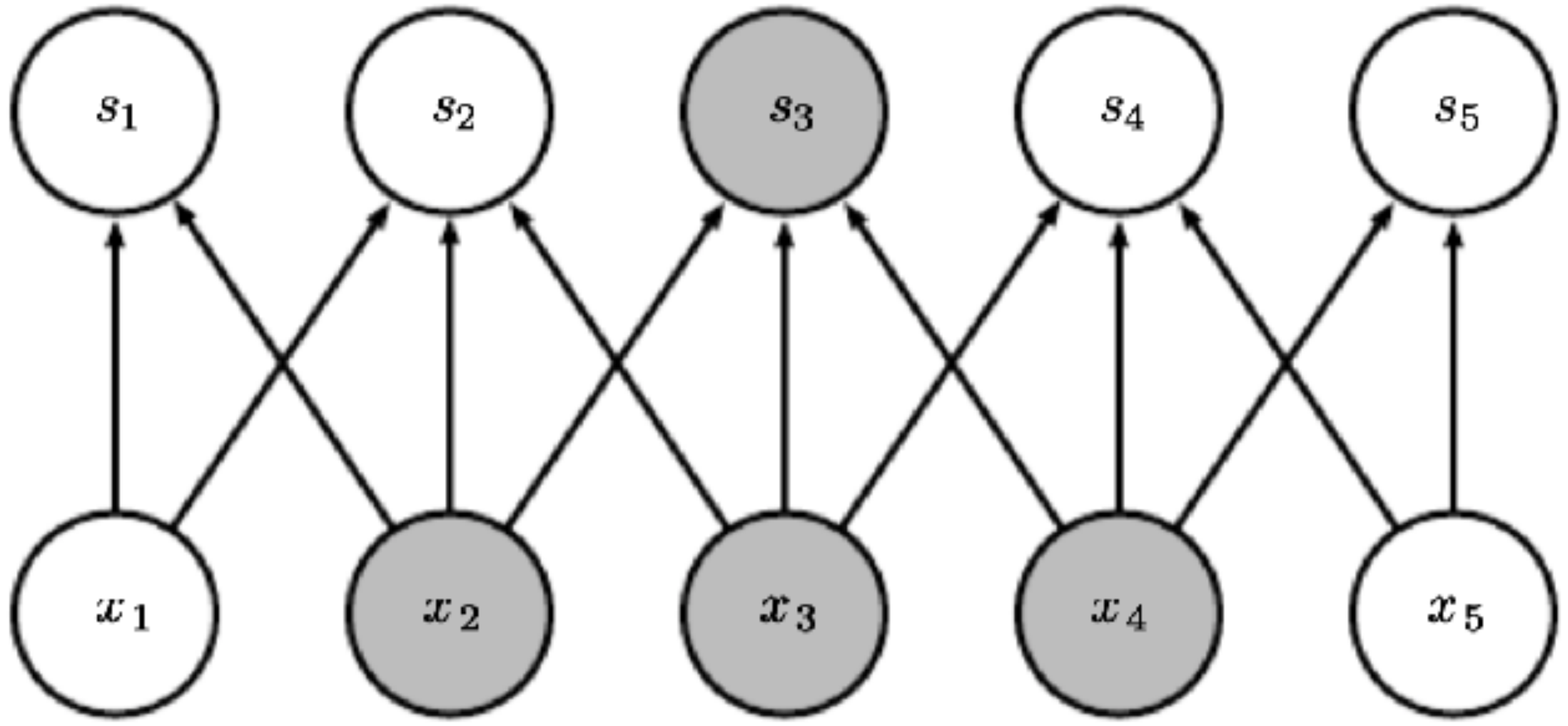




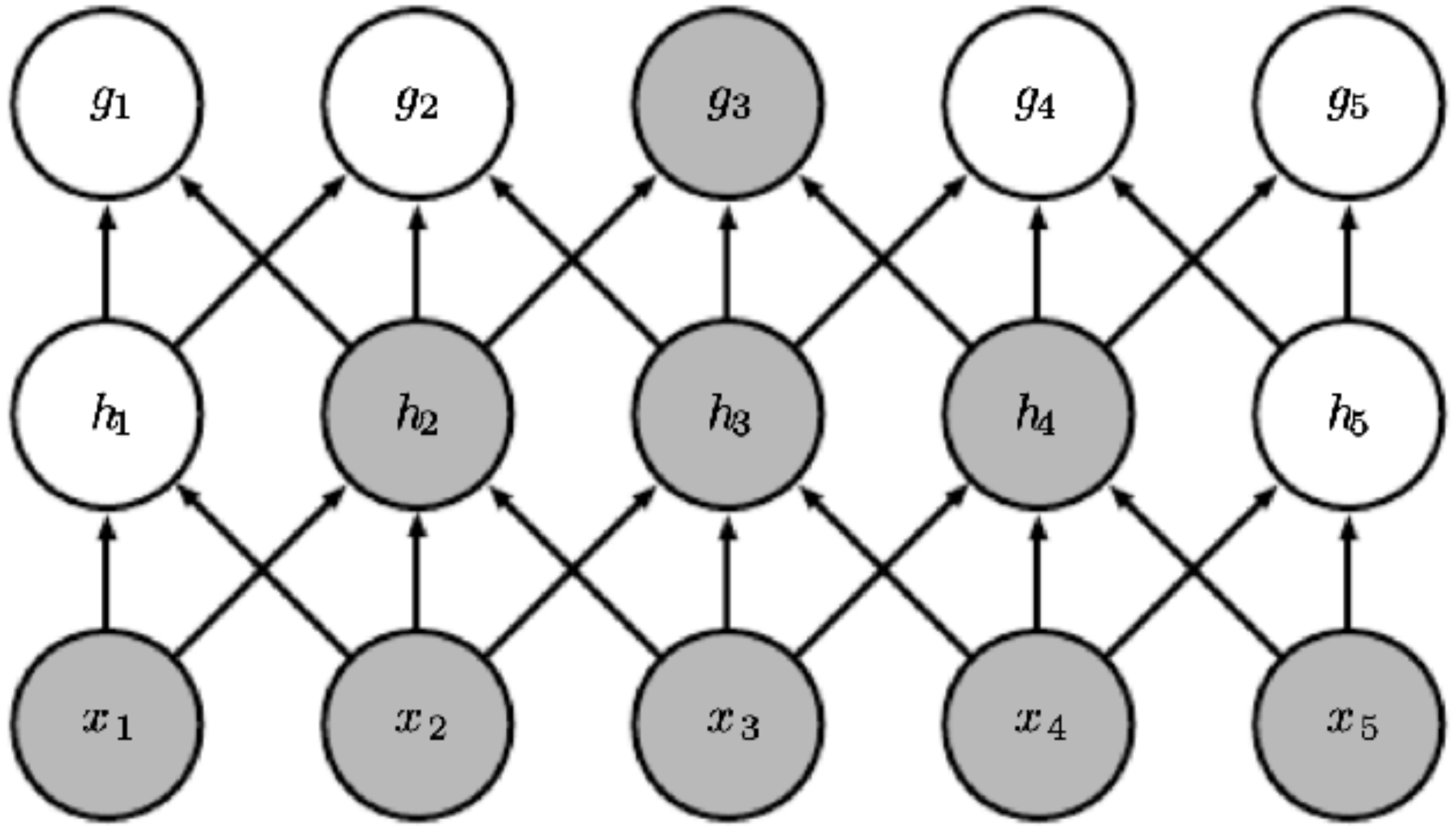
Local Connectivity / Convolution



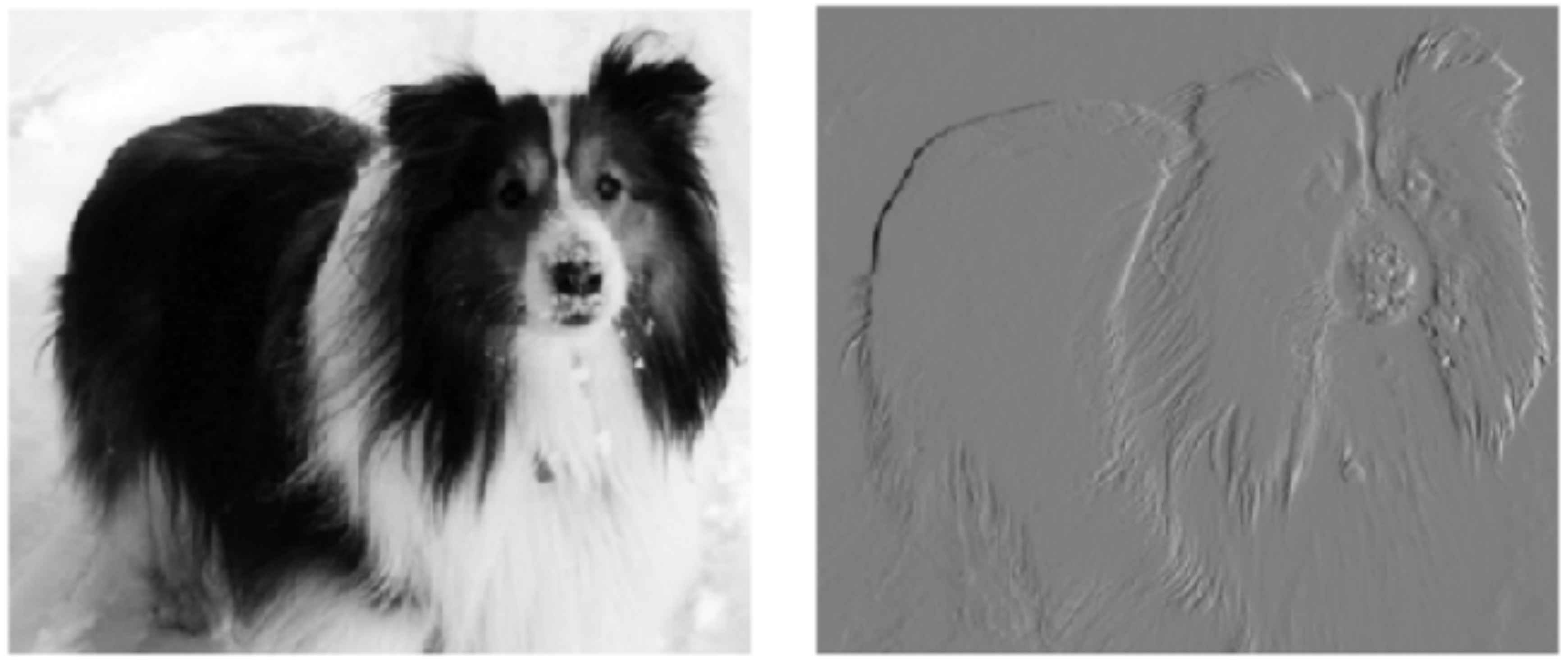
Each 1st layer neuron gets input from a local patch in the input image:



Higher layer neurons have larger "receptive fields":

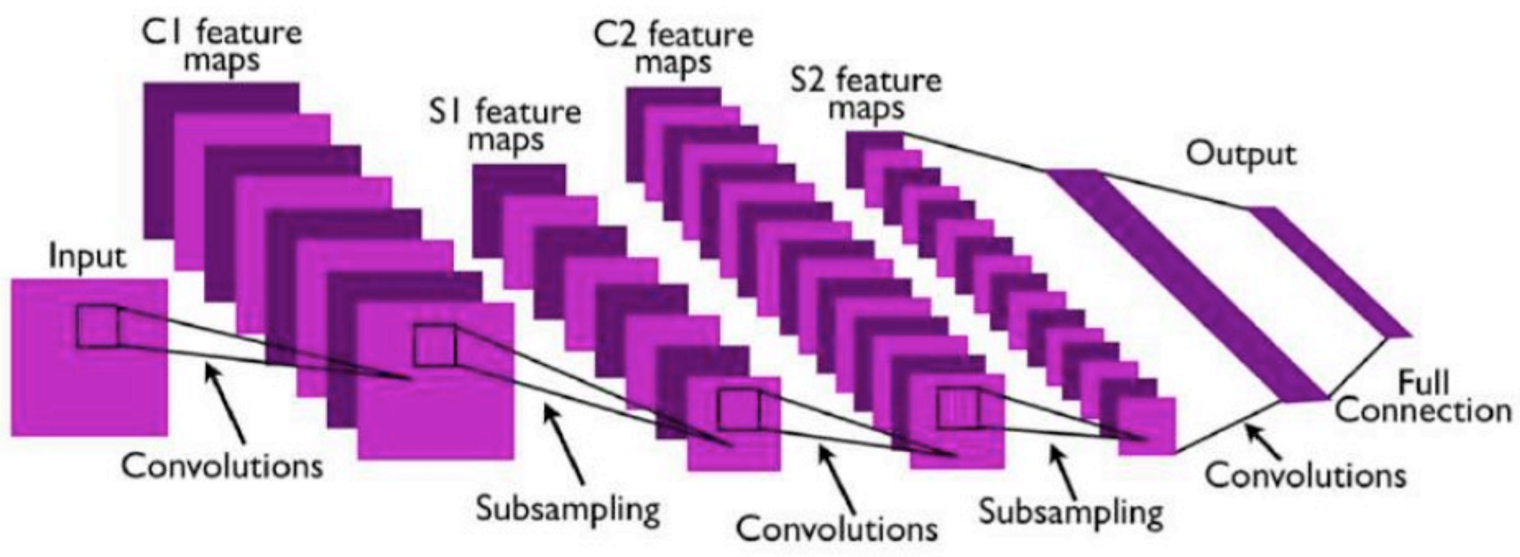


Weight sharing: All neurons in a feature map share the same weight values.

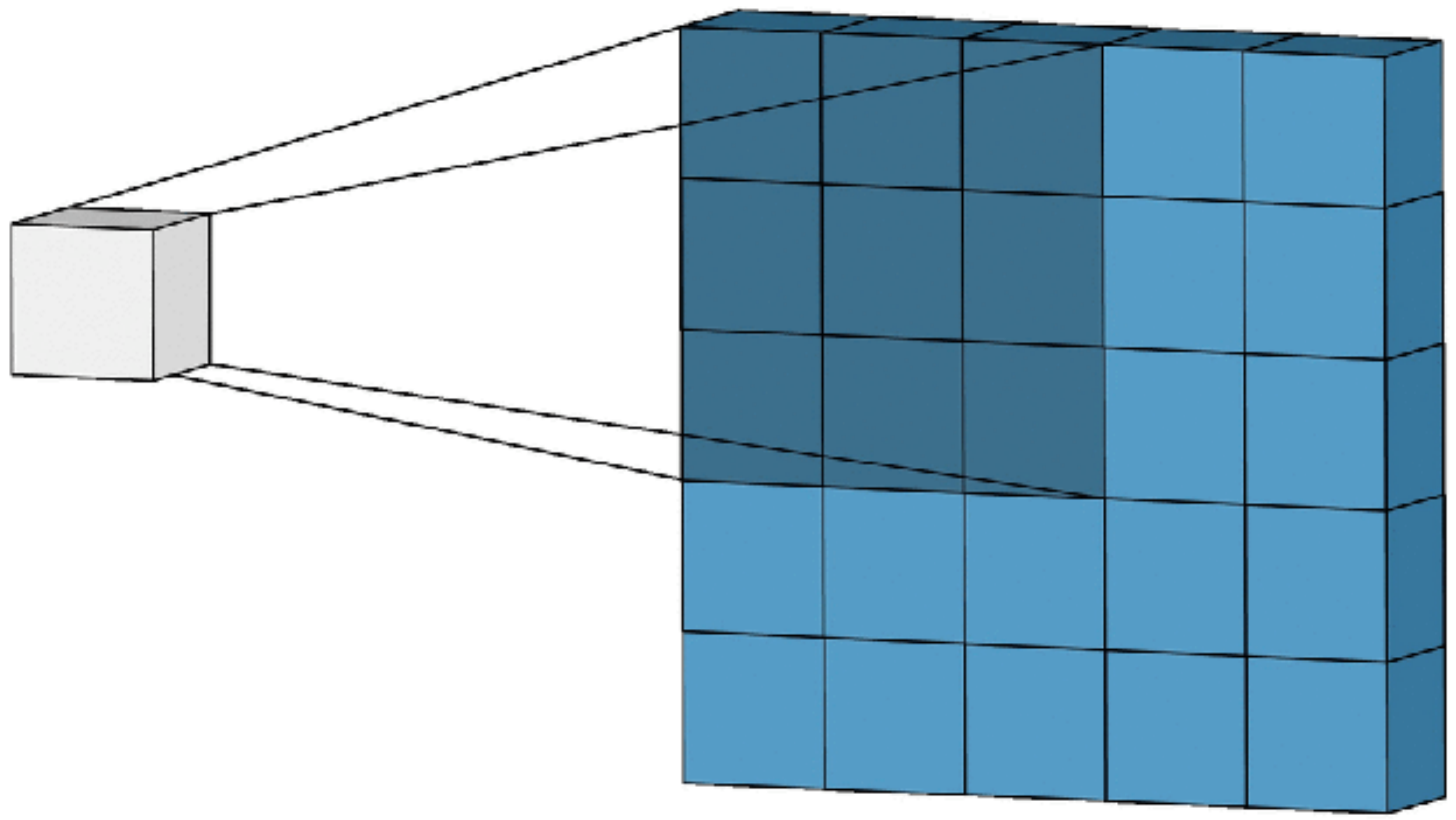


Example: Edge detection feature

Local Connectivity / Convolution



Weight sharing: All neurons in a feature map share the same weight values.



This is equivalent to a 2D convolution:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

$S(i,j)$: Output of convolution at position (i,j)

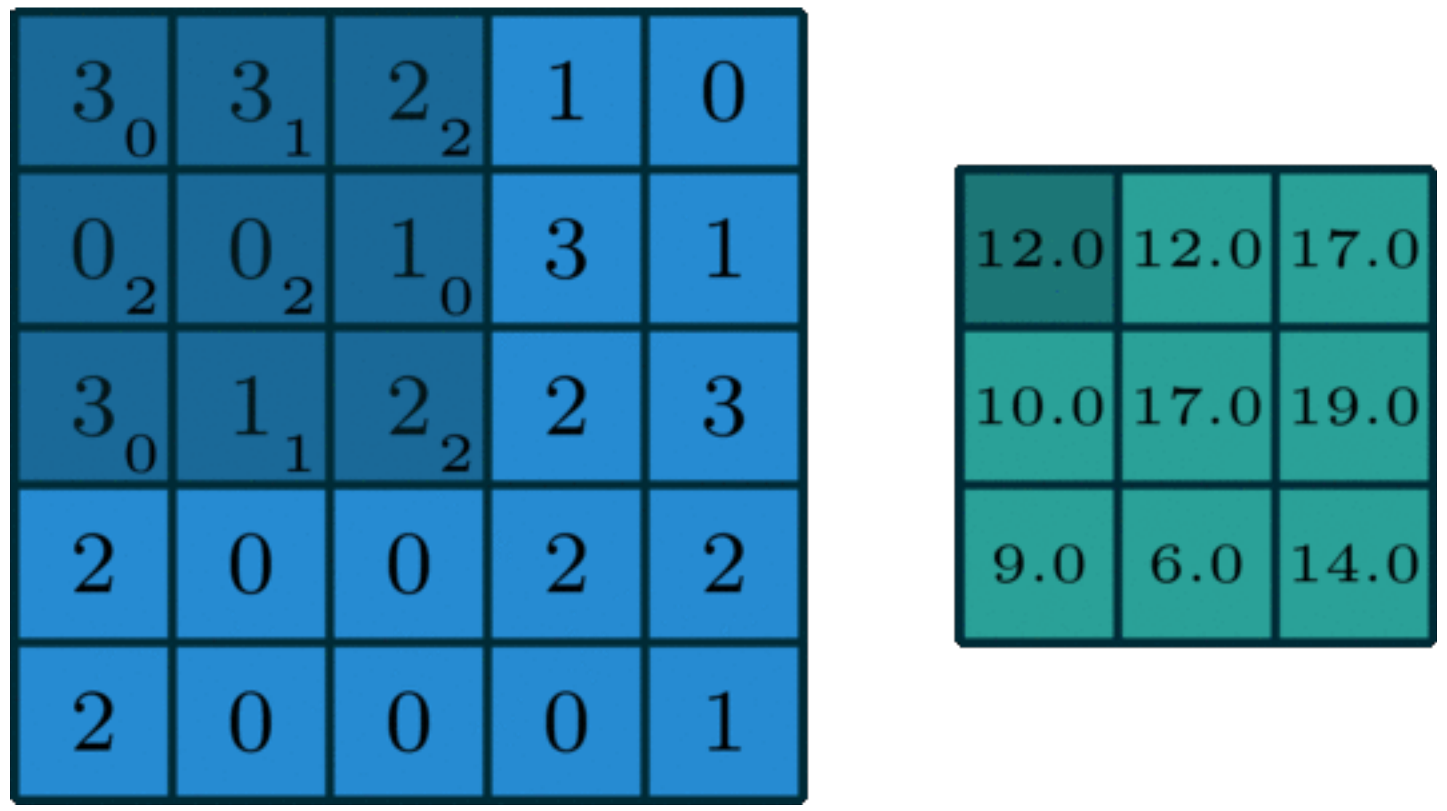
[activation of neuron in feature map]

$I(x,y)$: Pixel-value of image at position (x,y)

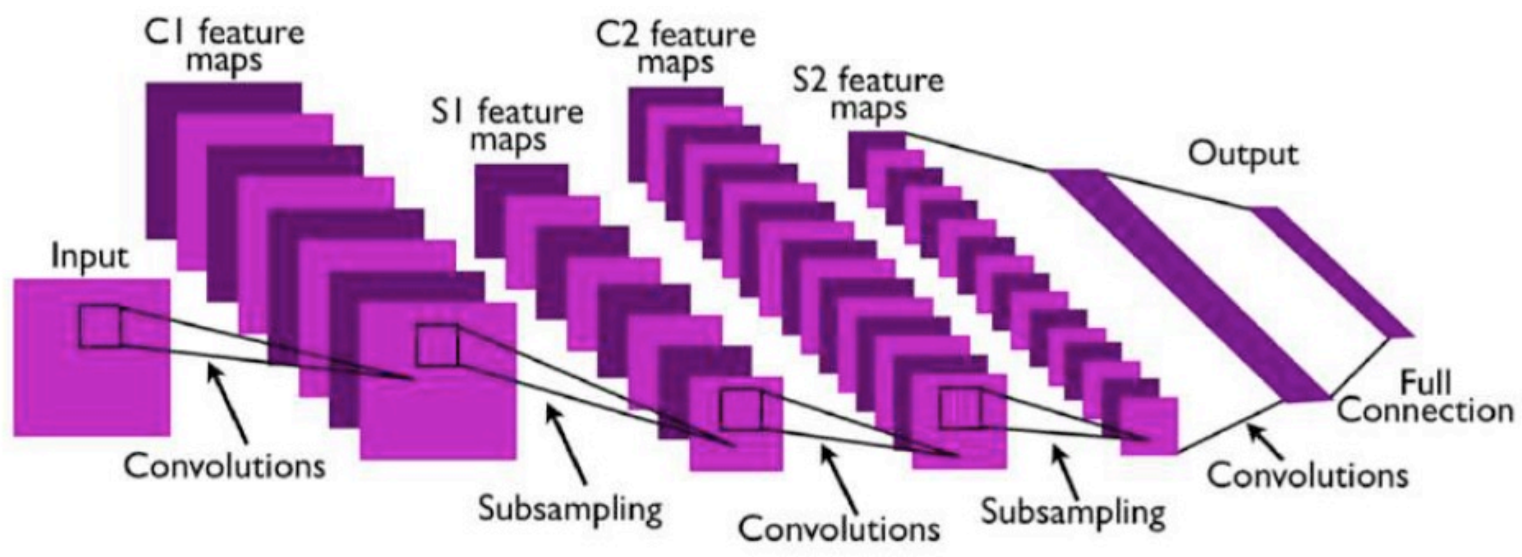
$K(m,n)$: Kernel-value at kernel-position (m,n) [K defined by weight vector]

Equivalently (by mirroring the kernel K):

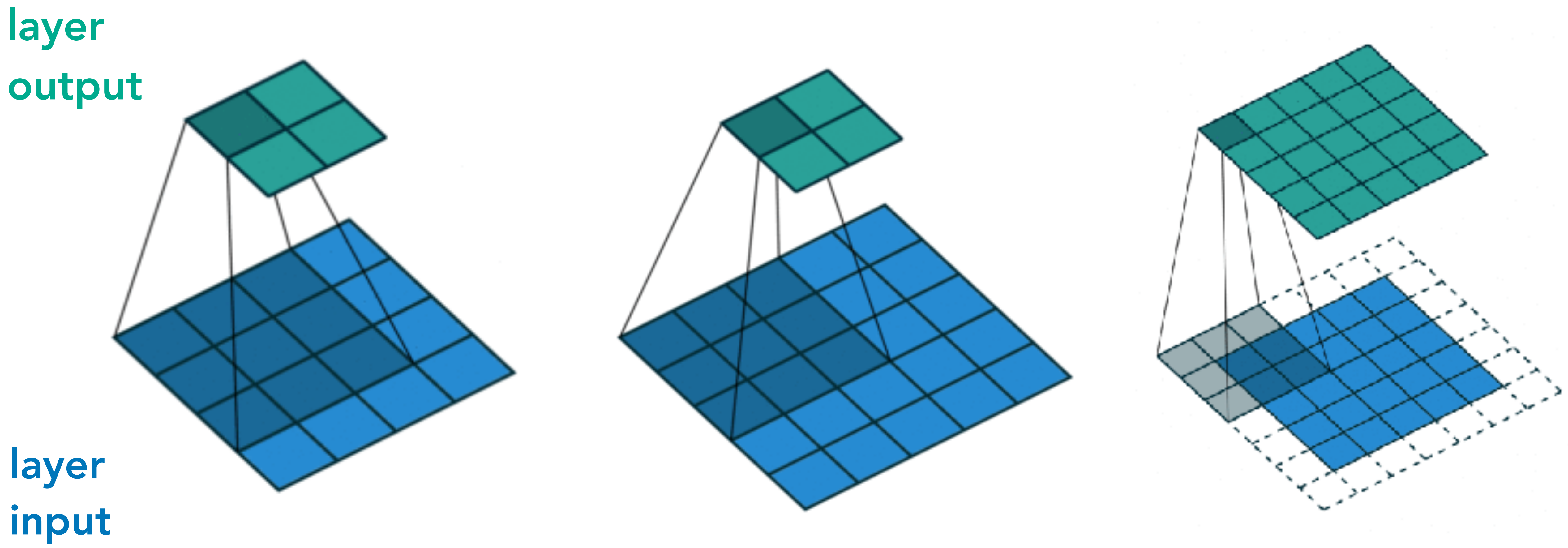
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$



Convolution Operation in 2D



Examples with a 3x3 convolution kernel:



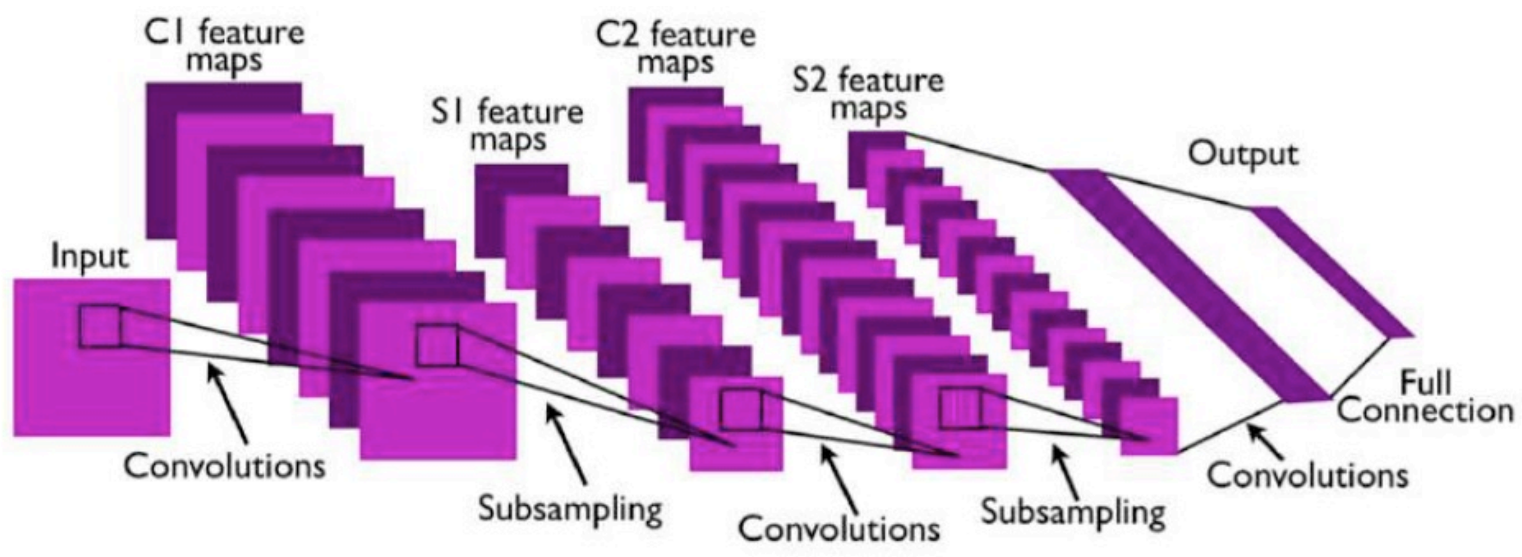
stride=(1,1)
no padding

stride=(2,2)
no padding

stride=(1,1)
with padding

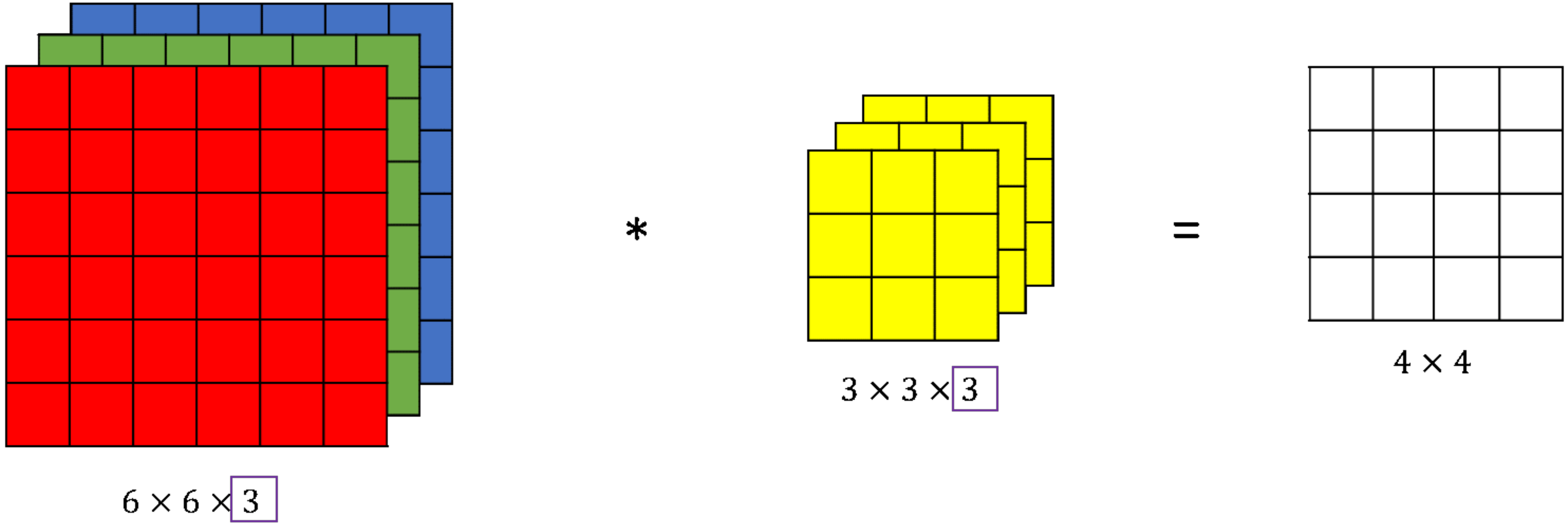
Animations from: Dumoulin & Visin (2018), "A guide to convolution arithmetic for deep learning".

Convolution Operation in 2D

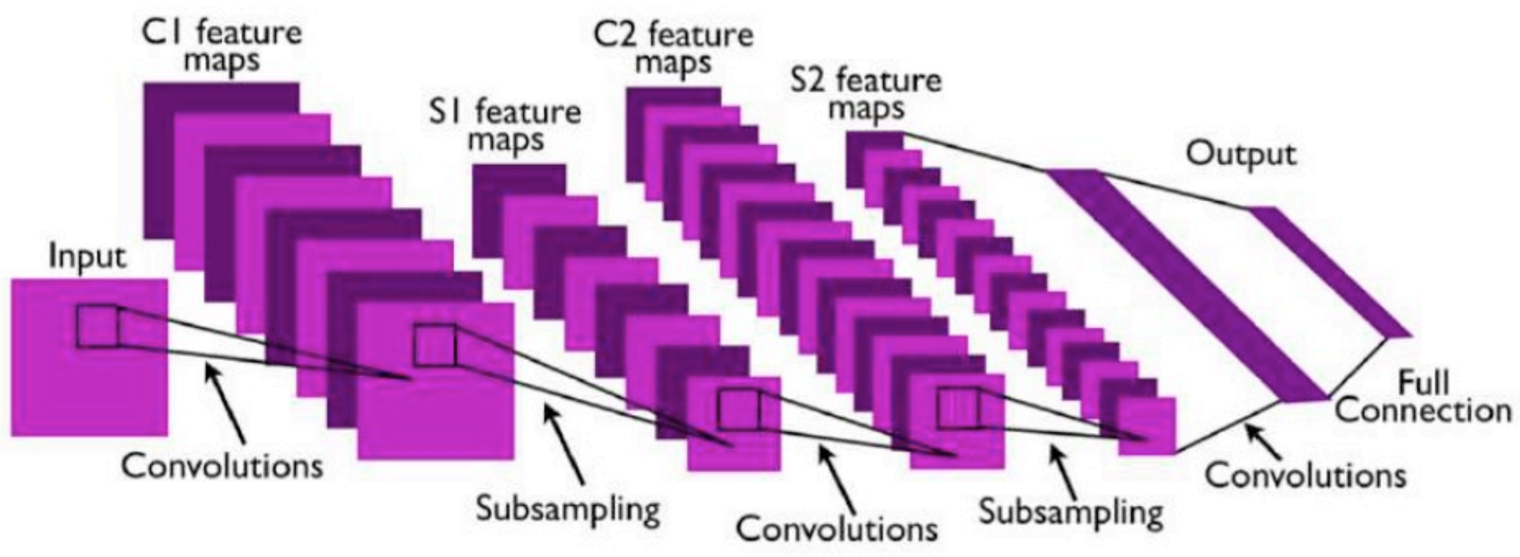


Convolution over several channels:

- e.g. RGB input
- Convolutional filter consists of one kernel per channel
- Sum over all channels

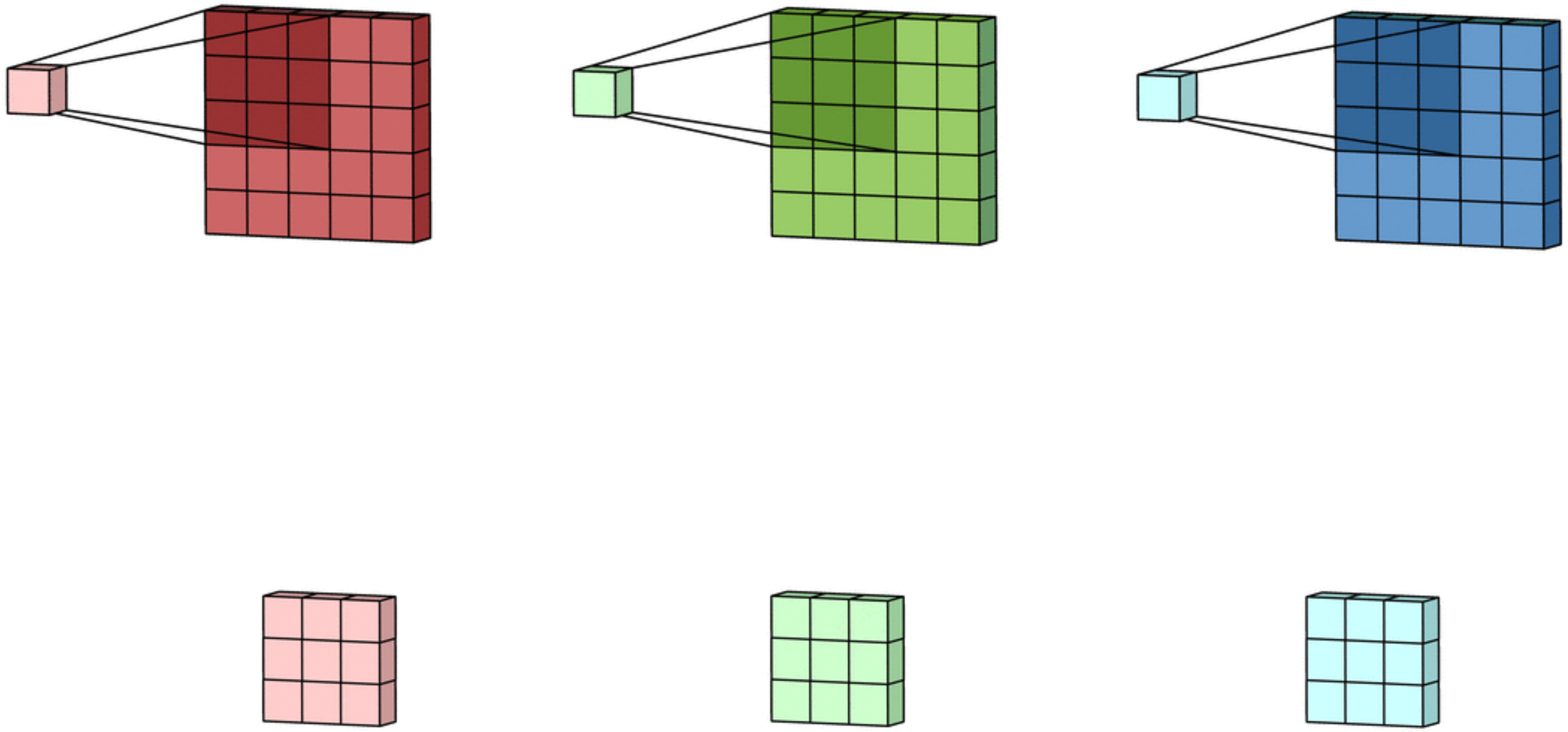


Convolution Operation in 2D

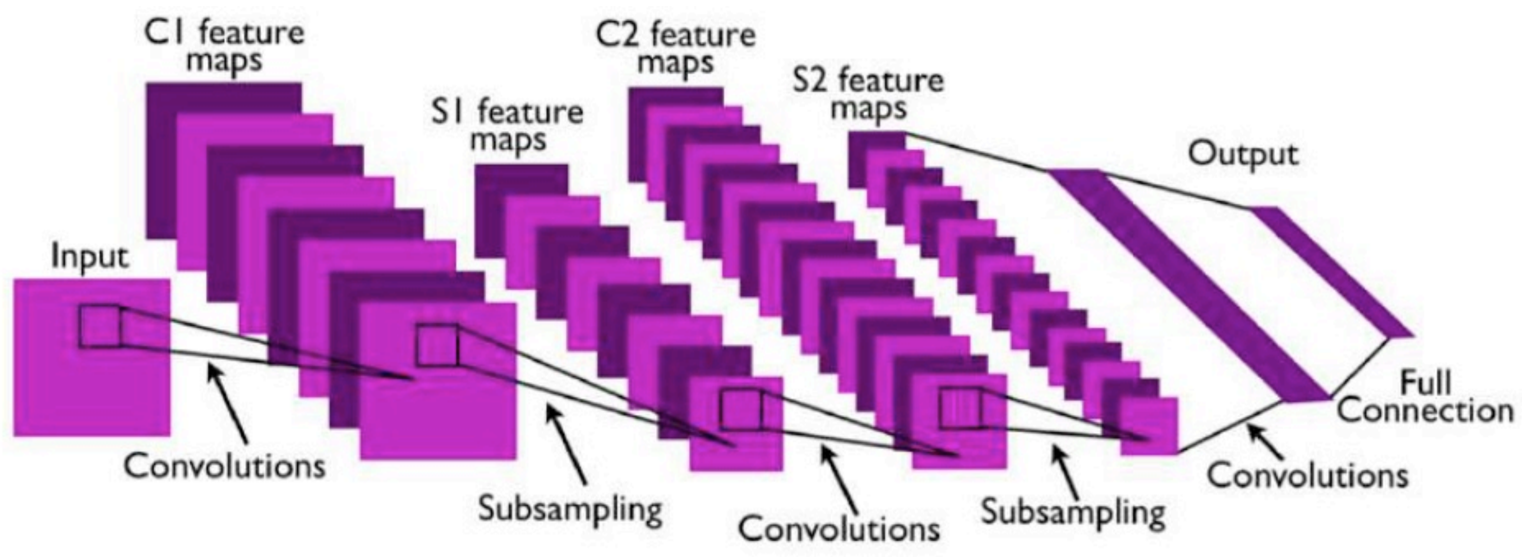


Convolution over several channels:

- e.g. RGB input
- Convolutional filter consists of one kernel per channel
- Sum over all channels



Pooling



Compute simple statistics over a local neighborhood.

Some pooling functions:

- **Max pooling:** output the max of the inputs.
- **Average pooling:** output the mean of the inputs.

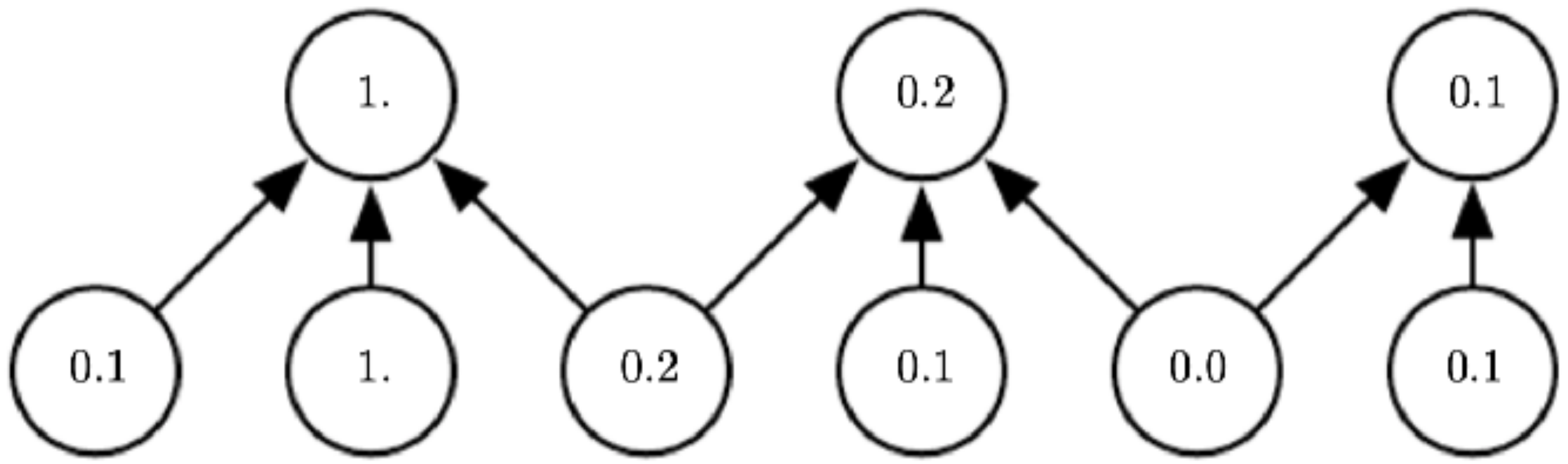
Application of pooling:

- One pooling layer per feature map.
- Stride of *k*: Move pooling window by *k* pixels when "moved" over the image.
  - Leads to a subsampling of the feature map (reduction of feature map size)

Examples:

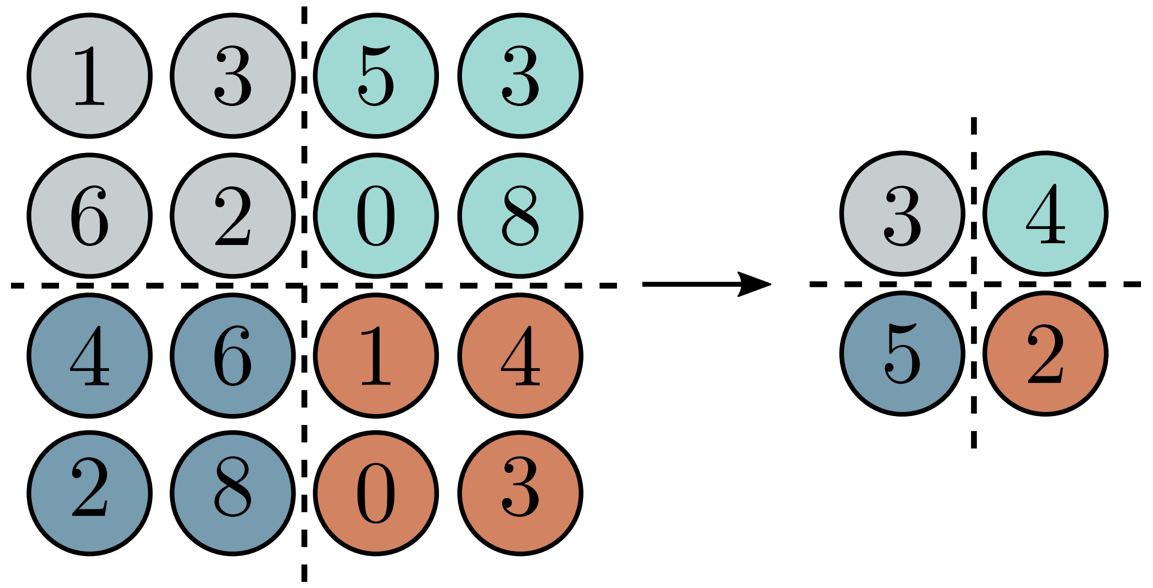
- Max-pooling (1D):

  pooling width: 3, stride: 2

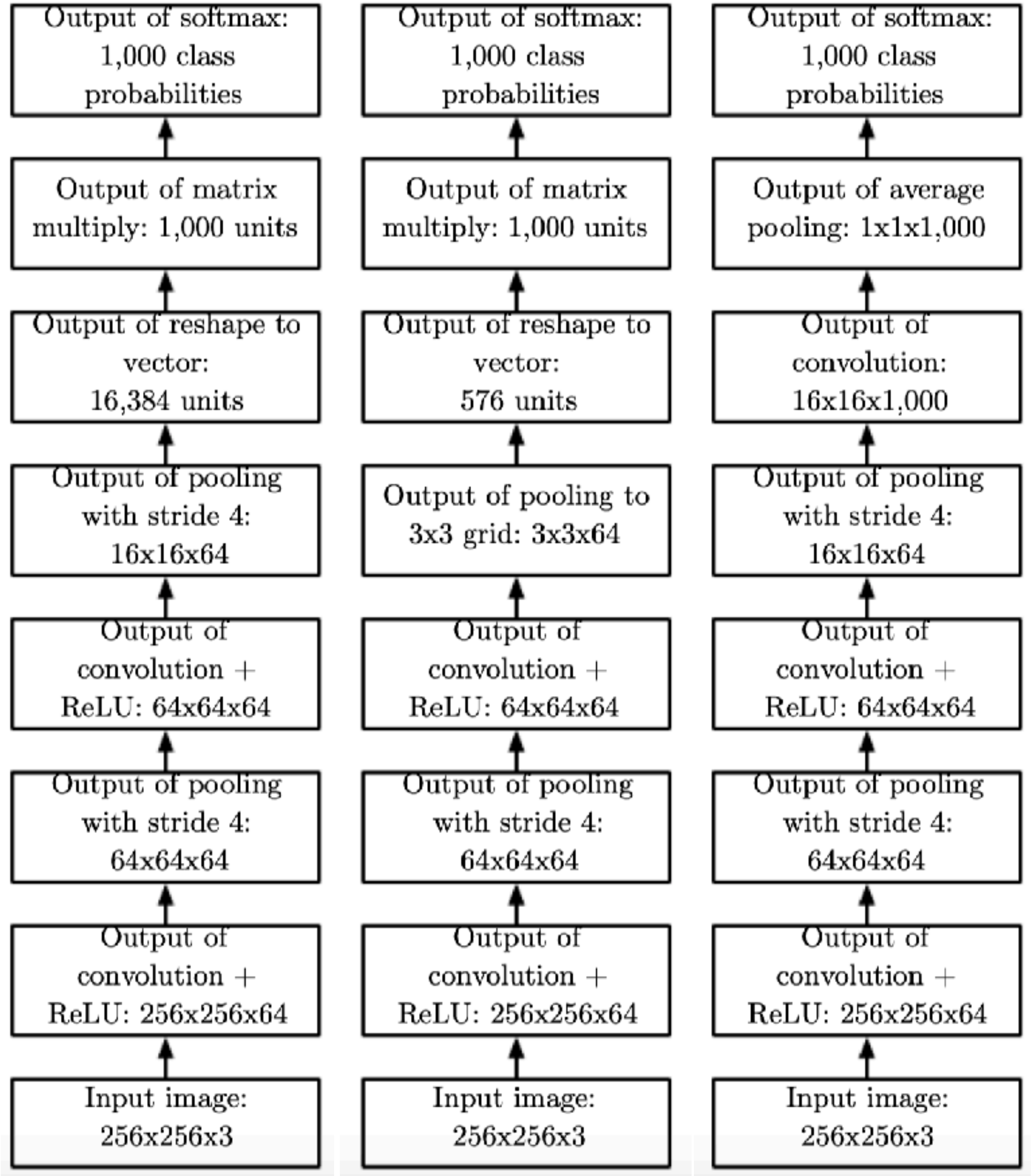


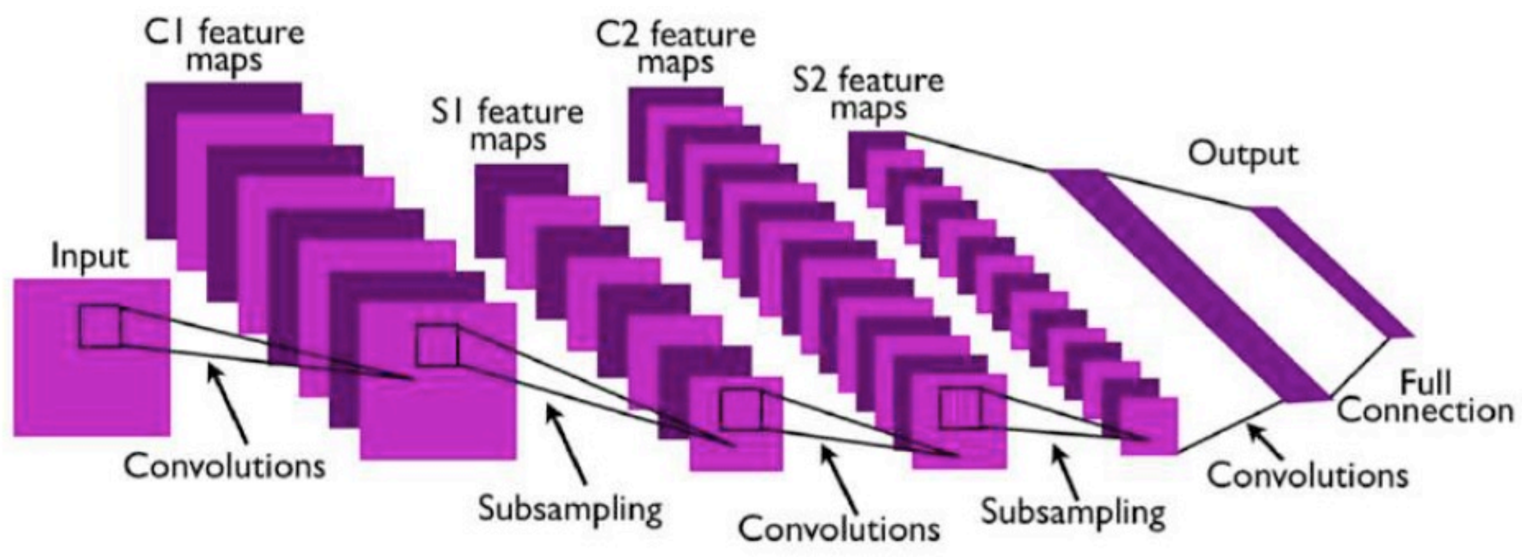
- Avg-pooling (2D):

  pooling size: (2,2), stride: (2,2)

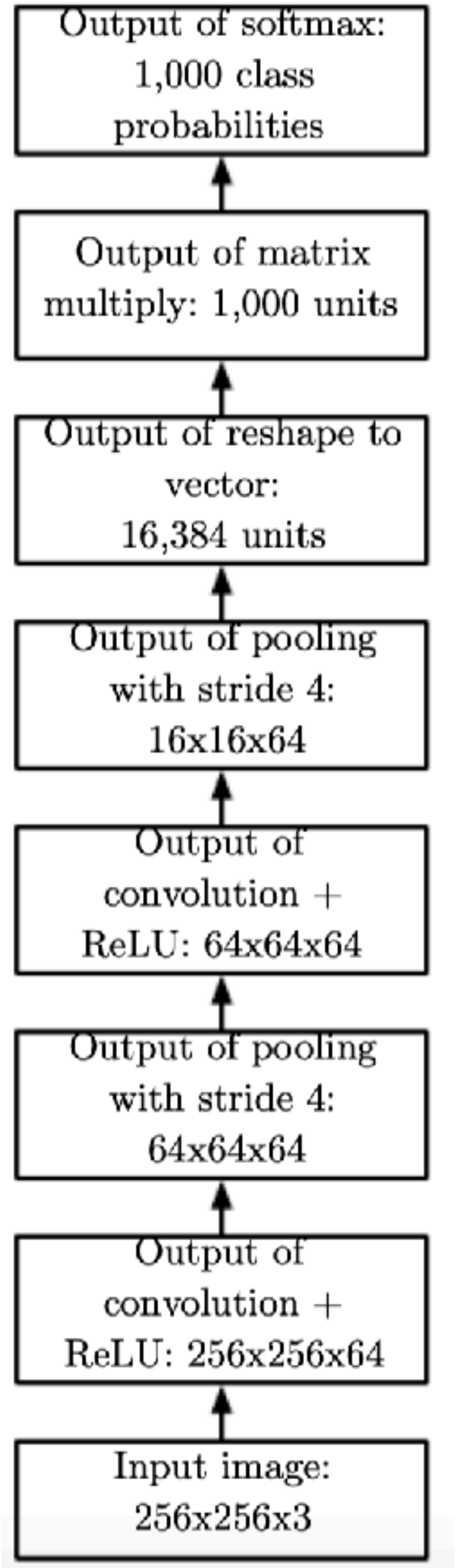


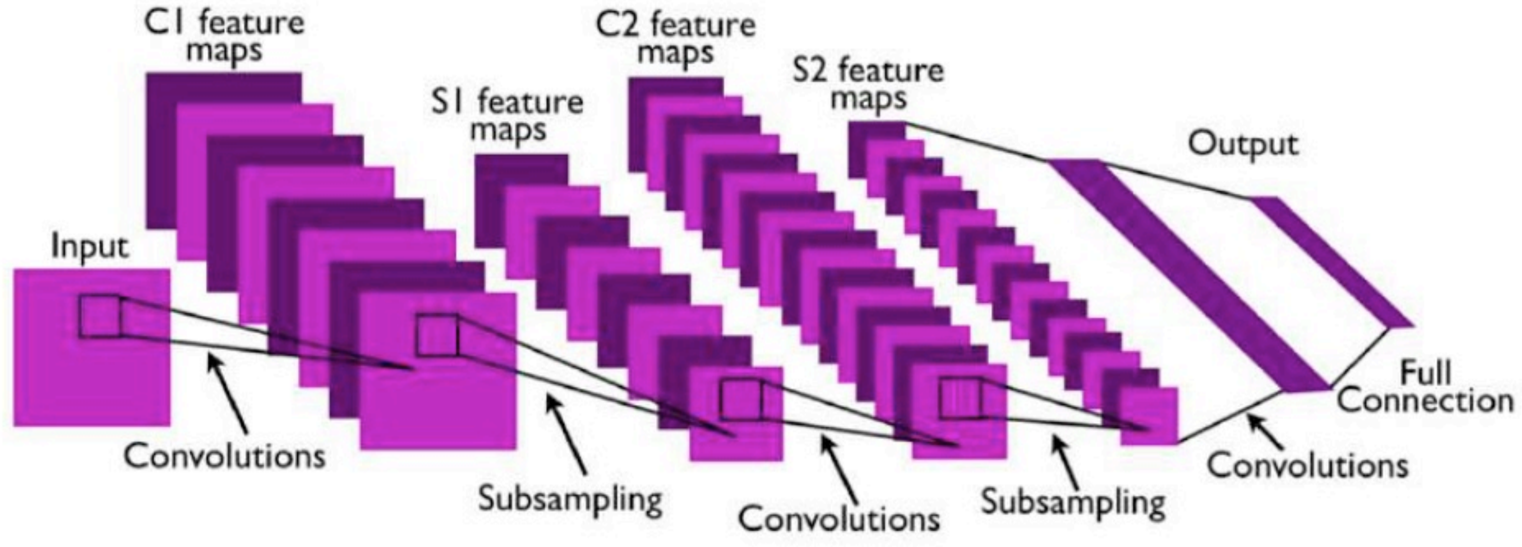
Illustrative Example Architectures





Illustrative Example Architecture





Conclusions from CNNs:

Advantages:

- Effective in avoiding overfitting.
- # of parameters drastically decreased in comparison to fully-connected nets.
- State-of-the-art in image classification.

(coming up next!)

Disadvantages:

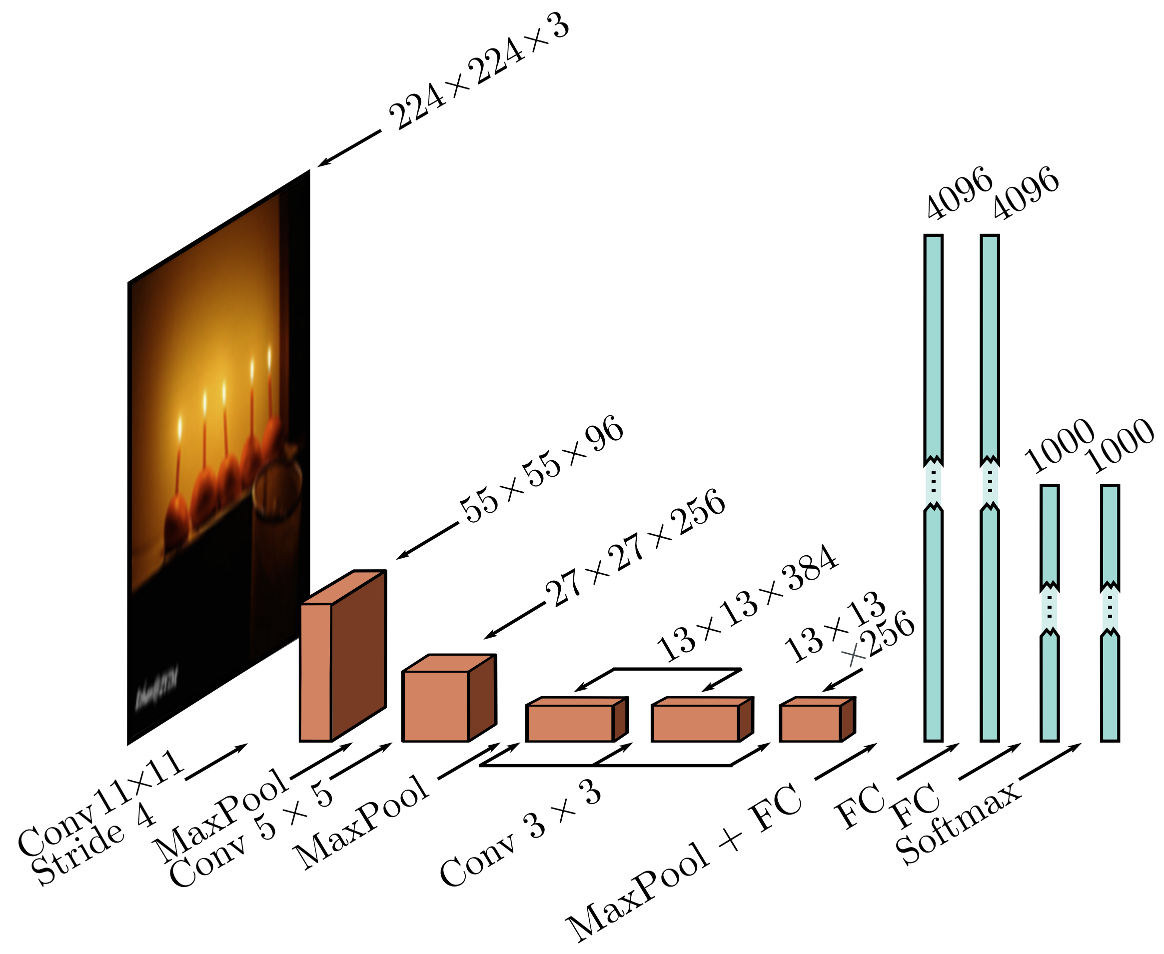
- Only applicable to data with invariance properties (e.g., image data).

Today

- ☐ Convolutional Neural Networks
  - ☑ Background
  - ☑ CNN Architecture
  - ☐ Going Deeper with CNNs
  - ☐ Residual Networks (ResNets)

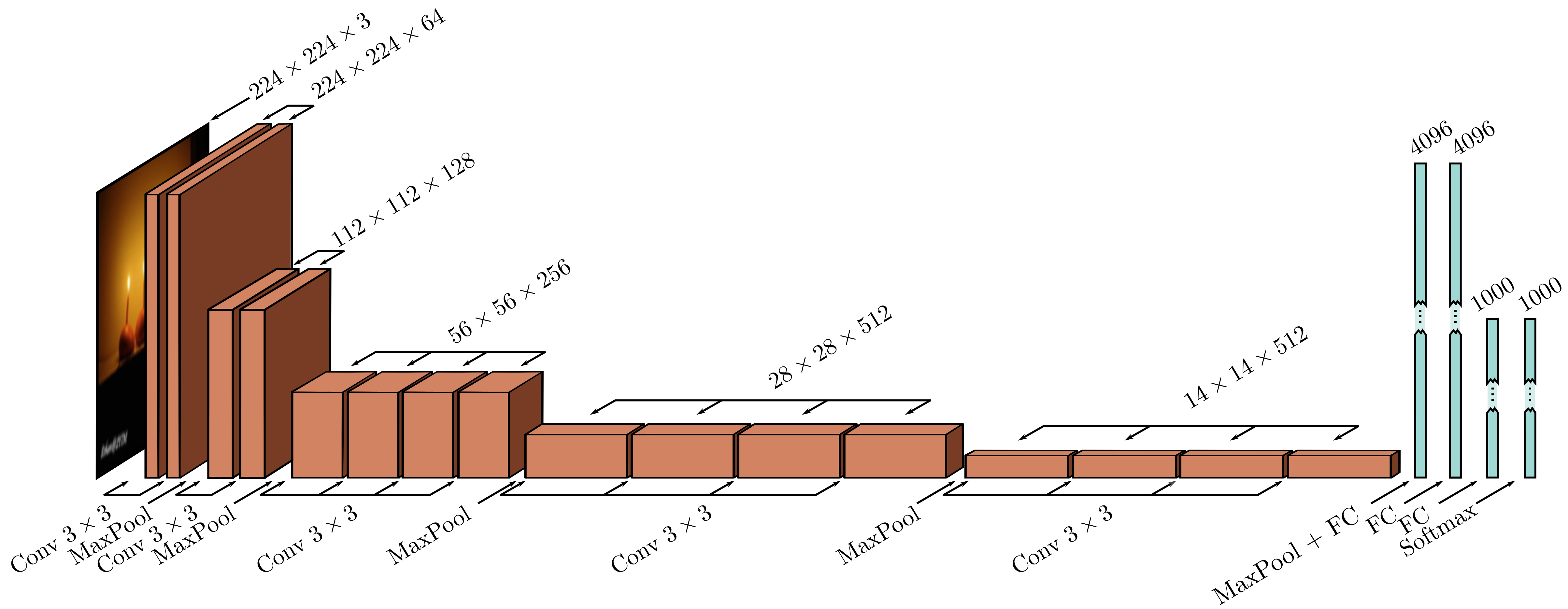
Going deeper with neural networks

AlexNet: CNN with 5 convolutional layers.



[A. Krizhevsky et al. (2012): “ImageNet Classification with Deep Convolutional Neural Networks”]

VGG: CNN with 19 layers.



[K. Simonyan, and A. Zisserman (2014): “Very deep convolutional networks for large-scale image recognition”]

Going deeper with neural networks

Observations:

- Very deep networks often show worse performance, even in terms of the **training error**.
- This is surprising, since the top layers could simply compute an identity mapping.
  - Hence, performance should be at least as good as a shallower network.

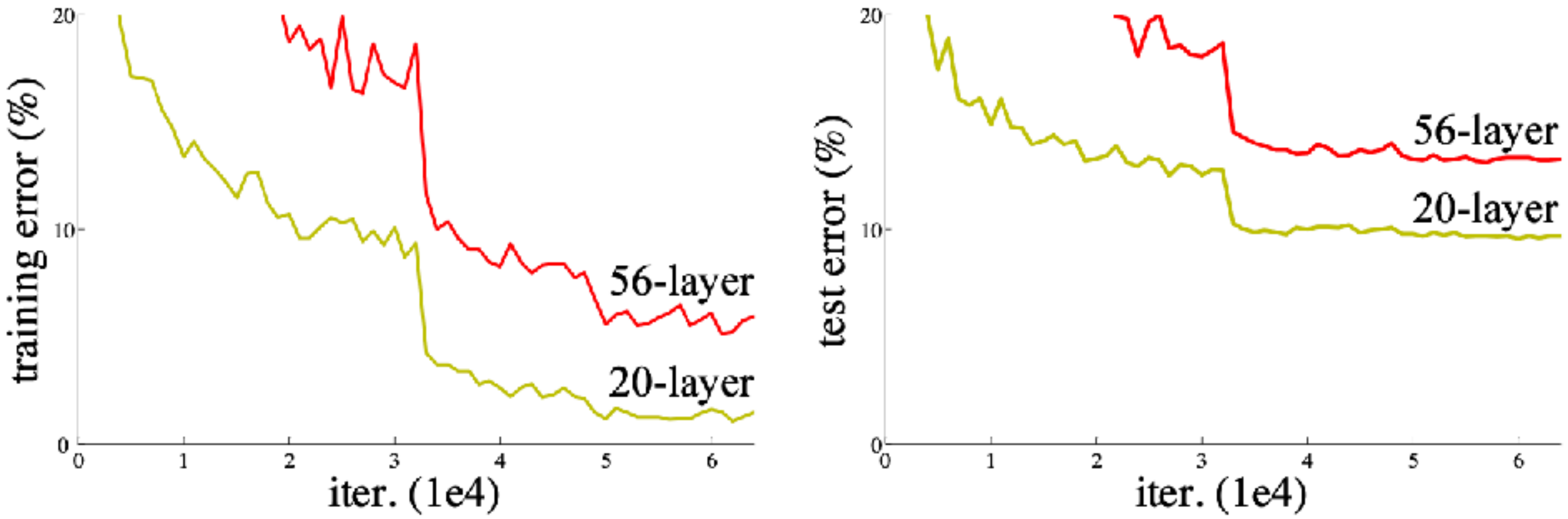
Deep networks are hard to optimize

Observations:

- Very deep networks often show worse performance, even in terms of the training error.
- This is surprising, since the top layers could simply compute an identity mapping.
    - Hence, performance should be at least as good as a shallower network.

Problem:

- Deeper networks are harder to train.
- “Vanishing gradient” problem.



[Kaiming He, et al. “Deep residual learning for image recognition”, CVPR 2016]

Previous solution:

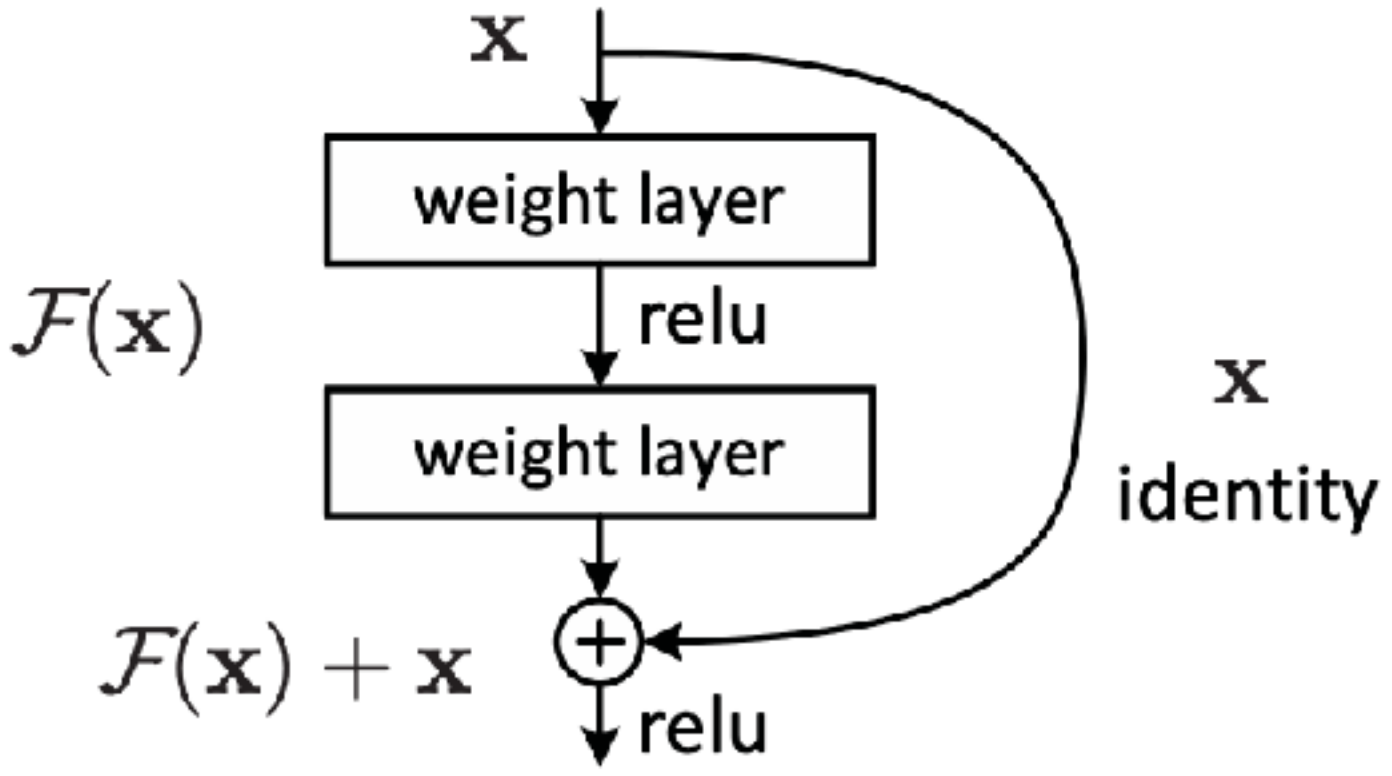
- Auxiliary loss in middle layers (e.g., GoogLeNet)

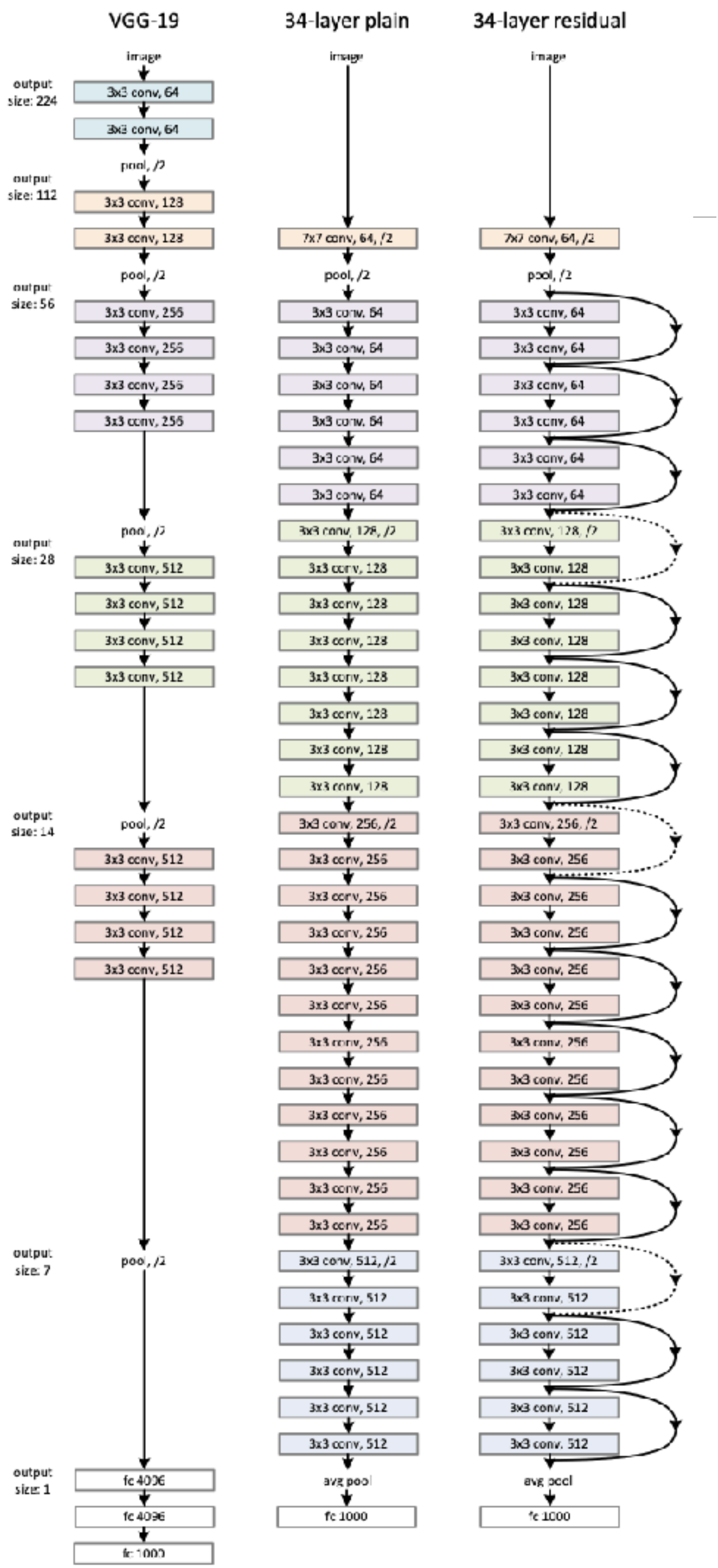
Today

- ☐ Convolutional Neural Networks
  - ☑ Background
  - ☑ CNN Architecture
  - ☑ Going Deeper with CNNs
  - ☐ Residual Networks (ResNets) & Extensions

Residual Networks (ResNets)

Core idea: Introduce “identity shortcut connections”





- Any additional “residual layer” should not hurt performance, as the network can simply use the identity.
- Network can however improve by adding something to the identity.

▶ ResNets with hundreds or thousands of layers can be trained.

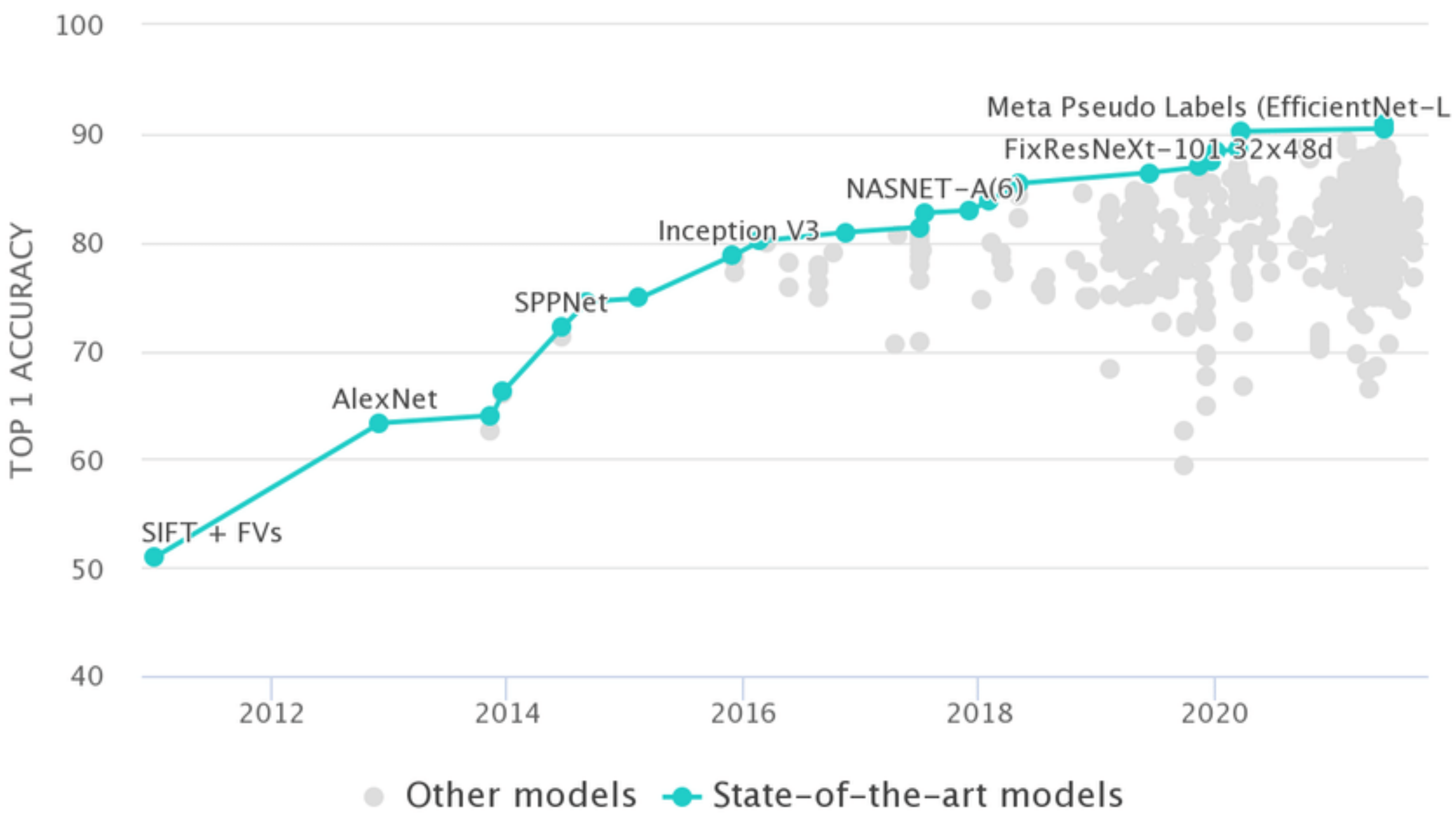
▶ Best-performing model on many datasets (including ResNet variants).

[Kaiming He, et al. “Deep residual learning for image recognition”, CVPR 2016]

Summary

- Inspired by the visual system of vertebrates.
  - Hierarchical organization, increasing receptive fields, increasingly complex features.
- CNNs are feed-forward networks.
- Weight sharing reduces the # of parameters.
- Suitable for data with local structure and invariance properties.
- Deeper networks are hard to train:
  - Vanishing gradients problem.
  - Residual layers help.
- State-of-the-art in image classification.

Source: https://paperswithcode.com/sota/image-classification-on-imagenet/



Today

☑ Convolutional Neural Networks

- ☑ Background
- ☑ CNN Architecture
- ☑ Going Deeper with CNNs
- ☑ Residual Networks (ResNets) & Extensions

Questions?