Convolutional Neural Networks

Paolo Favaro
UNIVERSITÄT BERN

Contents

- Motivation and classic pattern recognition, Convolutional Neural Networks
    - Convolutions (standard, unshared, tiled), pooling, structured outputs
- Based on **Chapter 9** of Deep Learning by Goodfellow, Bengio, Courville
- Credits also to Yan LeCun “Learning Invariant Feature Hierarchies” presentation

Pattern Recognition

- The task is to classify an image as one of several possible objects

x

car

bicycle

pedestrian

y

p(y|x)

Pattern Recognition

- The classic pipeline

handcrafted feature extractor → trainable (linear) classifier → car / bicycle / pedestrian

fixed

supervised

Pattern Recognition

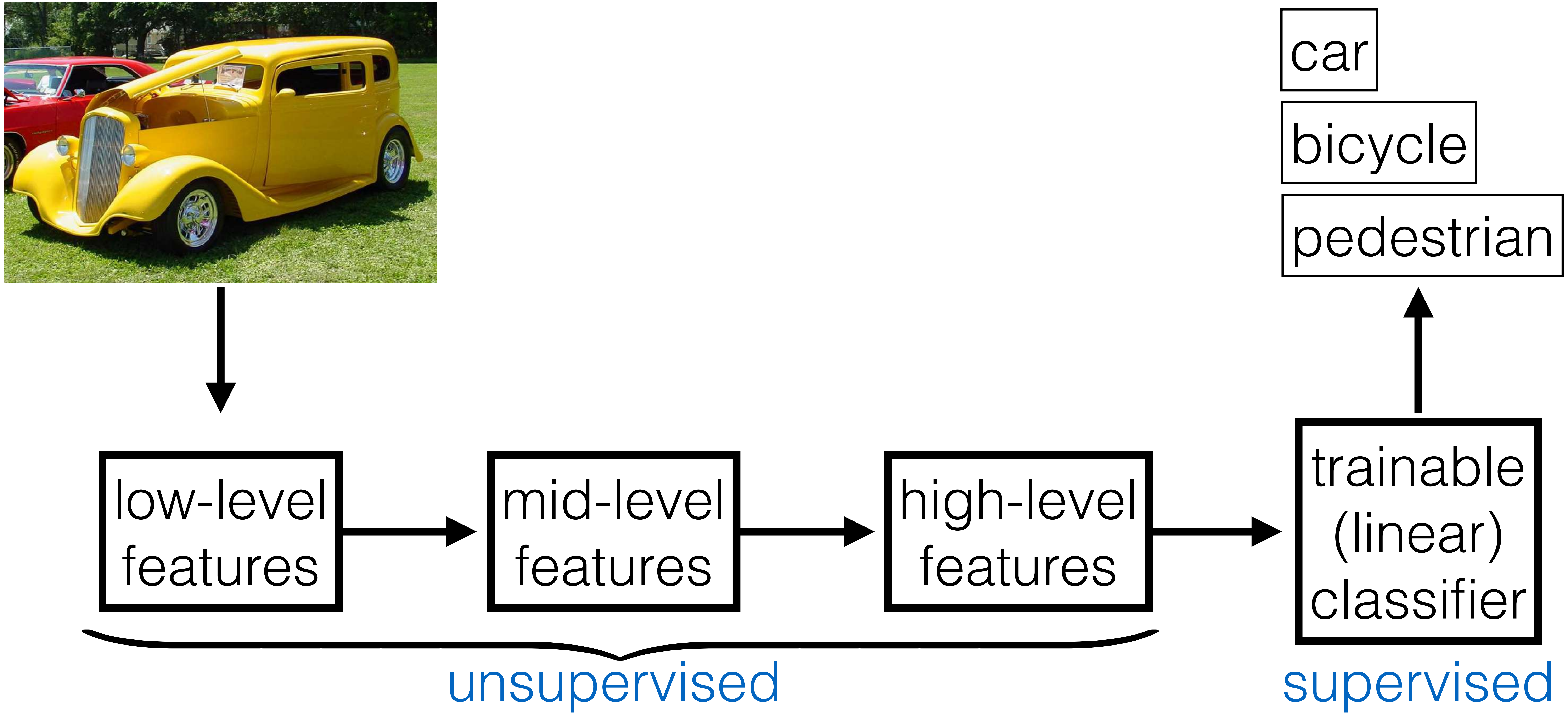
- The modern pipeline (2000s)

handcrafted features (SIFT, HOG) → k-means sparse coding → trainable (linear) classifier → car / bicycle / pedestrian

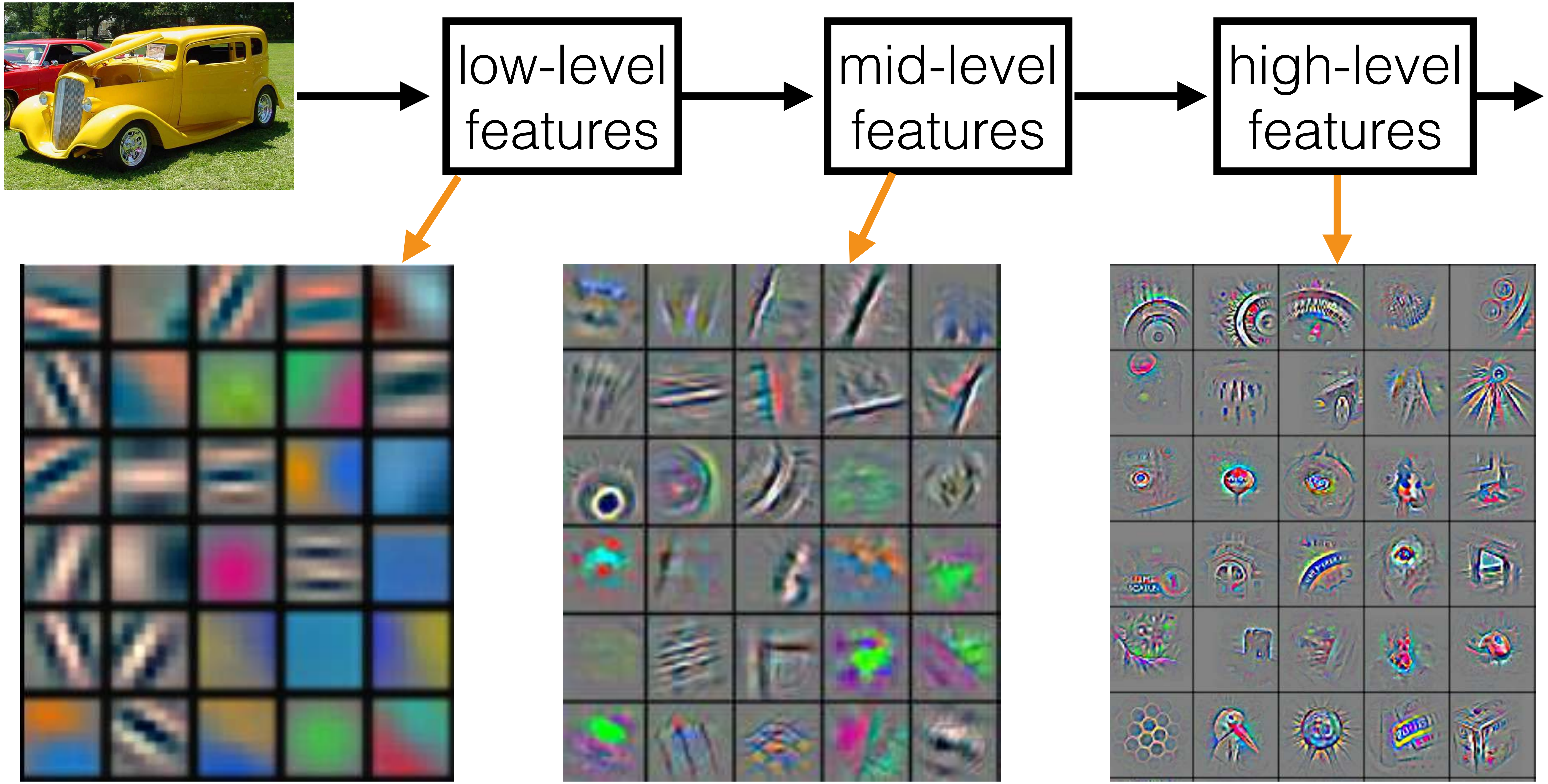
fixed | unsupervised | supervised

Pattern Recognition

- Deep learning



Pattern Recognition



Trainable Feature Hierarchy

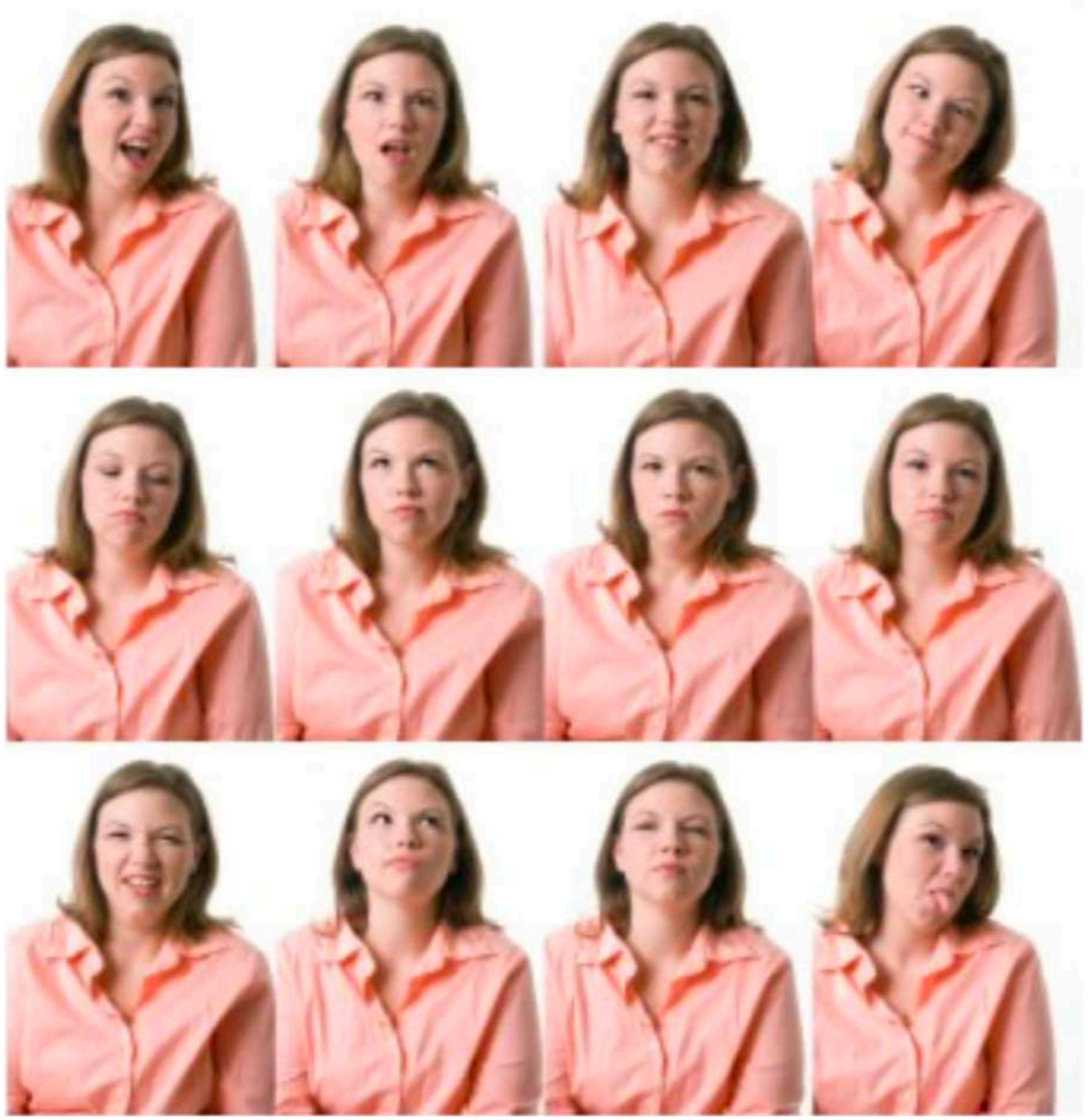
- Hierarchy builds increasing level of abstraction, where each level is trainable
- Hierarchy examples
  - Image: Pixel → edge → texton → motif → part → object
  - Text: Character → word → word group → clause → sentence → story

Learning Representations

- How do we learn representations of the perceptual world?
- How can a perceptual system build **itself** by looking at the world?
- How much prior structure is necessary?

What Are Good Features?

- Discover & disentangle the independent **explanatory factors**
- The manifold hypothesis
  - Natural data lives in a low-dim (non-linear) manifold, because variables in natural data are mutually dependent



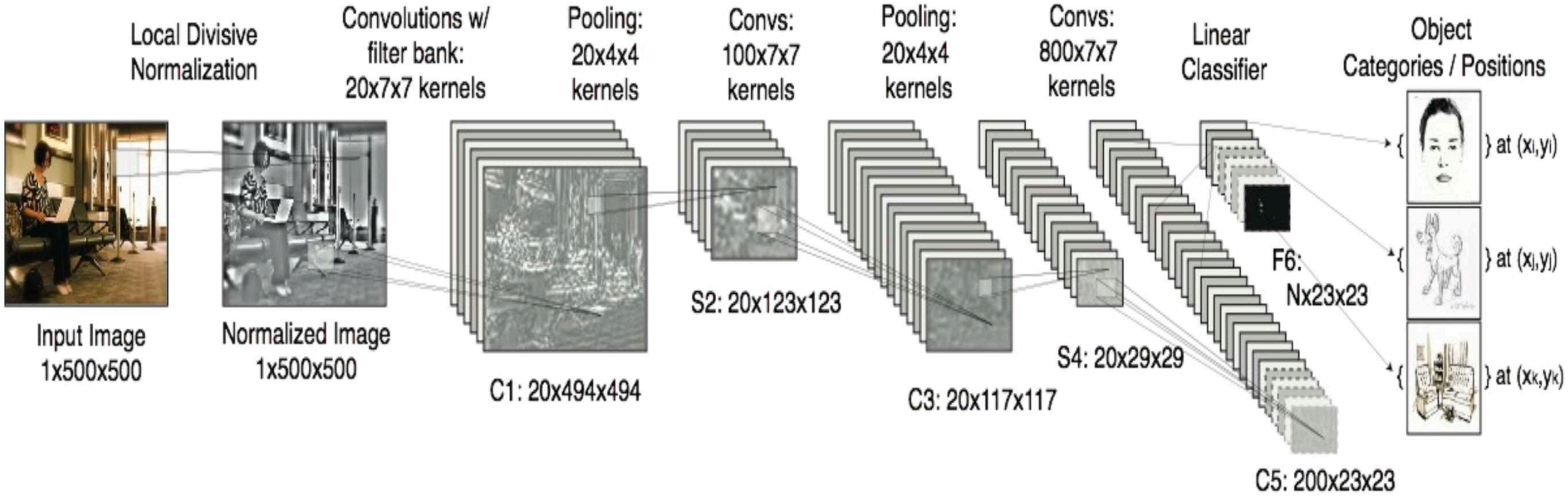
Feature Learning Scheme

- Embed features in a high dimensional space (data points become separable)
    - Pattern matching
    - Nonlinearity
- Pool regions that have similarities

Deep Learning Architecture

- Stack multiple stages of
  - **Normalization**: builds invariance to nuisance factors
  - **Filtering**: mapping to high-dim space
  - **NonLinearity**: separation of features
  - **Pooling**: aggregation by similarity

The Convolutional Network

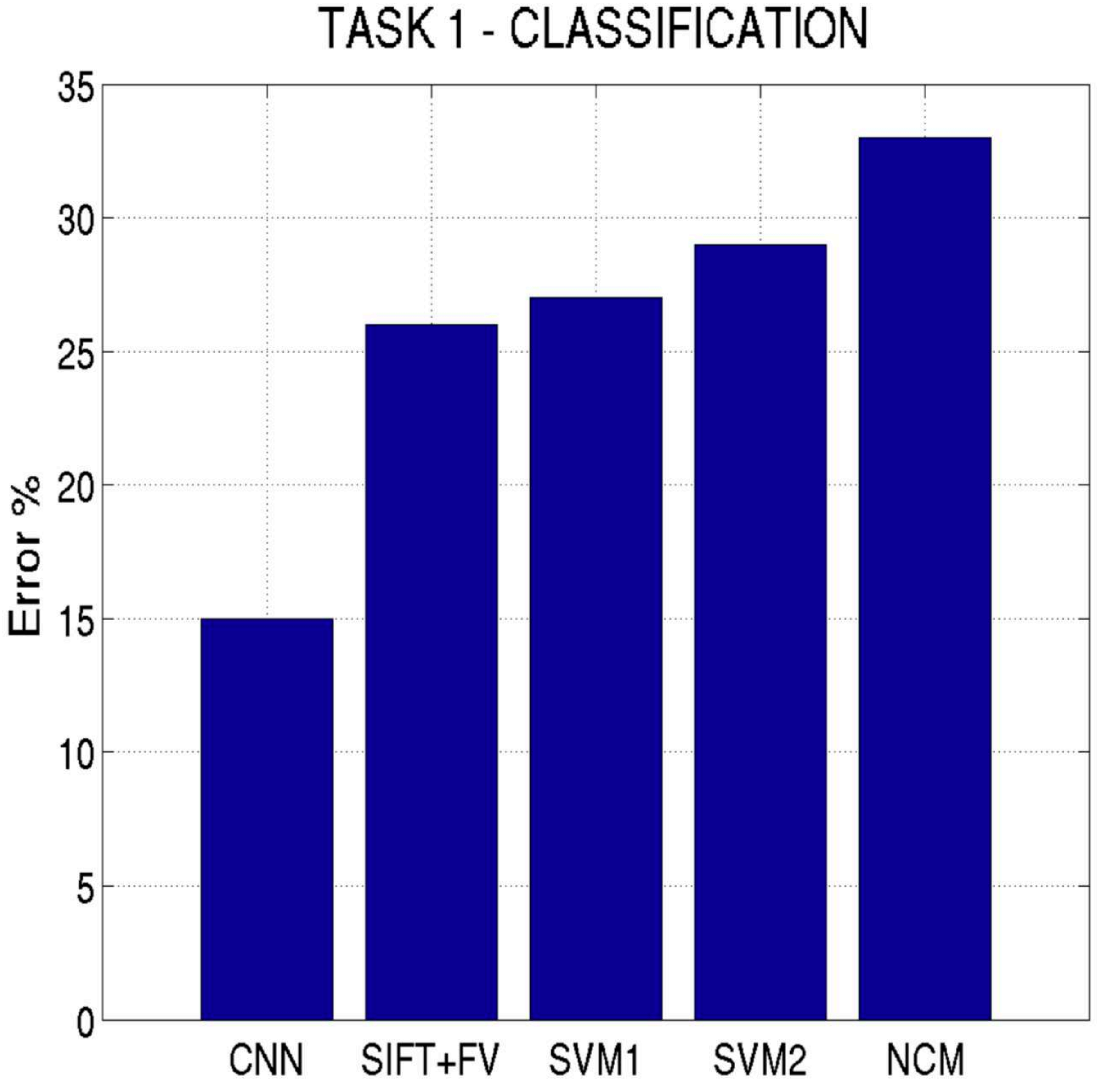


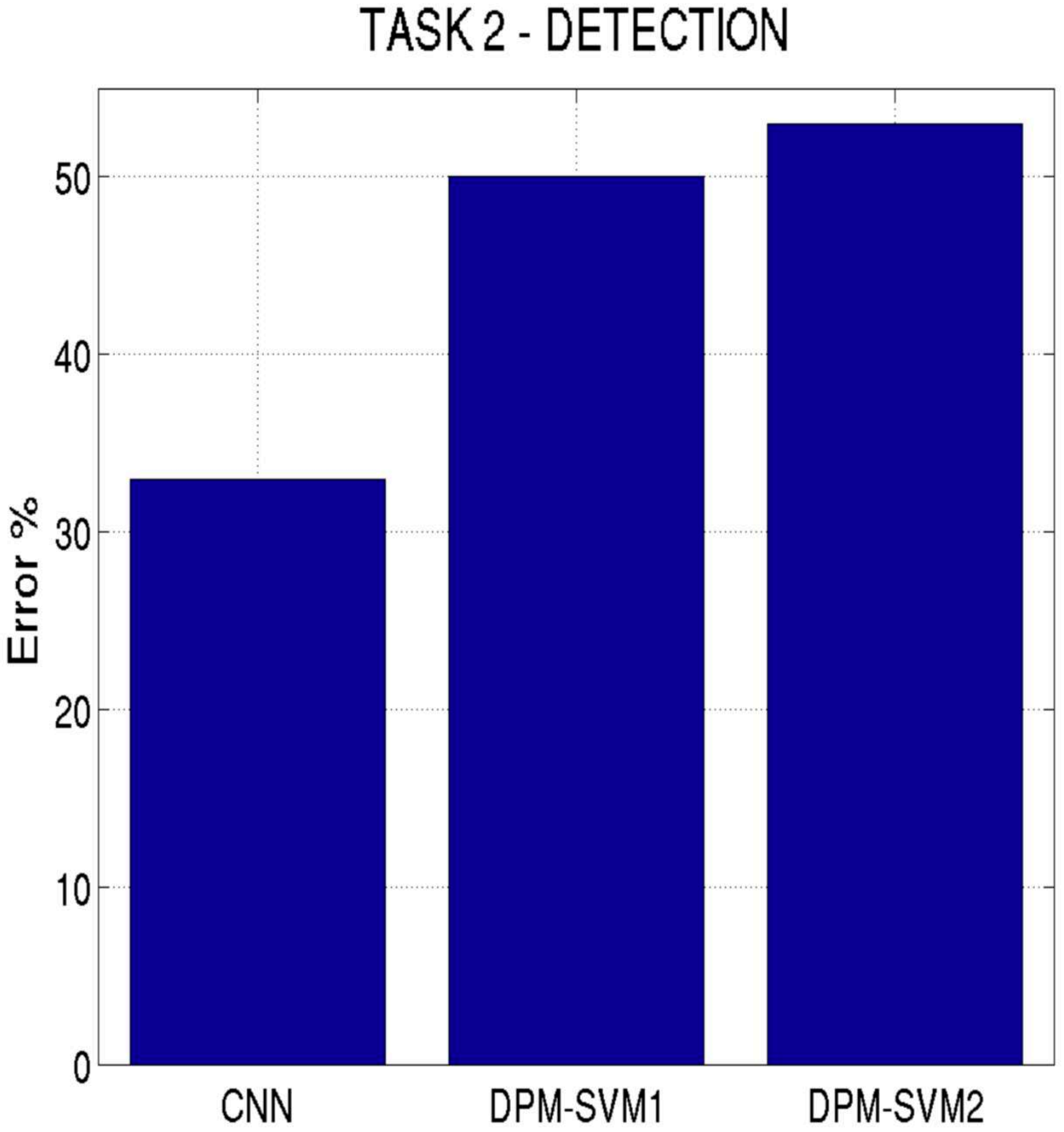
Convolutional Networks (ConvNets)

- Deployed in many practical applications
  - Image/speech reco, Google's and Baidu's photo taggers
- Have won several competitions
  - ImageNet, Kaggle Facial Expression/Multimodal Learning, German Traffic Signs, etc
- Applicable to array data where nearby values are correlated
  - Images, sound, time-frequency representations, video, volumetric images, RGB-Depth images, etc

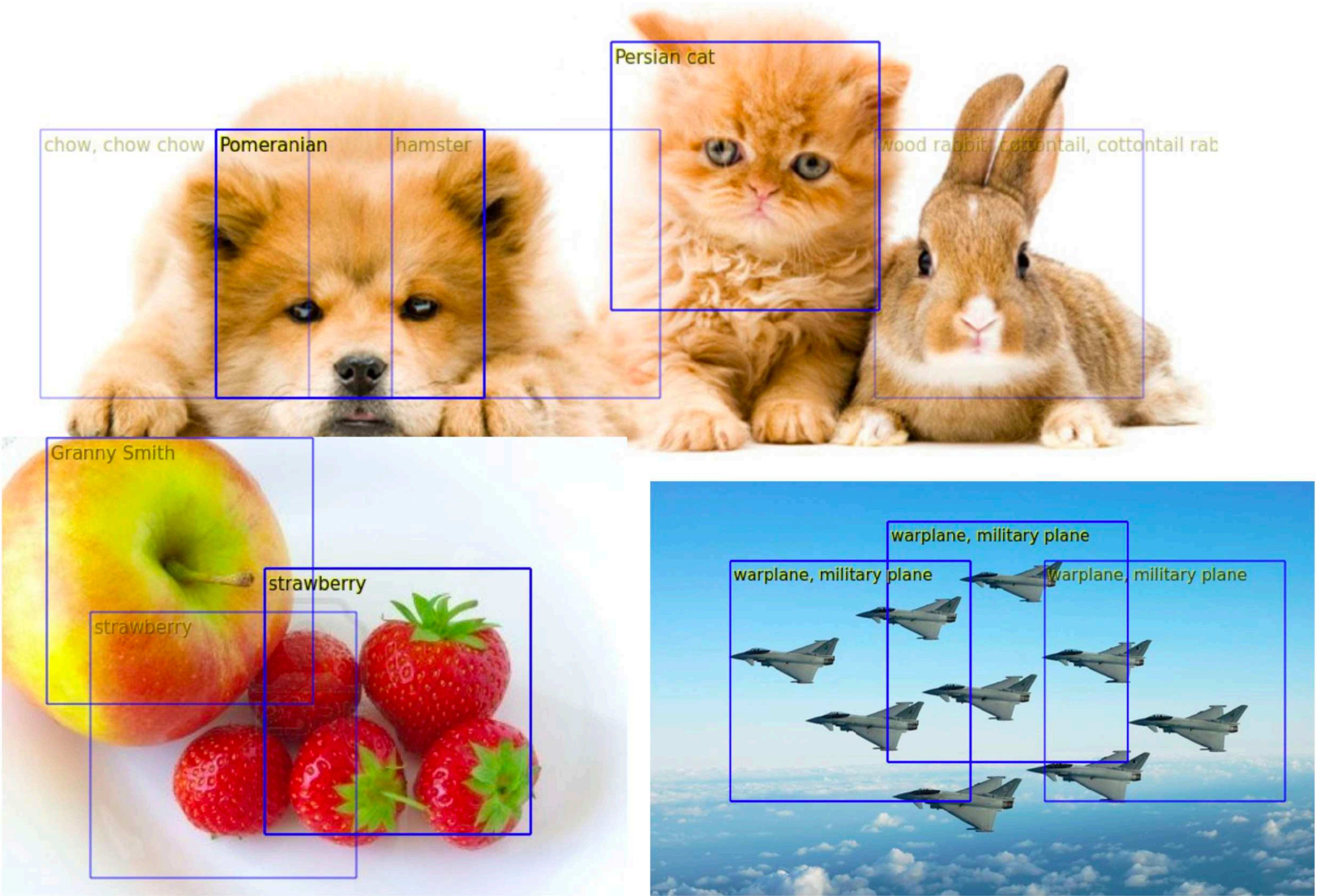
Object Recognition with ConvNets

- AlexNet (Krizhevsky, Sutskever, Hinton 2012) won the 2012 ImageNet LSVRC (1K categories, 1.3M labeled training samples)

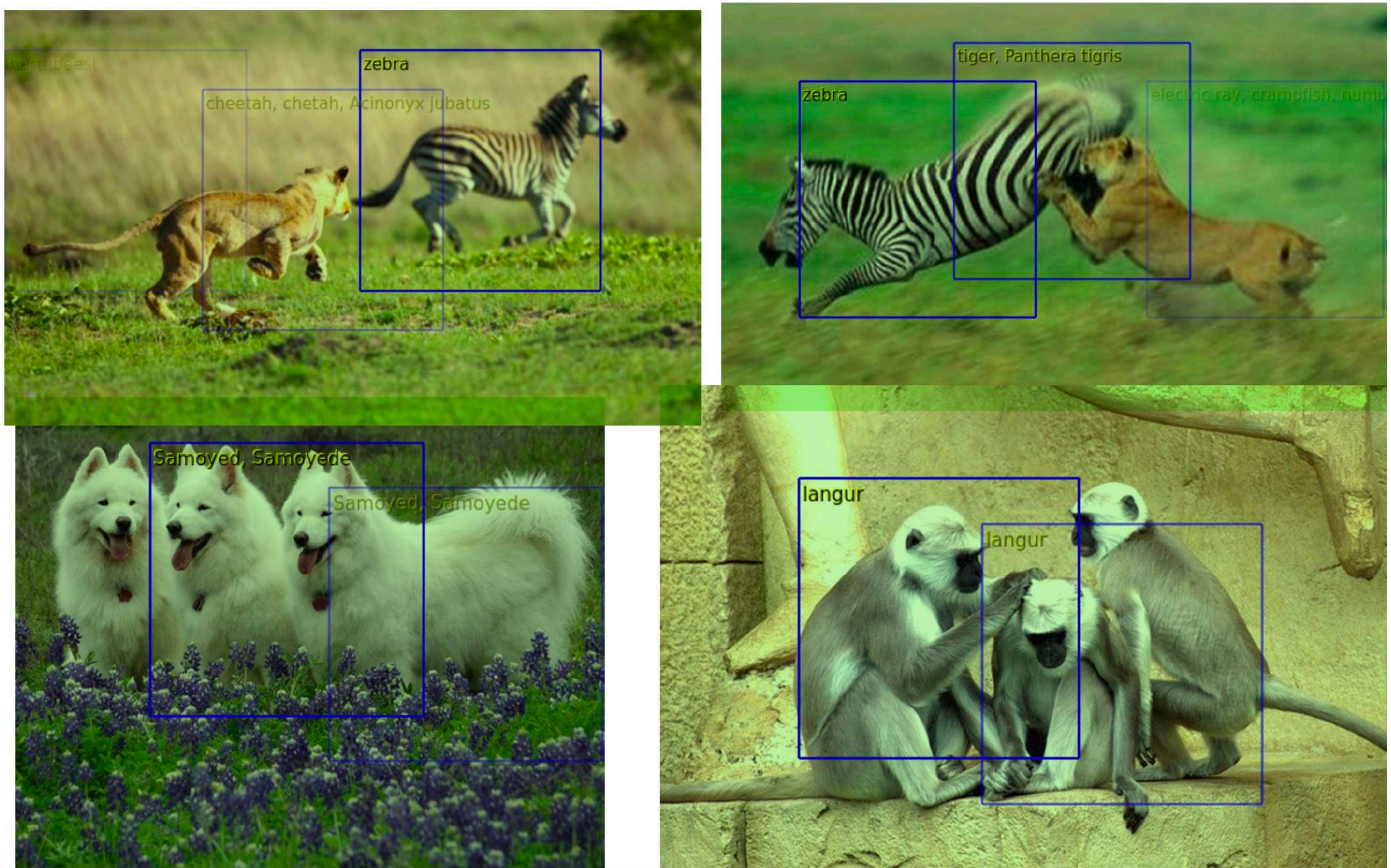




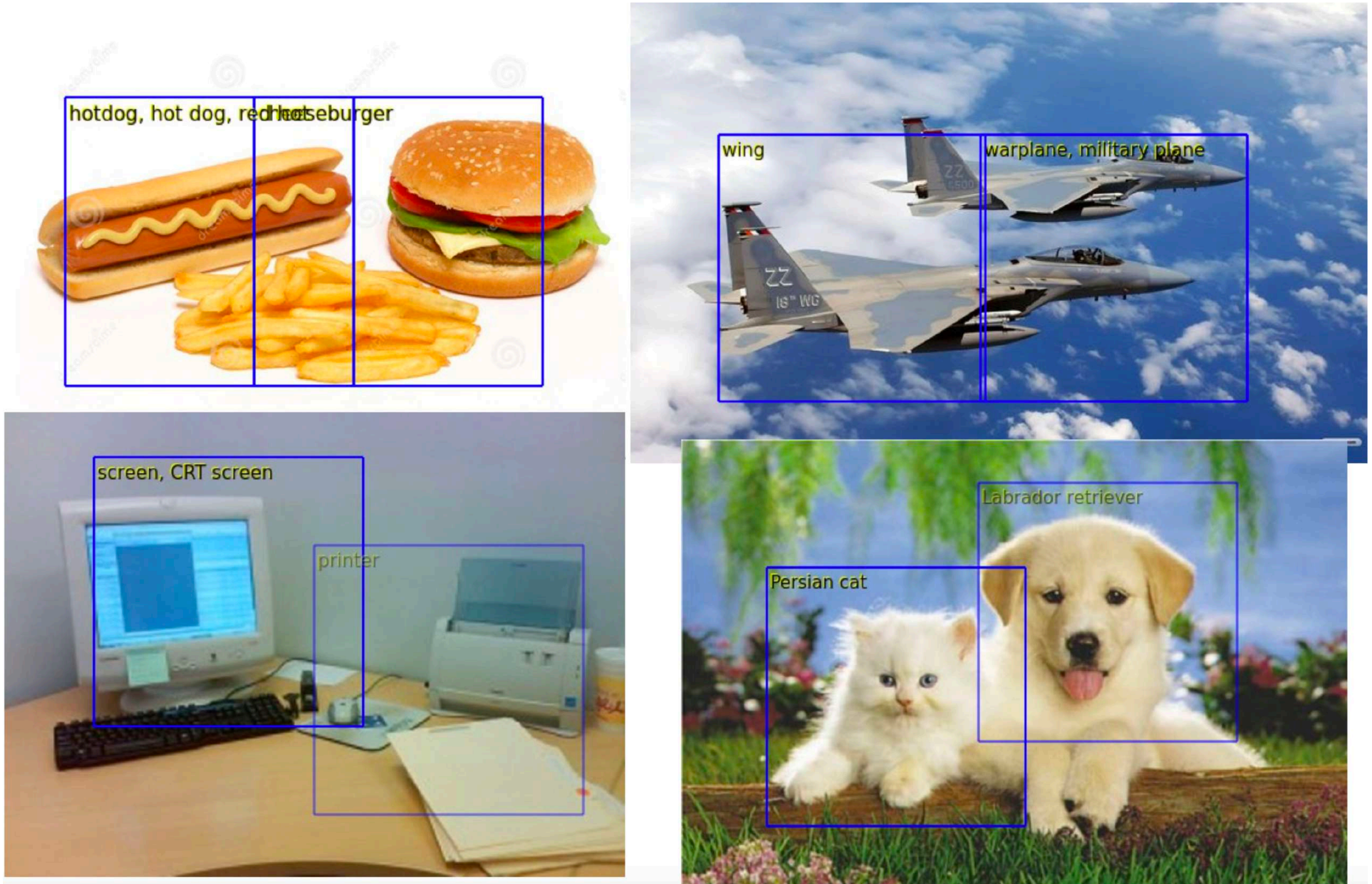
Detection with a Sliding Window



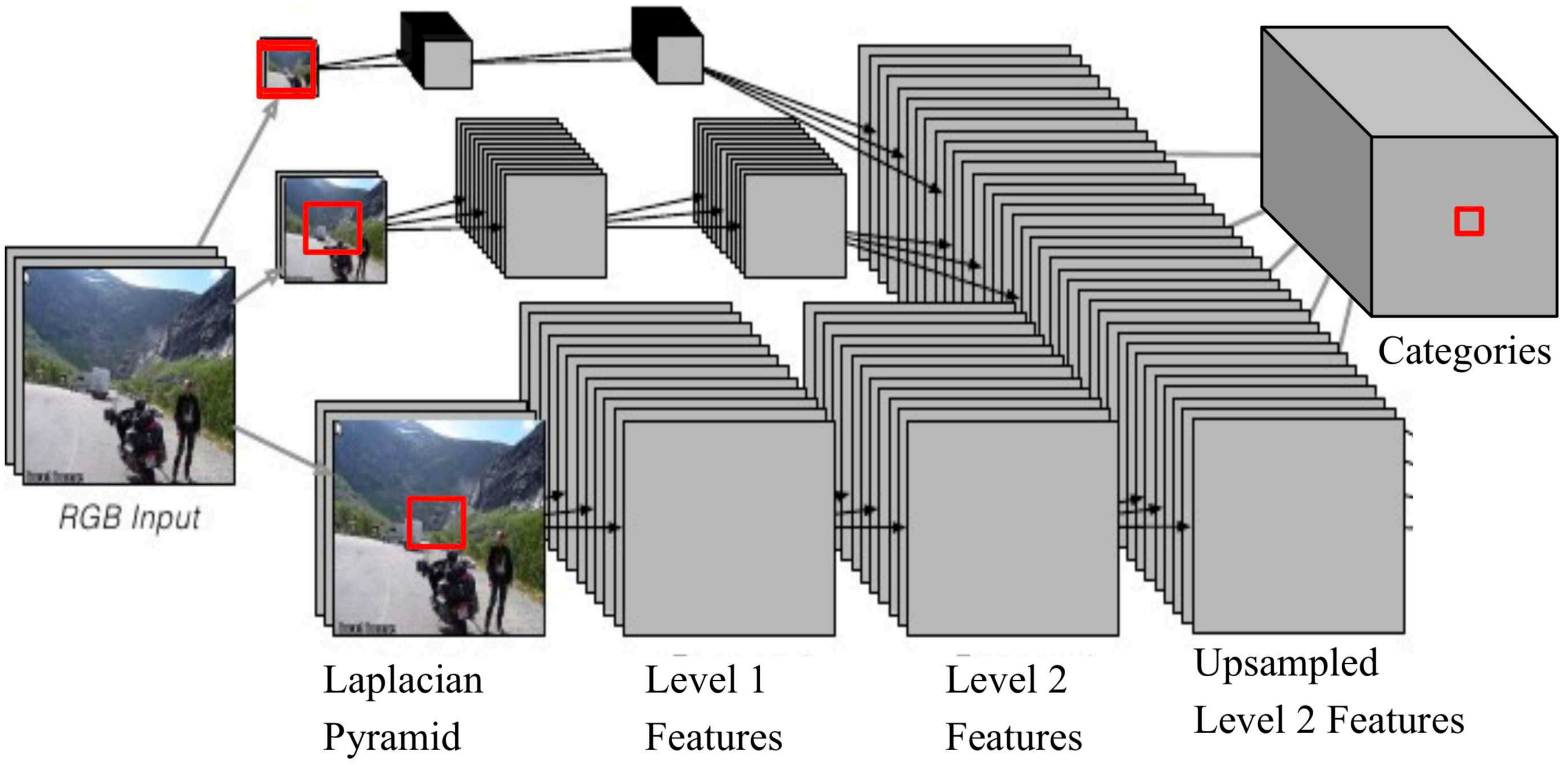
Detection with a Sliding Window



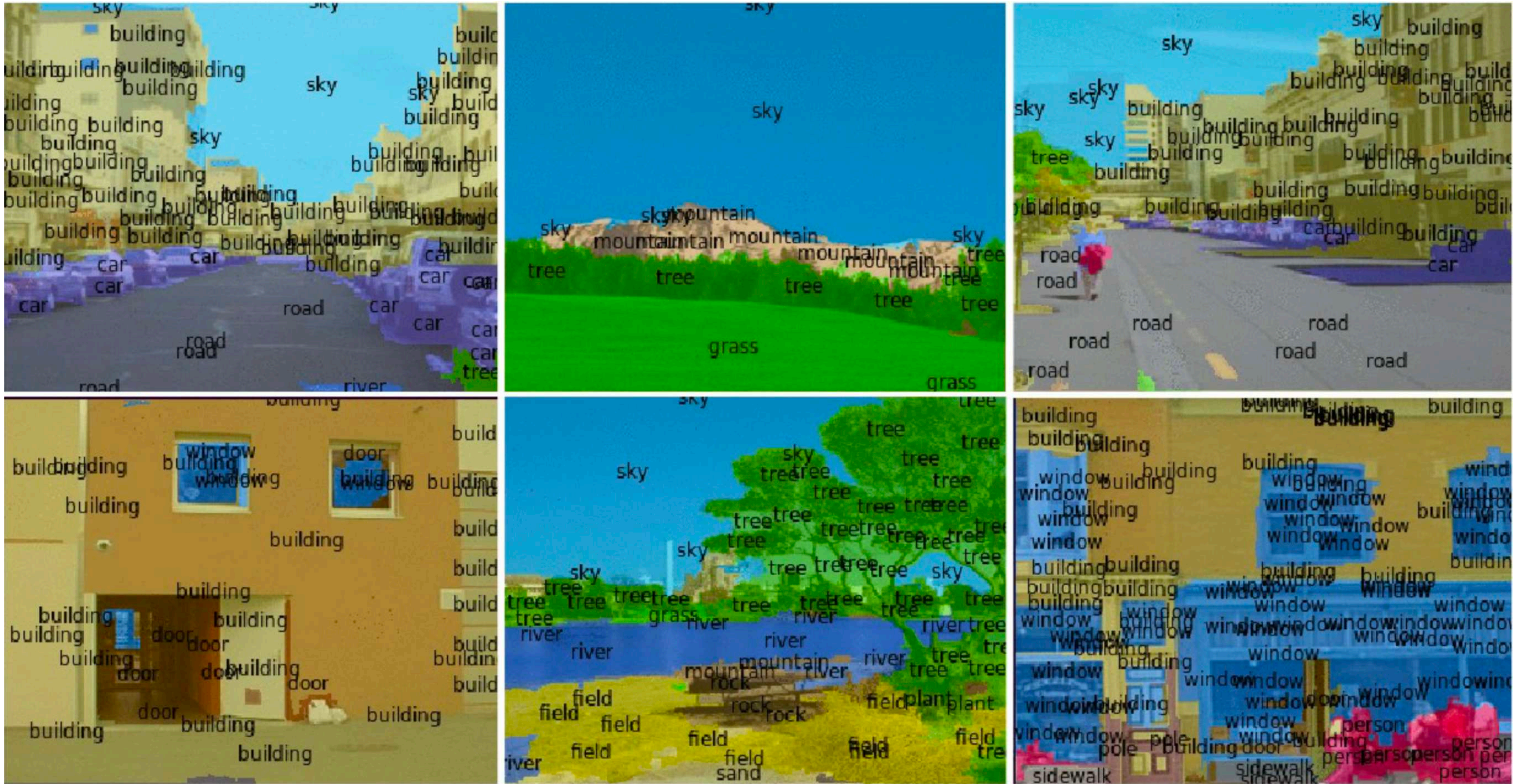
Detection with a Sliding Window



Semantic Segmentation with ConvNets

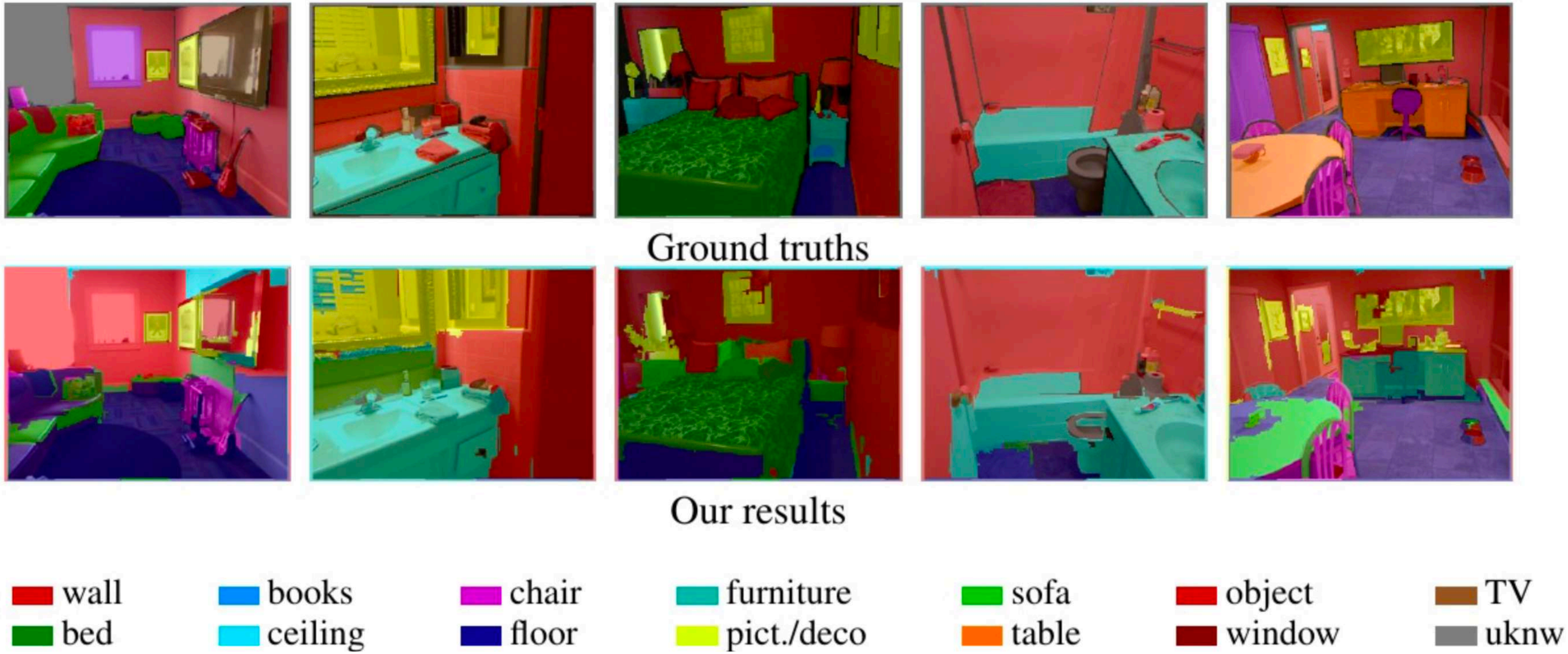


Semantic Segmentation with ConvNets



Semantic Segmentation with ConvNets

Semantic Segmentation with ConvNets



Scene parsing on RGB-depth images

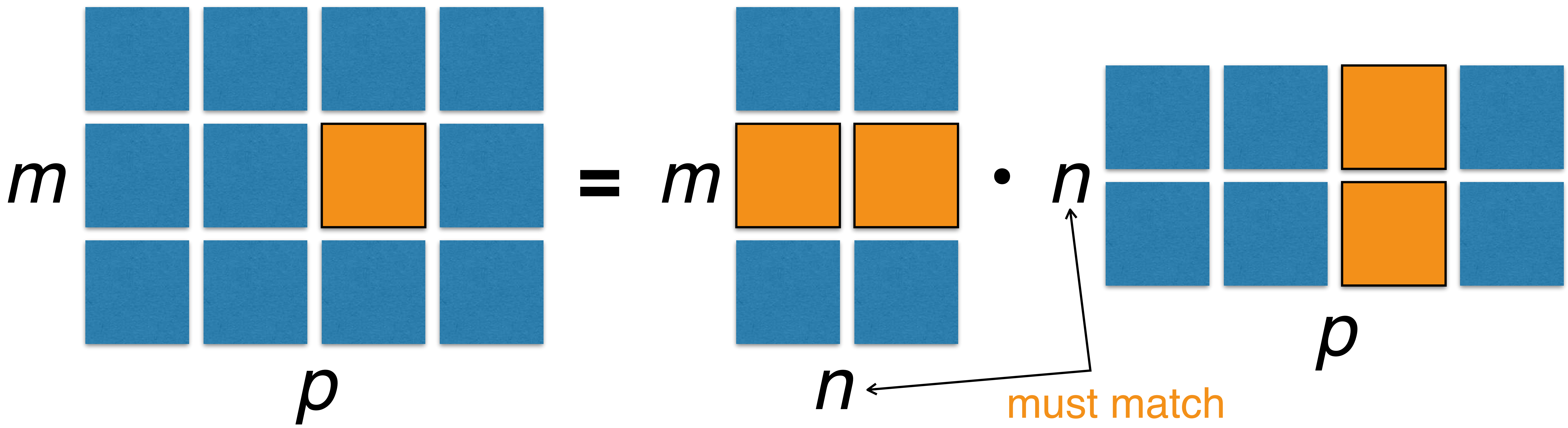
Commercial Applications with ConvNets

- Form Reading: AT&T 1994
- Check reading: AT&T 1996 (read 10-20% of all US checks in 2000) Handwriting recognition: Microsoft early 2000
- Face and person detection: NEC 2005
- Face and License Plate Detection: Google/StreetView 2009
- Gender and age recognition: NEC 2010 (vending machines)
- OCR in natural images: Google 2013 (StreetView house numbers) Photo tagging: Google 2013
- Image Search by Similarity: Baidu 2013
- Today: Lots of google services, etc

Convolutional Networks

- A specialized neural network for data arranged on a grid (e.g., audio signals, images)
- Allow neural networks to deal with high-dimensional data
- Key idea is to substitute fully connected layers with a convolution

Fully Connected Layers



matrix product

The Convolution Operation

feature map

input

kernel

$$s[m,n] = (x * w)[m,n] = \sum_{i,j} x[m-i, n-j] w[i,j]$$

symmetric

$$= \sum_{i,j} w[m-i, n-j] x[i,j]$$

linear in x
with fixed w

The Correlation Operation

symbol

sign

$$s[m,n] = (x \otimes w)[m,n] = \sum_{i,j} x[m+i, n+j]w[i,j]$$

not symmetric as the convolution

$$= \sum_{i,j} w[i-m, j-n]x[i,j]$$

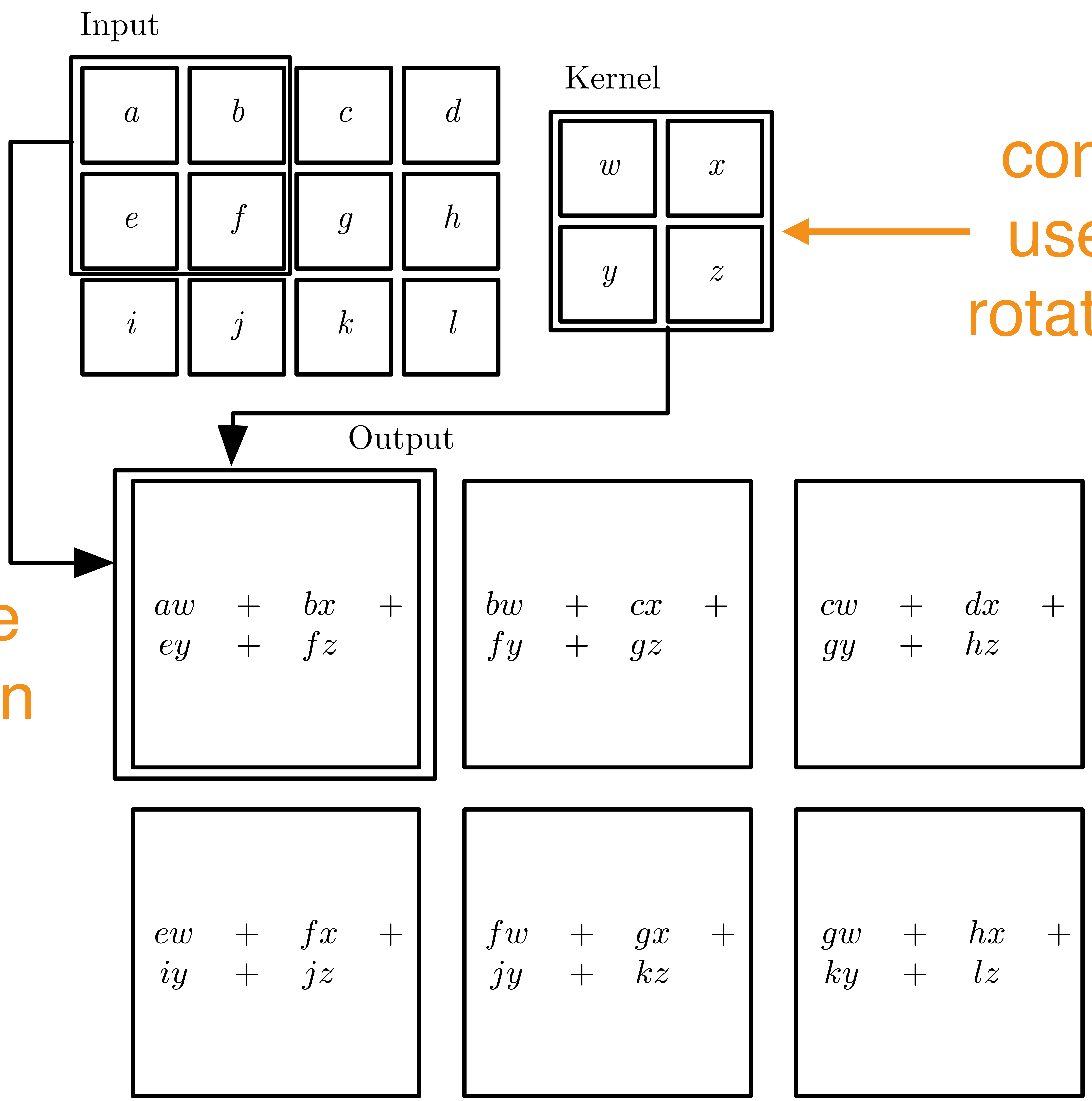
$$= \sum_{i,j} w_{-}[m-i, n-j]x[i,j]$$

related to the convolution

$$= (x * w_{-})[m,n]$$

flipping

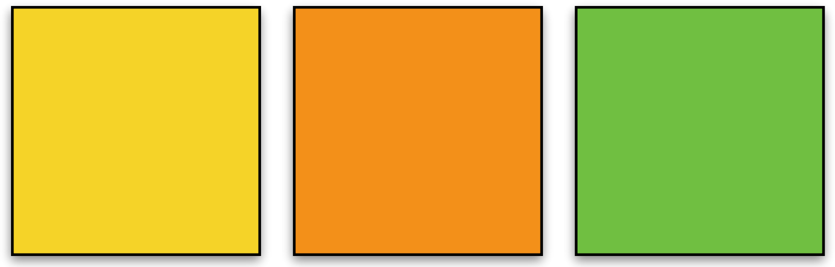
The Convolution Operation



convolution uses a 180 rotated kernel

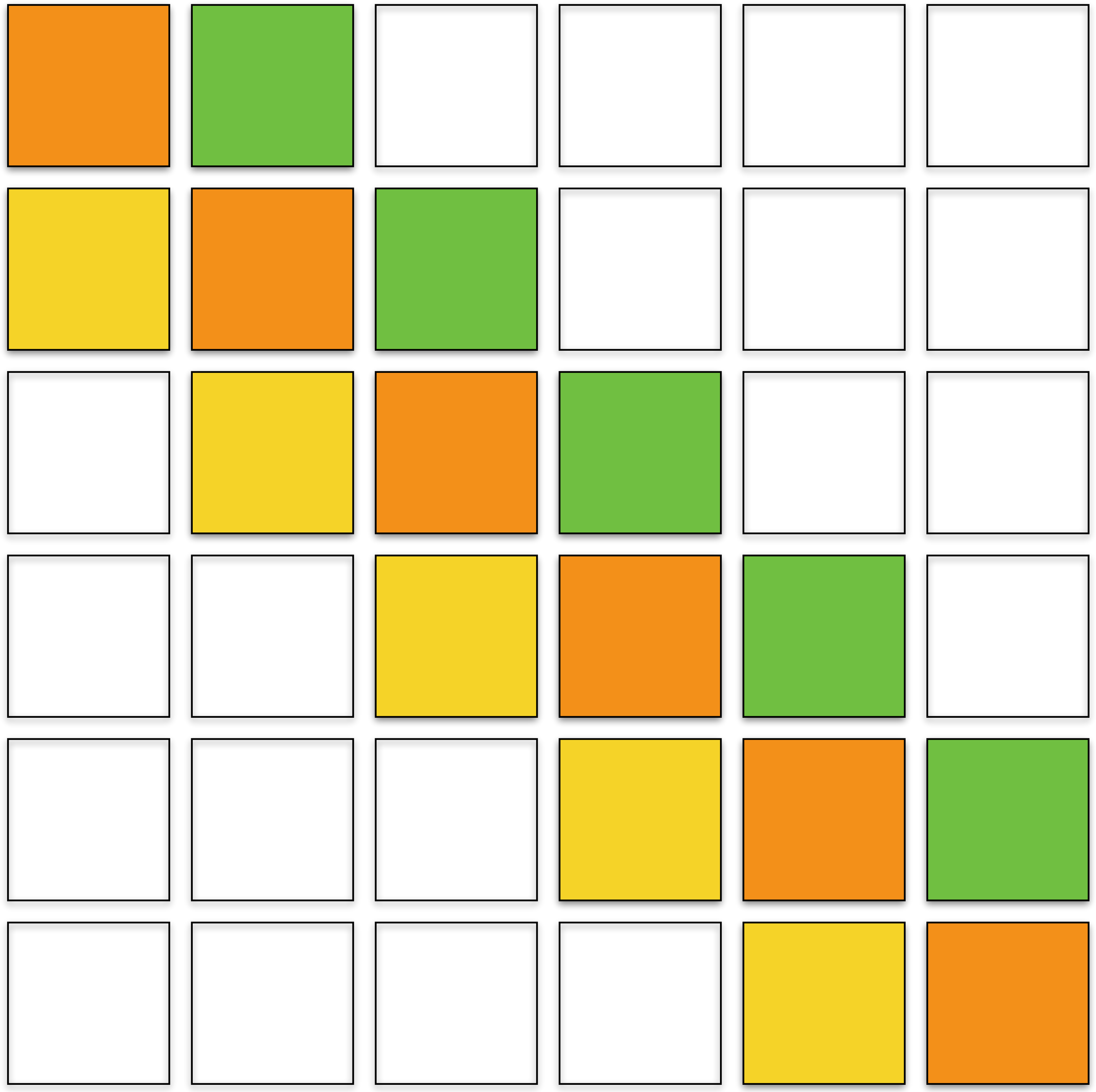
this is the correlation

Toeplitz Matrix



kernel

$$s[n] = (x * w)[n]$$
$$= \sum_i A[n, i]x[i]$$



Toeplitz matrix

Motivation

- Convolutions leverage four ideas

1. Sparse interaction

2. Parameter sharing

3. Equivariant representations

4. It can equally handle inputs of different sizes*

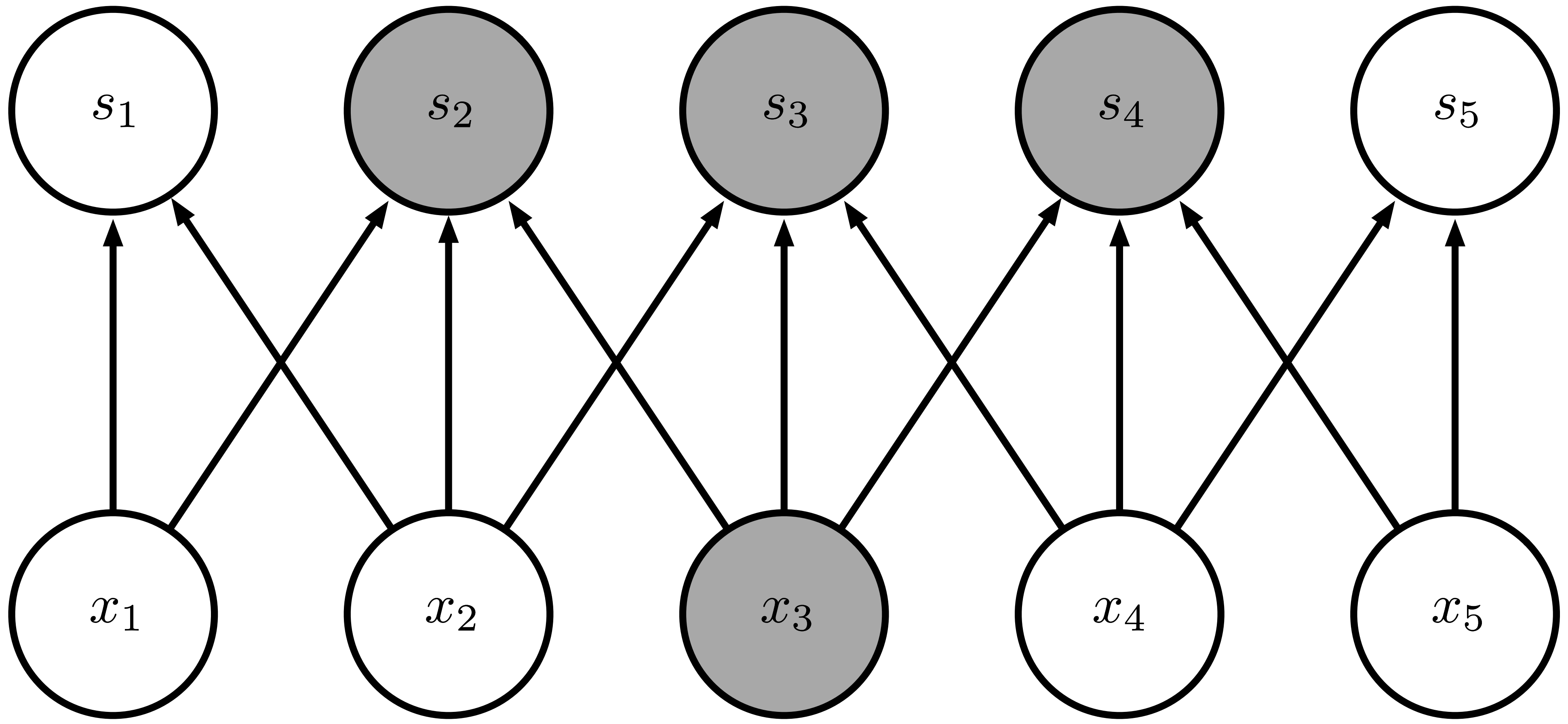
*pay attention to the boundary conditions!

Sparse Interaction

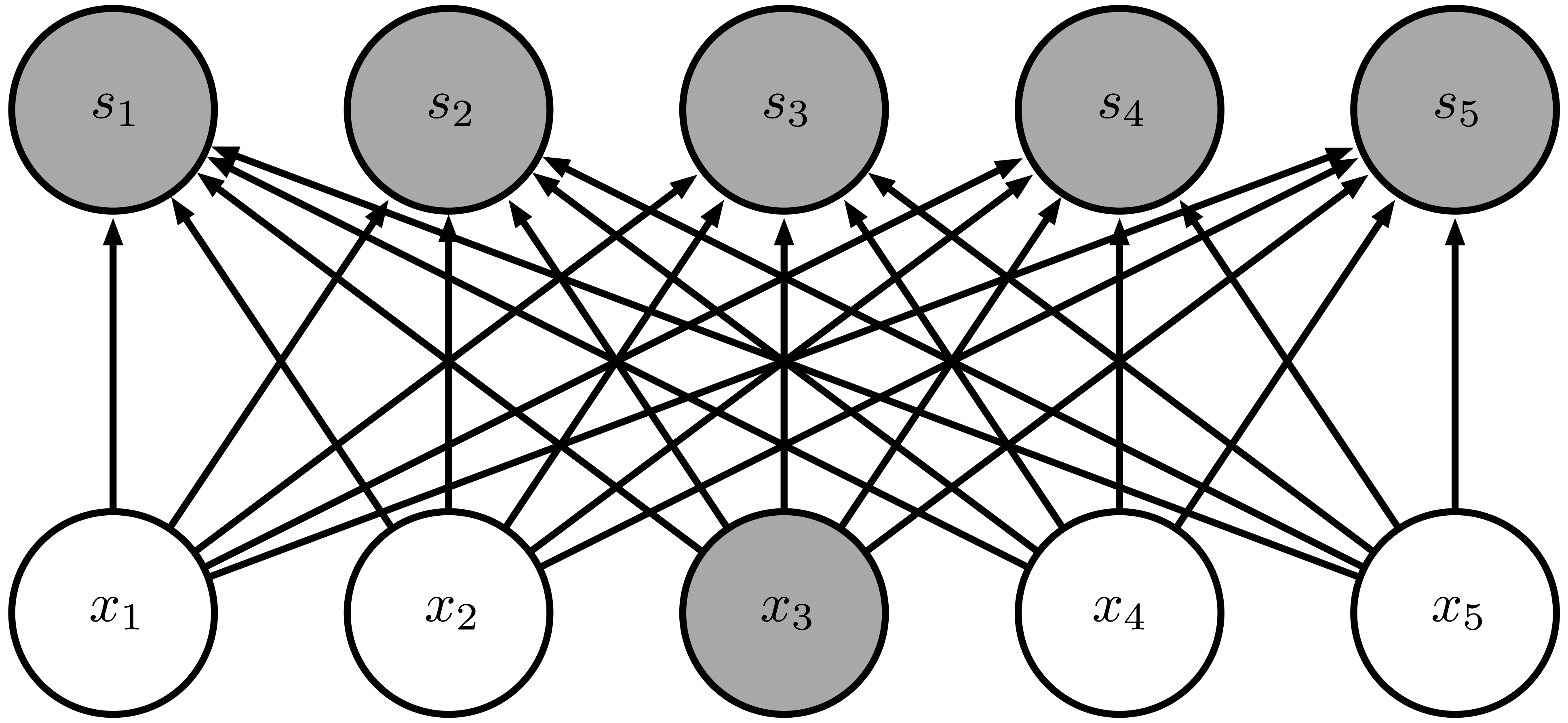
- In a fully connected layer every output can potentially depend on all inputs
- In a convolutional layer an output depends only on a small neighborhood of inputs (when the kernel is smaller than the input)
- The number of calculations is limited by the kernel size

Sparse Interaction

Sparse connections due to small convolution kernel

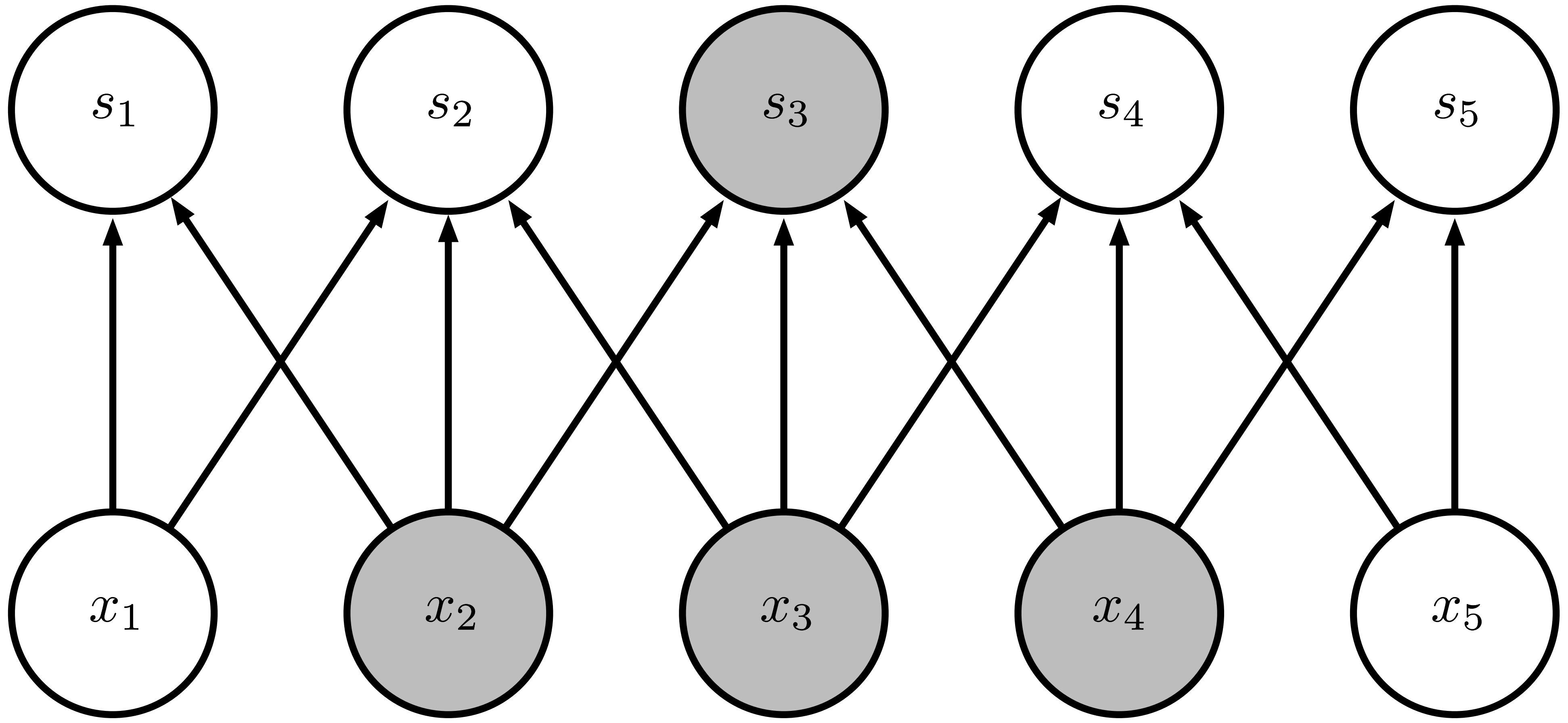


Dense connections

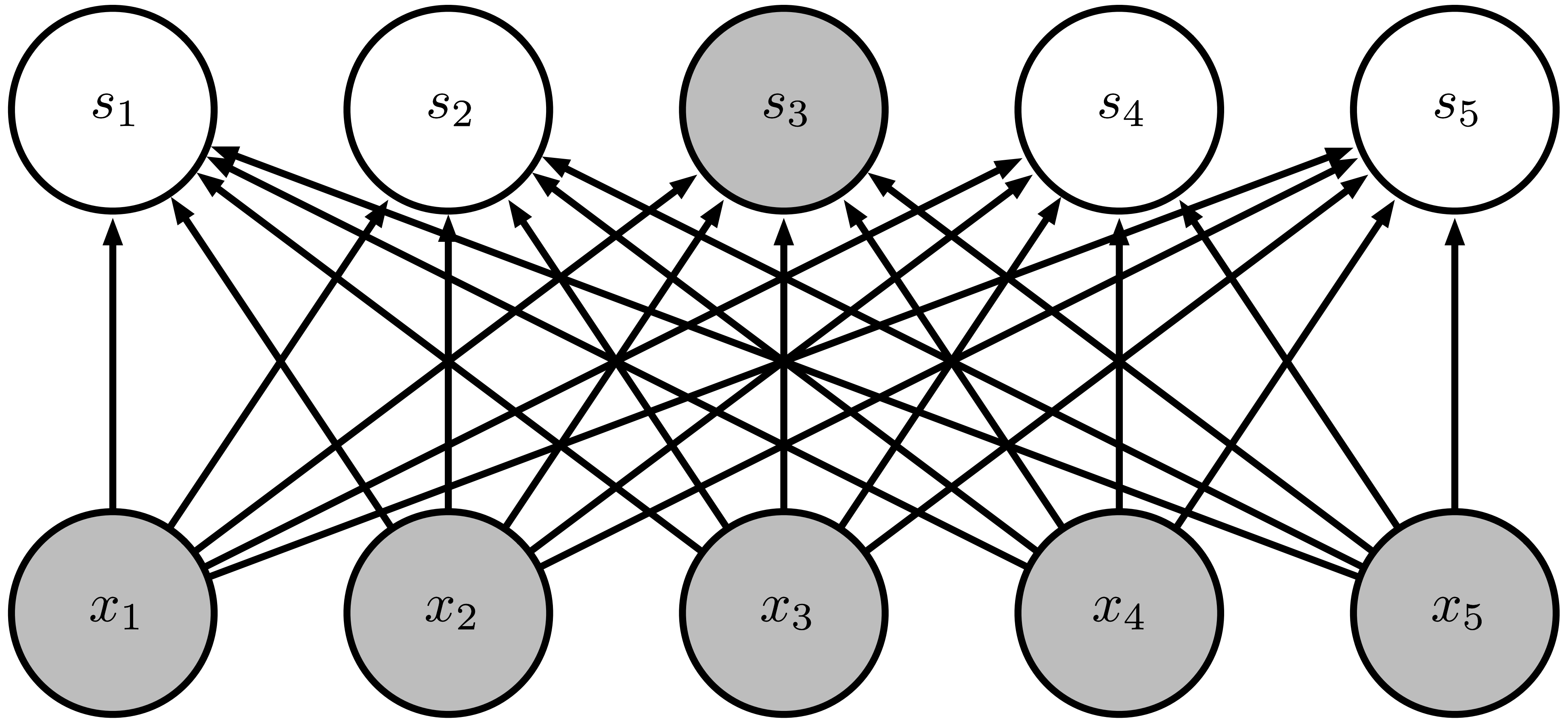


Sparse Interaction

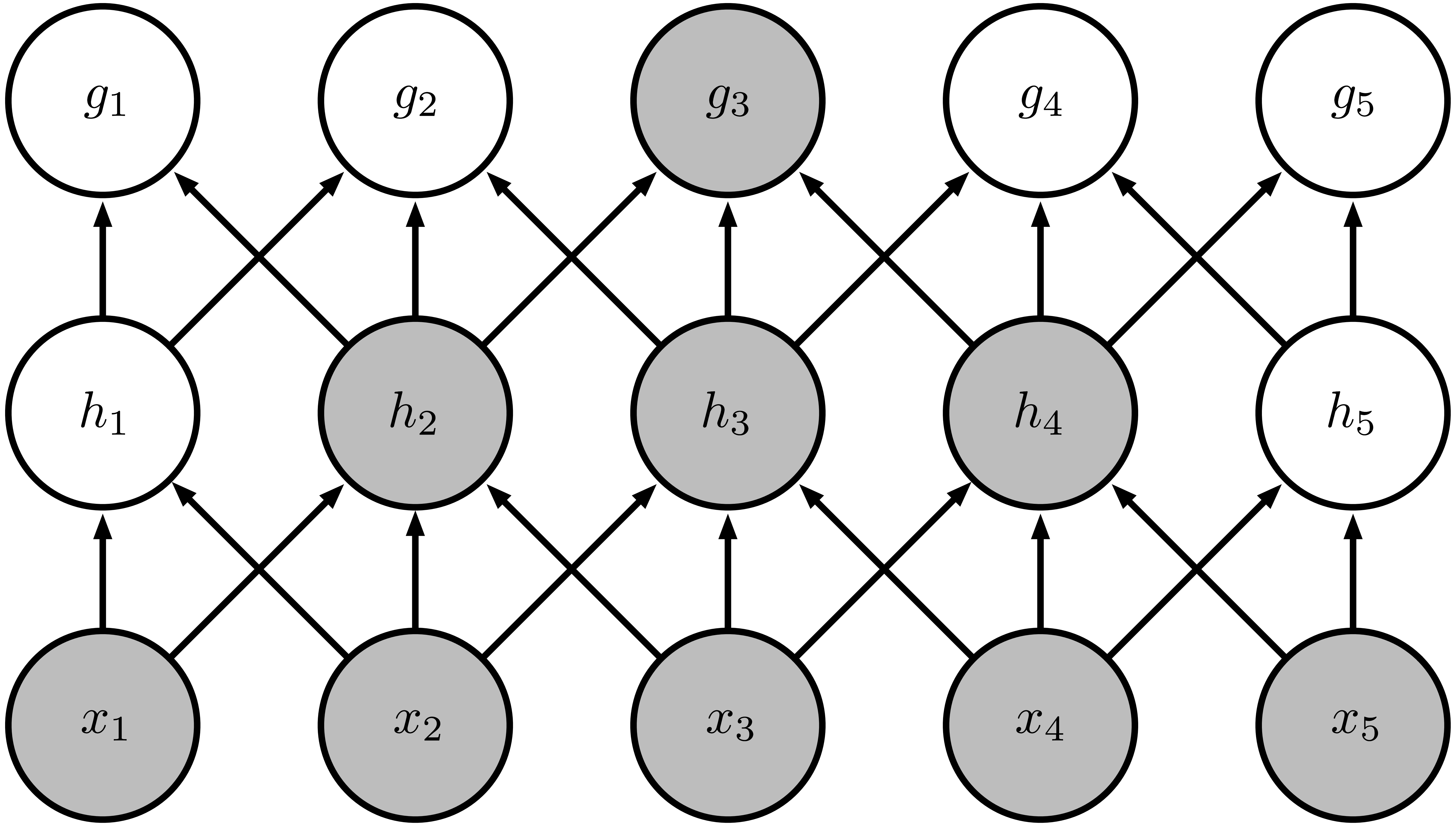
Sparse connections due to small convolution kernel



Dense connections



Receptive Field



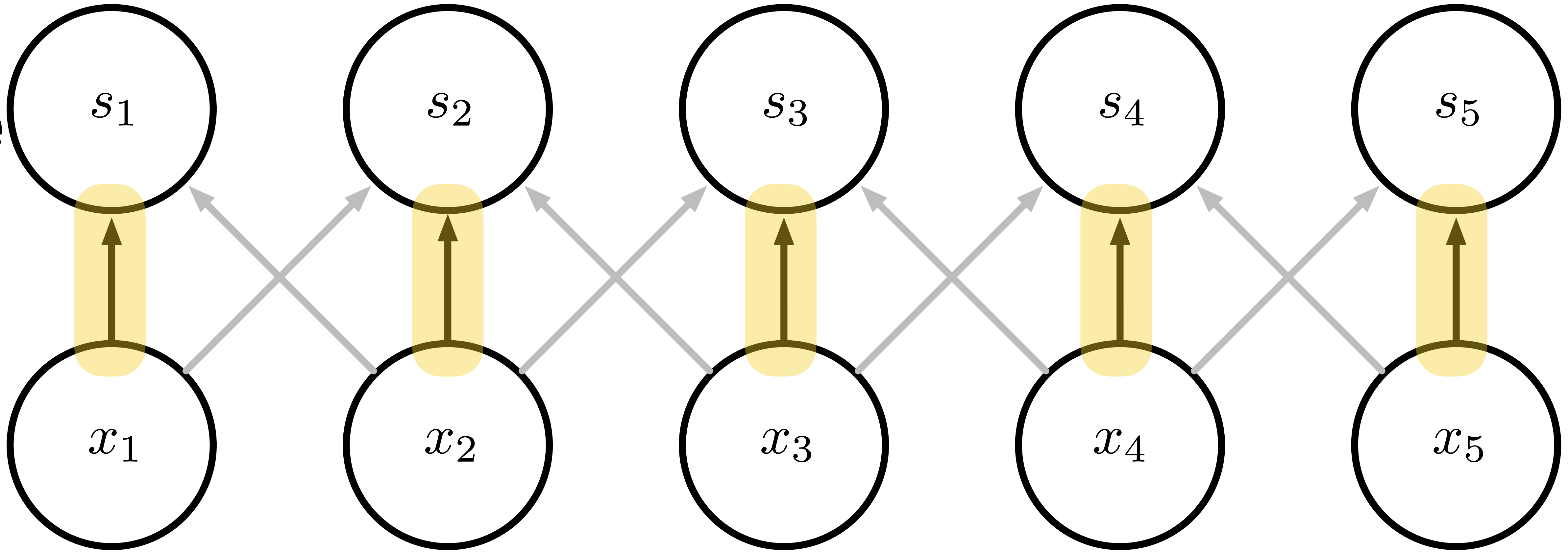
A **receptive field** is defined as all the units in a certain layer that affect a later unit

Parameter Sharing

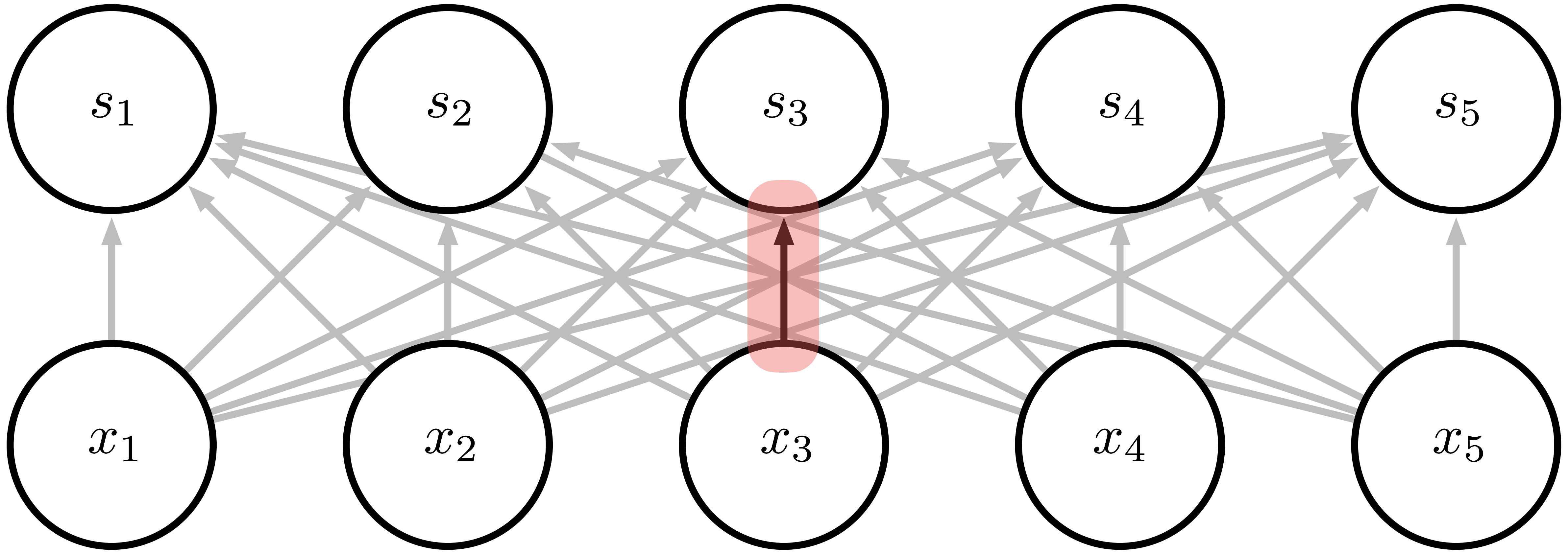
- Each output depends on the same parameters (the kernel weights)
- Exceptions are the boundaries (here padding and boundary assumptions are important)
- Fewer parameters means less storage

Parameter Sharing

Convolution shares the same parameters across all spatial locations



Traditional matrix multiplication does not share any parameters

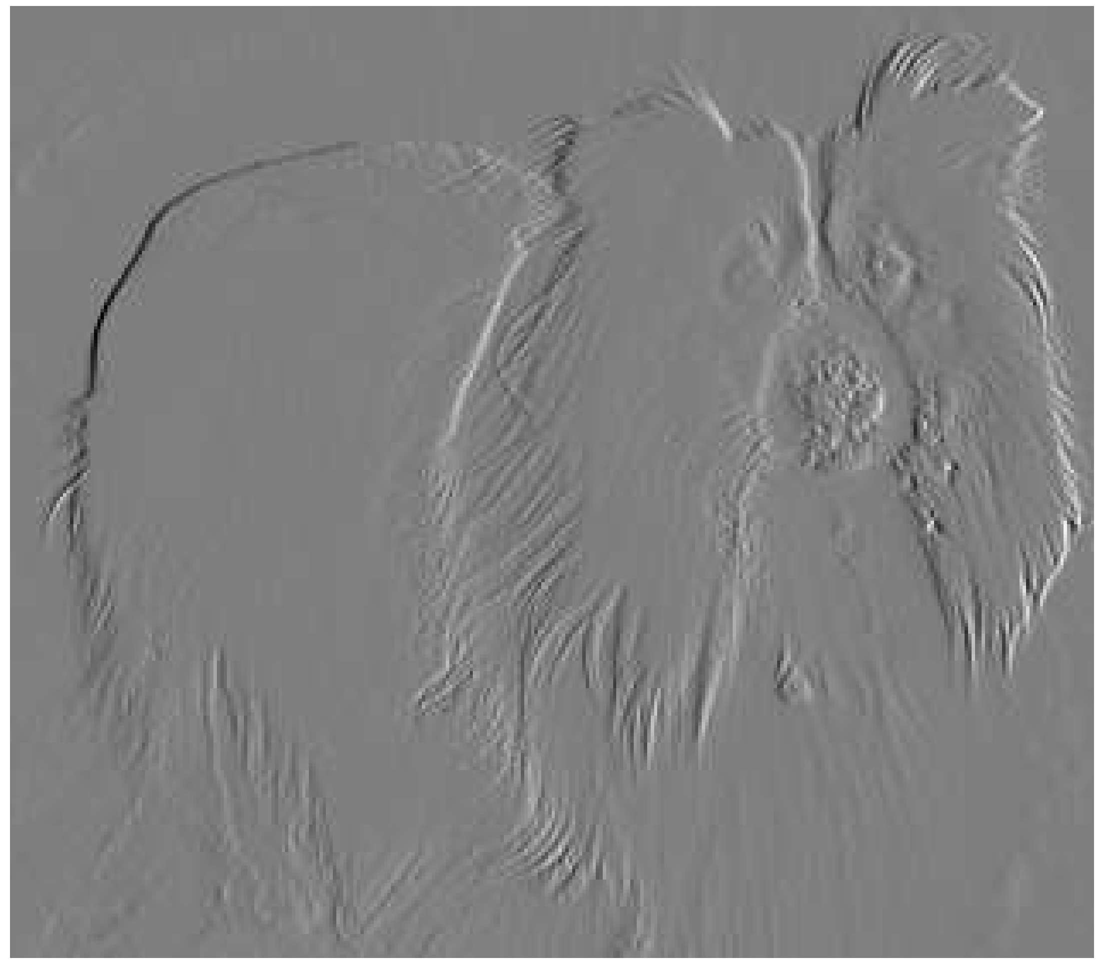


Example

Input

1	-1

Kernel



Output

Equivariant Representations

- An important property of the convolution is that it is shift-invariant (invariant to translation)
- In many signals this is the correct assumption: the same object might appear anywhere in the image and we should have the same (feature) response at any of these locations
- This concept is captured by **equivariance**

Equivariant Representations

- A function **f** is **equivariant** to a function **g** if **f(g(x)) = g(f(x))**
- If **f** is the convolution and **g** the translation, then **f** is equivariant to **g**
- The convolution is not equivariant to rotation and scaling

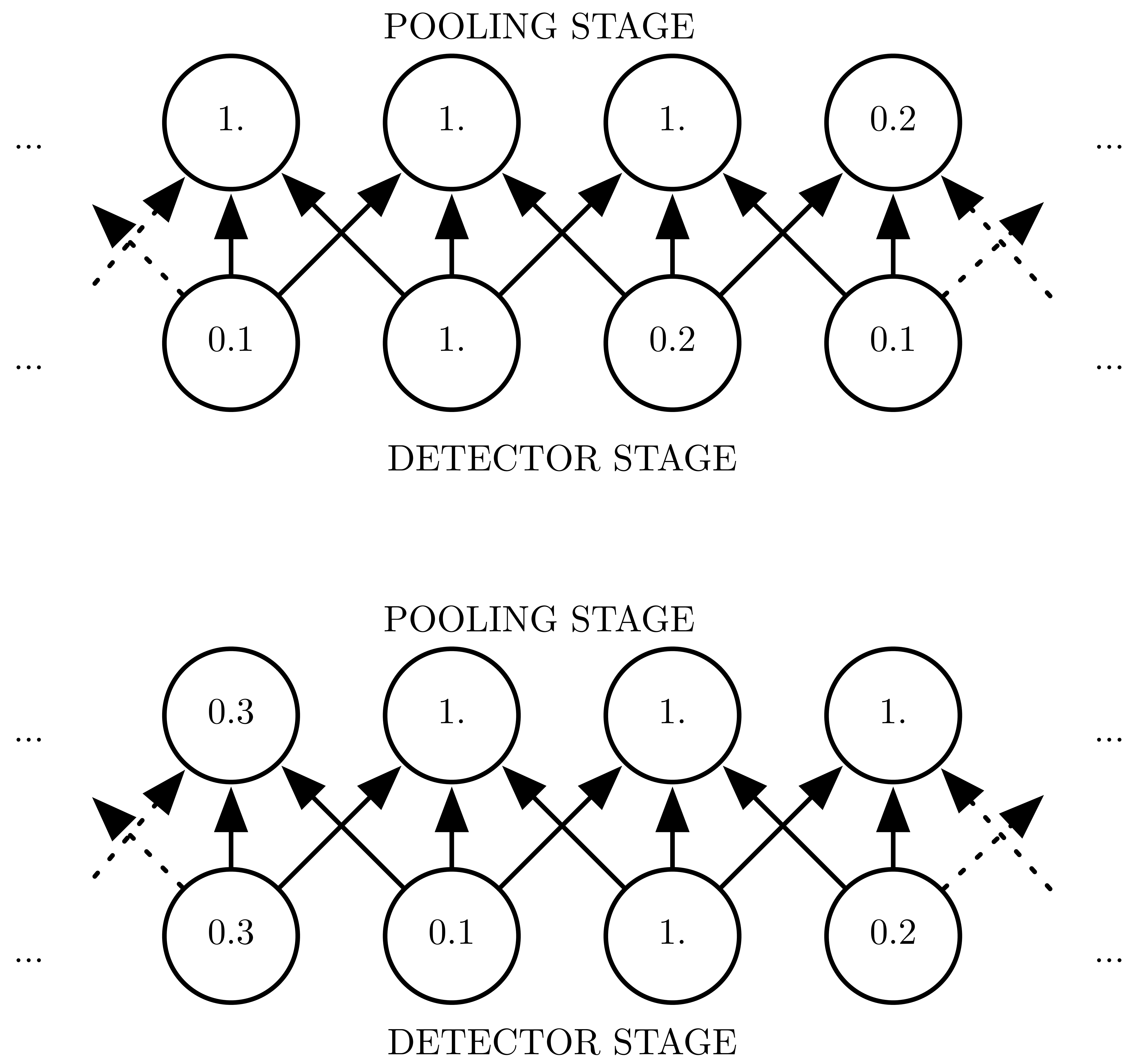
Pooling

- Typical layers of convolutional neural networks consist of three stages: 1) a convolutional layer, 2) a nonlinear activation function (e.g., ReLU) and 3) a **pooling** function
- 1) and 2) are also called a **detector**

Pooling

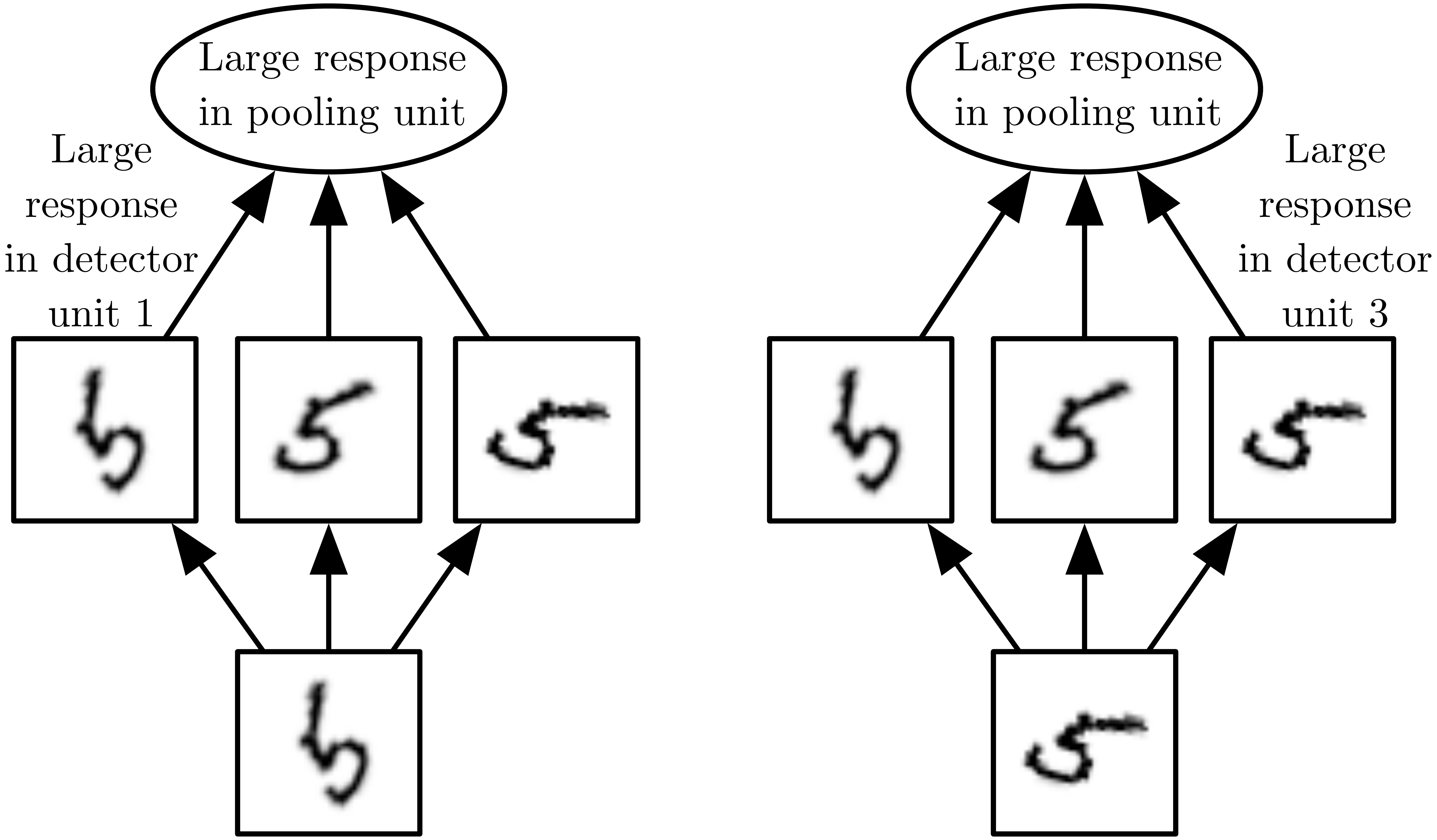
- Pooling performs a calculation on a sliding window
- Some pooling operations: the **max** within the window, the **average** of the window, the **L_2 norm** of the window, the **weighted average** of the window
- Pooling gives a local (to a window) invariance to translation

Max Pooling


POOLING STAGE
1.
1.
1.
0.2
0.1
1.
0.2
0.1
DETECTOR STAGE
POOLING STAGE
0.3
1.
1.
1.
0.3
0.1
1.
0.2
DETECTOR STAGE

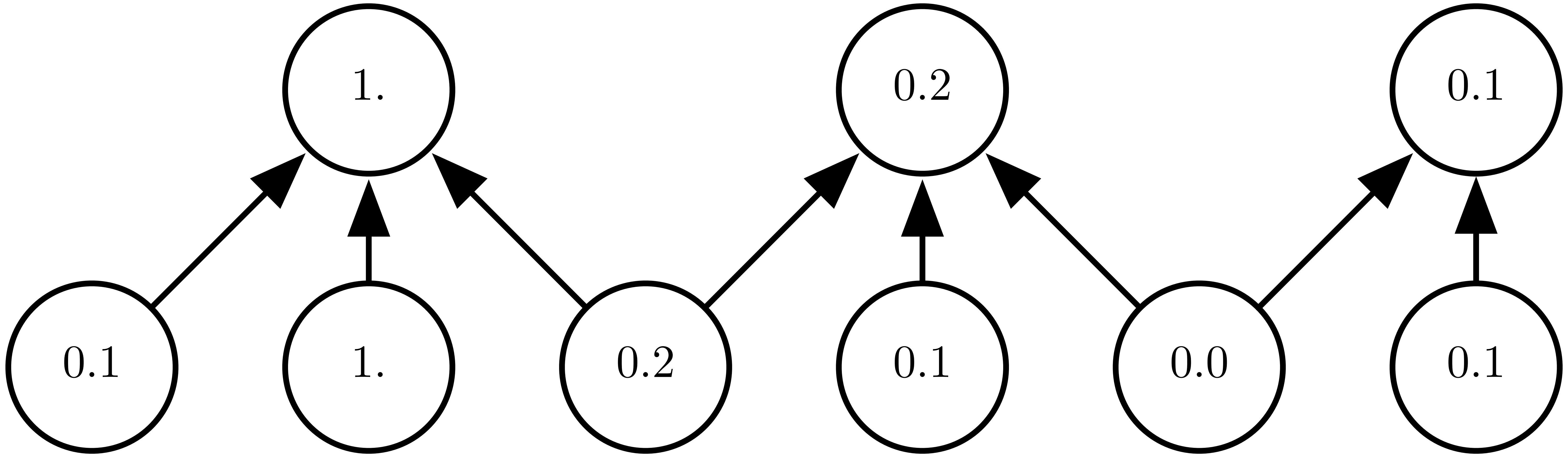
Learning Invariances

- Pooling applied across channels can learn invariances to transformations



Pooling

- Because pooling summarizes the responses over a neighborhood, it is sufficient to under sample the pooling output (e.g., by the size of the window)

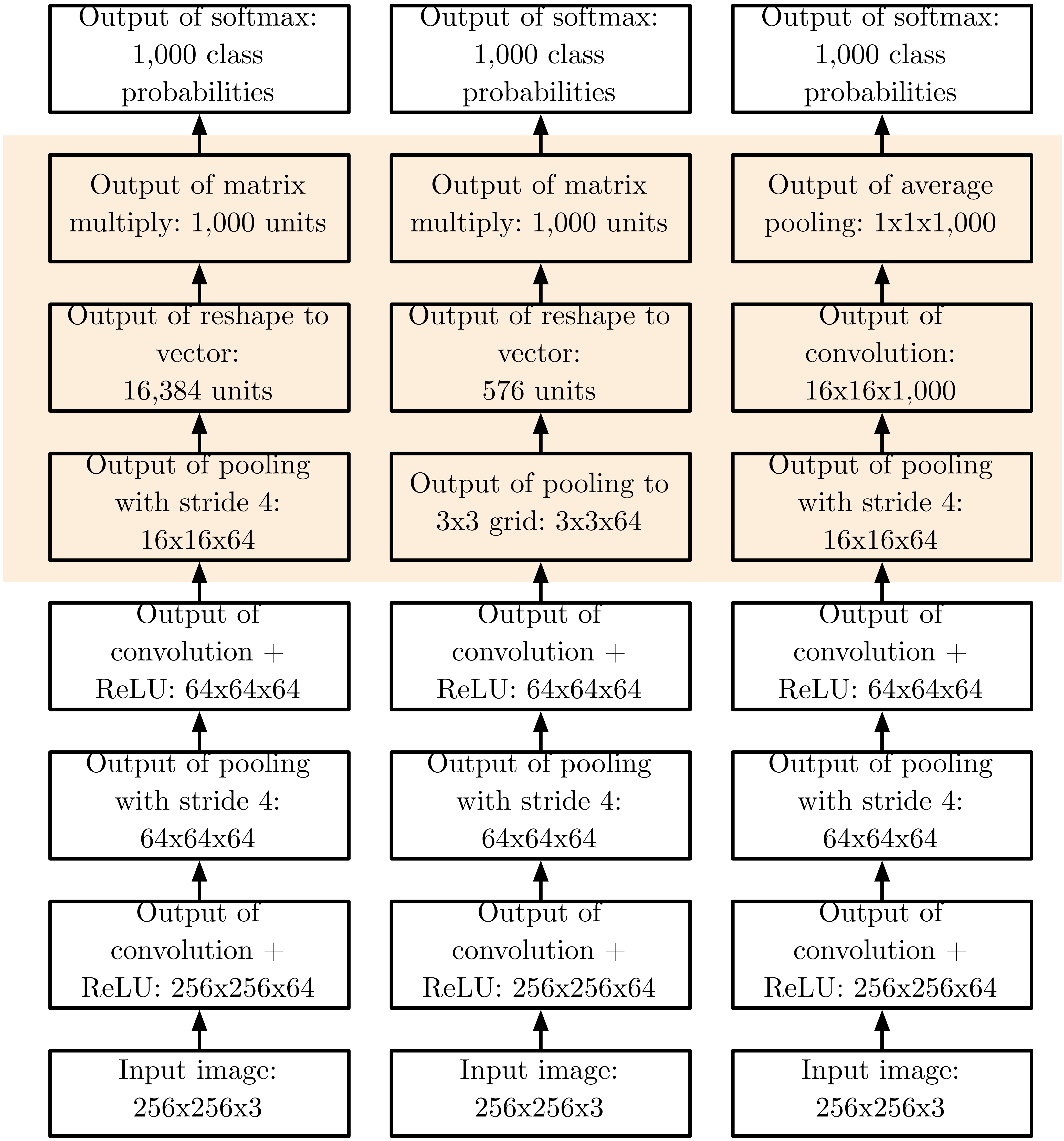


Pooling

- In many tasks, this is useful to reduce the initial dimension of the input and eventually reach the desired output size

- With an adaptive neighborhood size it can produce the same output size regardless of the input size

Classification Architectures



Variants

- Input data is typically a 4D tensor: 2 dimensions for the spatial domain, 1 dimension for the channels (e.g., colors), and 1 dimension for the batch

- The convolution (correlation) applies to the spatial domain only

$$\underbrace{Z_{i,j,k}}_{\text{output}} = \sum_{l,m,n} \underbrace{V_{l,j+m,k+n}}_{\text{input}} \underbrace{K_{i,l,m,n}}_{\text{kernel}}$$

Stride

- We can also skip outputs by defining a **stride** s larger than 1

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j\times s+m,k\times s+n} K_{i,l,m,n}$$

Padding

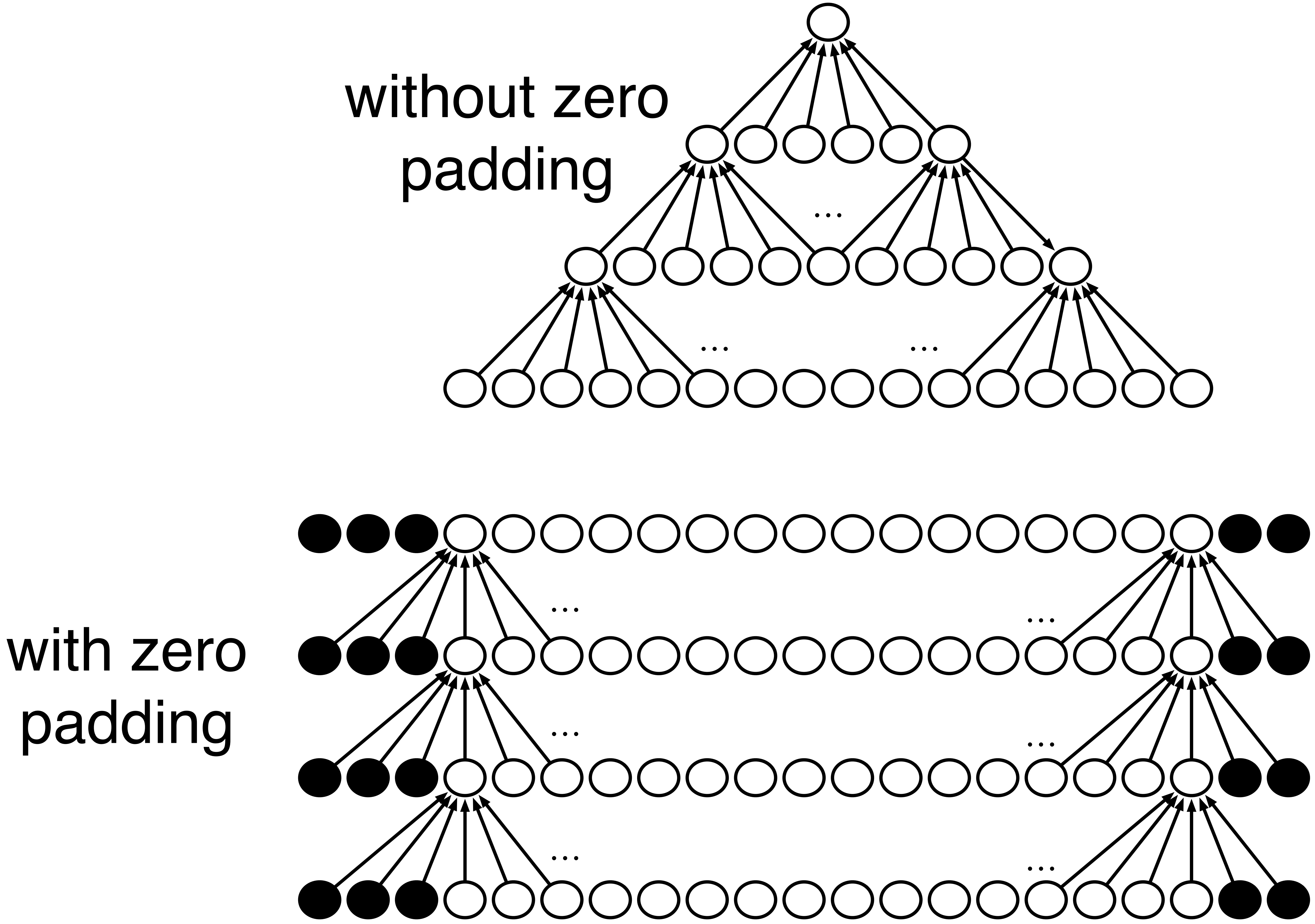
- The output of a convolution is valid as long as the summation uses available values

- In a convolution the valid output size is equal to:
  the input size - the size of the kernel + 1

- Unless we make boundary assumptions, a convolution will lead to a progressive shrinking of the input

- **Padding** is the assumption that outside the given domain the input takes some fixed values (e.g., zero)

Padding

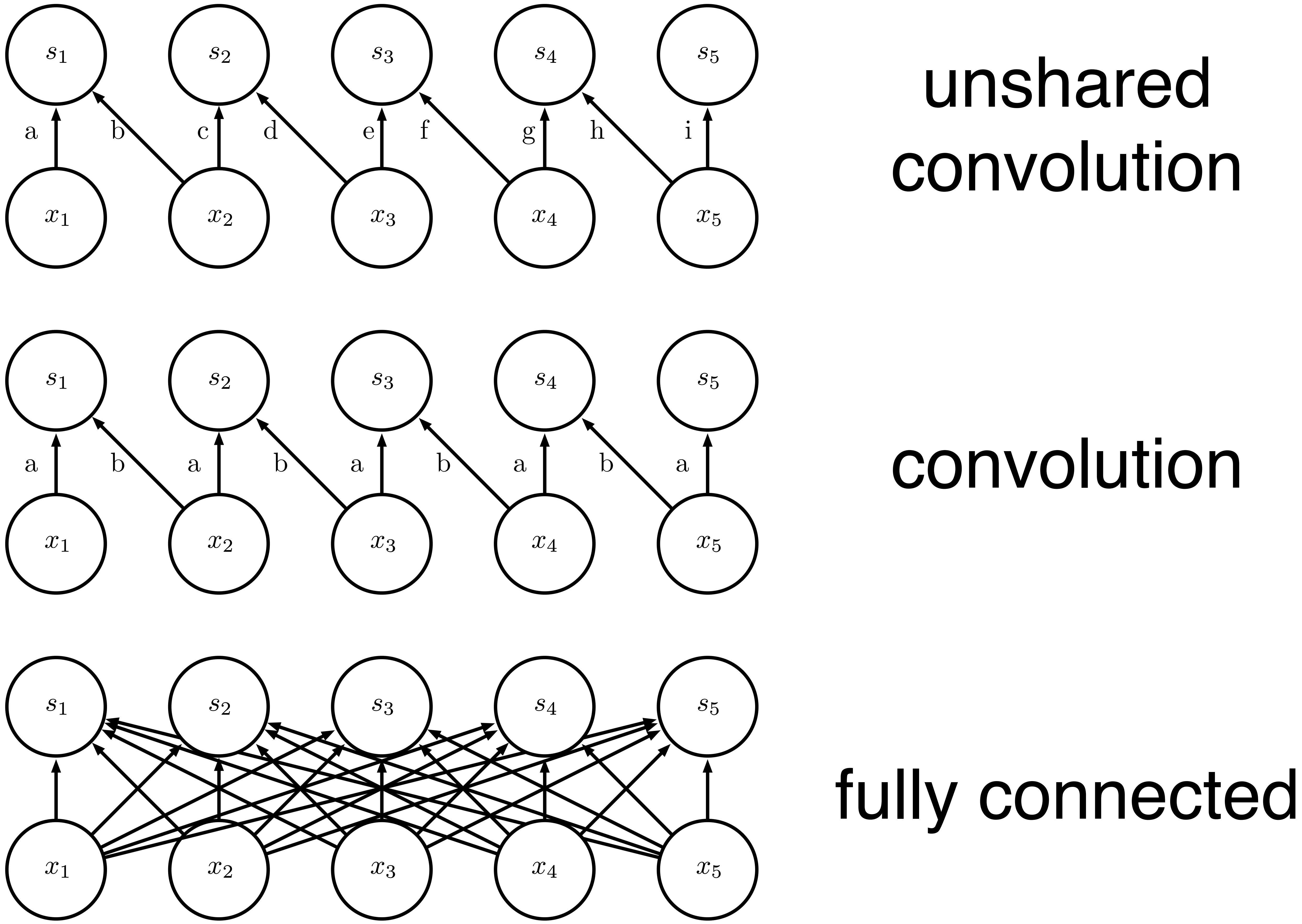


Unshared Convolution

- Unshared convolutions use kernels whose weights change at every location, but only apply to a small neighborhood

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m,k+n} w_{i,j,k,l,m,n}$$

Unshared Convolution



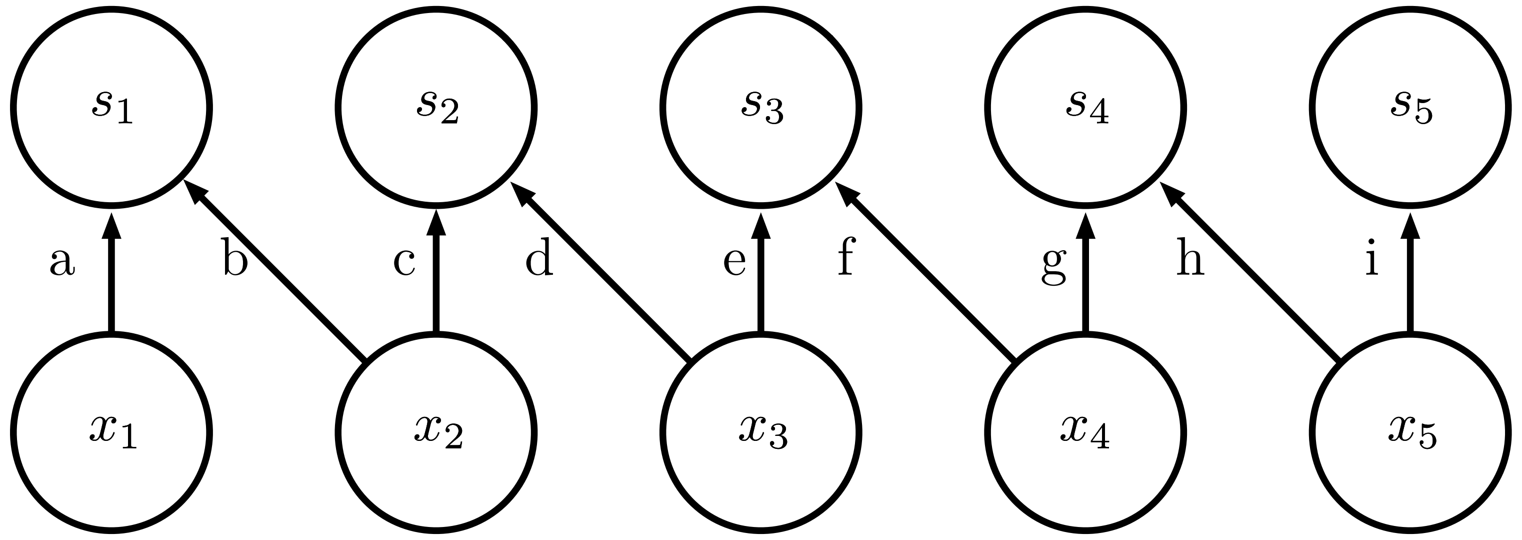
Tiled Convolution

- A tradeoff between convolutions and locally connected layers
- The kernels repeat on a spatial lattice (locally they may be different)

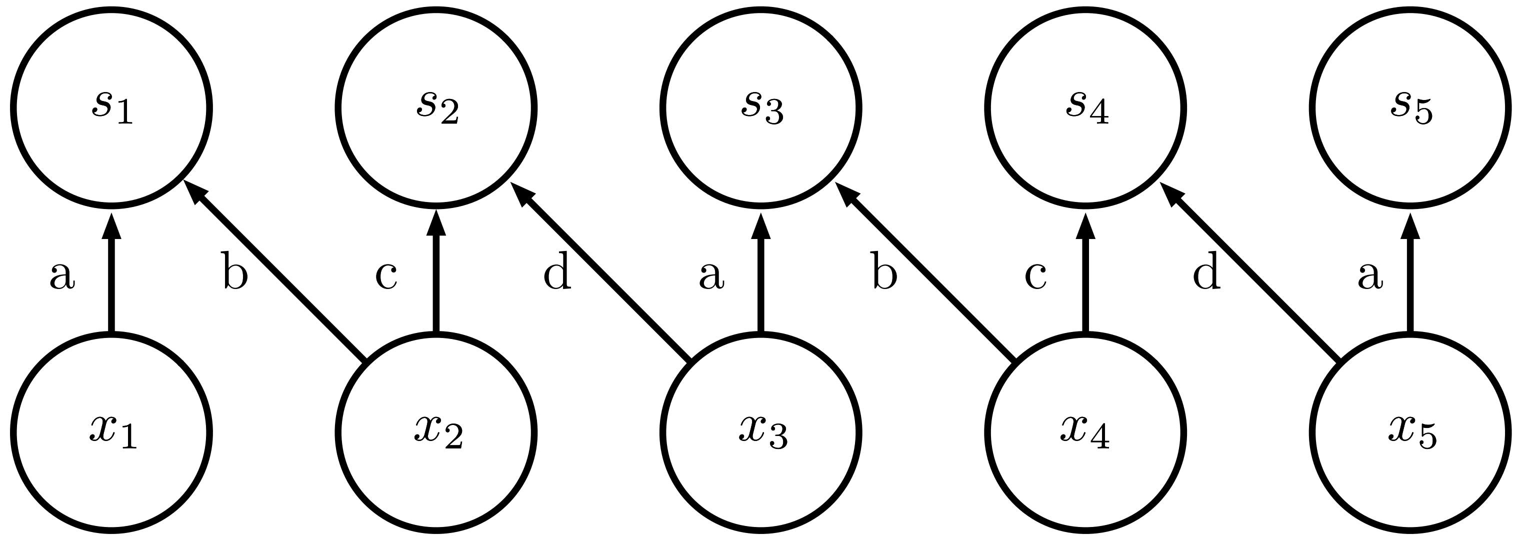
$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m,k+n} K_{i,l,m,n,j\%t,i\%t}$$

where % denotes the modulo operator

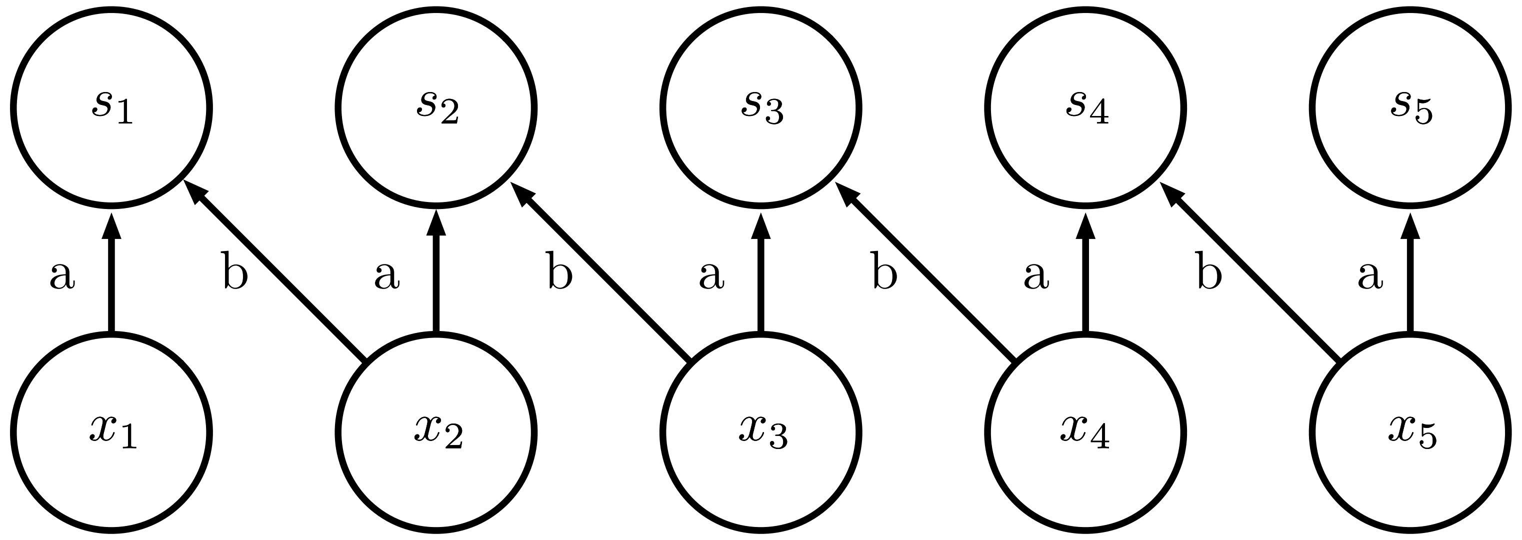
Tiled Convolution



local connection
(no sharing)



tiled convolution
(cycle between
groups of shared
parameters)



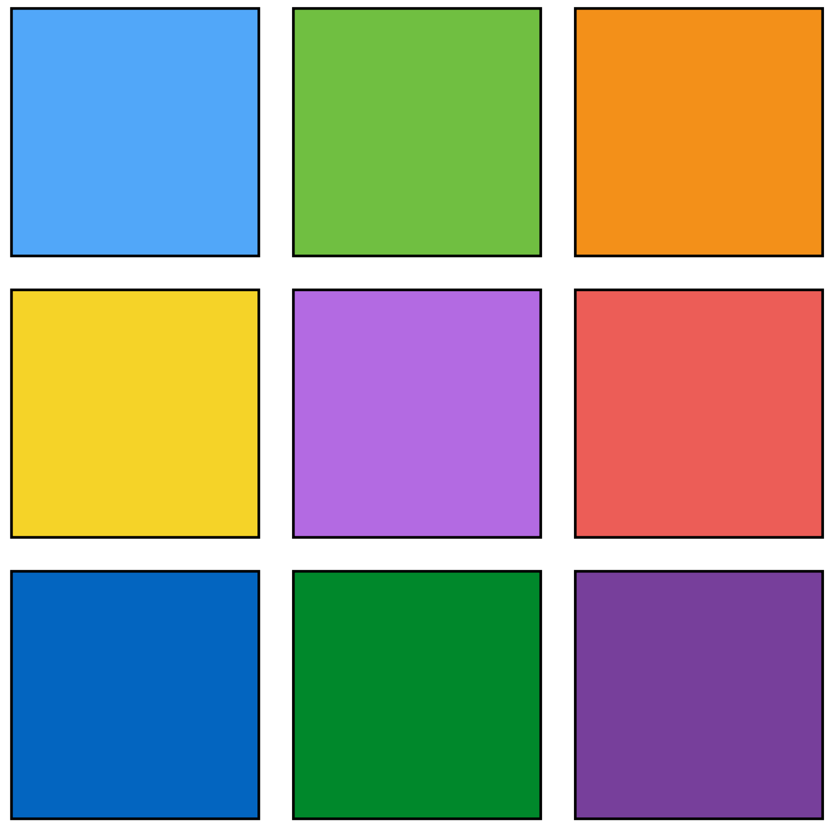
convolution
(one group shared
everywhere)

Dilated Convolutions

- They expand the kernel size by introducing zeros in between the non-zero components

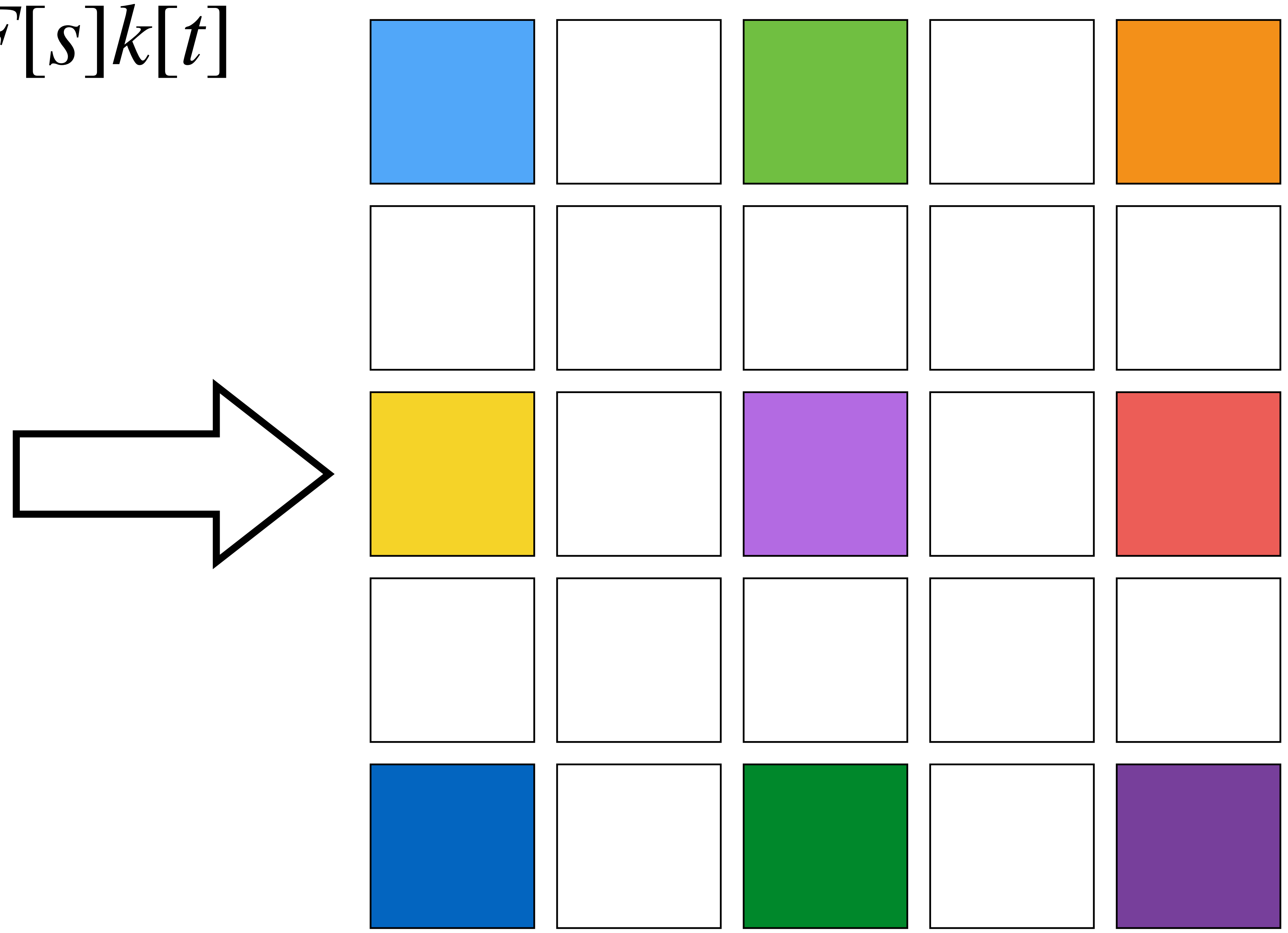
- Also called Atrous Convolution

$$(F \circledast k)[p] = \sum_{s+t=p} F[s]k[t]$$



standard convolution

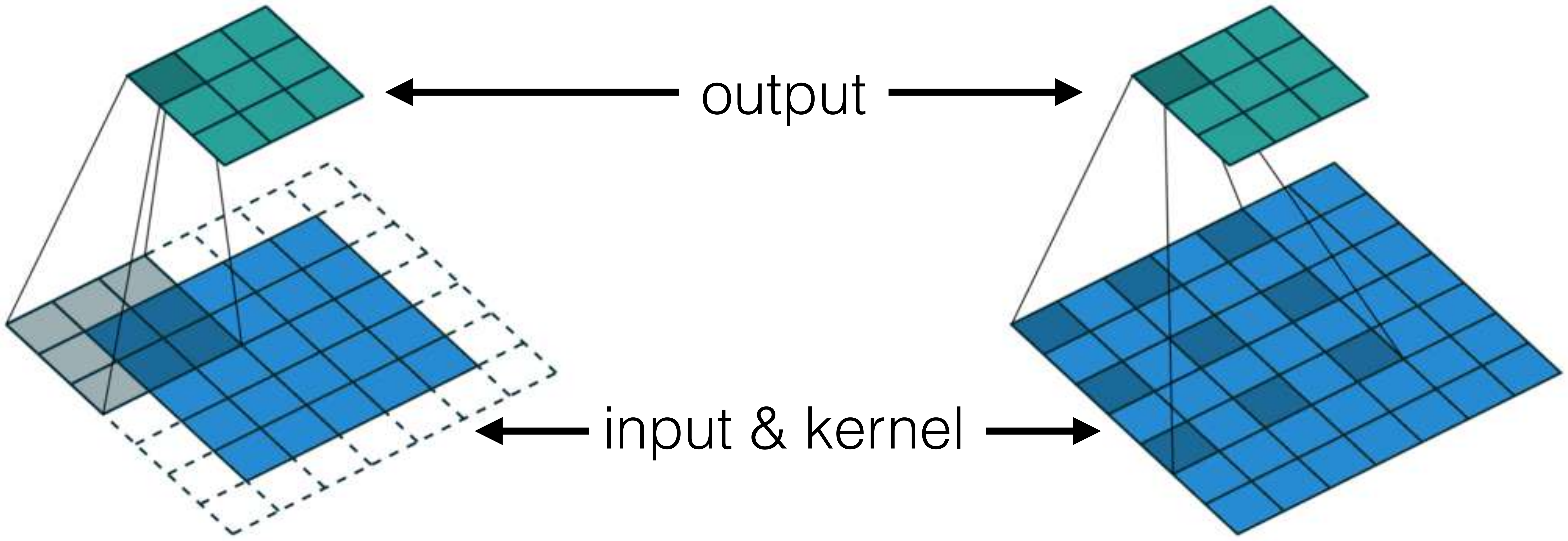
$$(F \circledast_q k)[p] = \sum_{s+qt=p} F[s]k[t]$$



2-dilated convolution

Yu, F and Koltun, V. Multi-scale context aggregation by dilated convolutions. arXiv 2015
Chen LC, et al. Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv 2014

Dilated Convolutions



standard convolution

2-dilated convolution

animations from https://iq.opengenus.org/dilated-convolution/

Deformable Convolutions

- Convolutions cannot adapt to geometric transformations of the input
- Deformable convolutions learn also a shift for each entry in the kernel

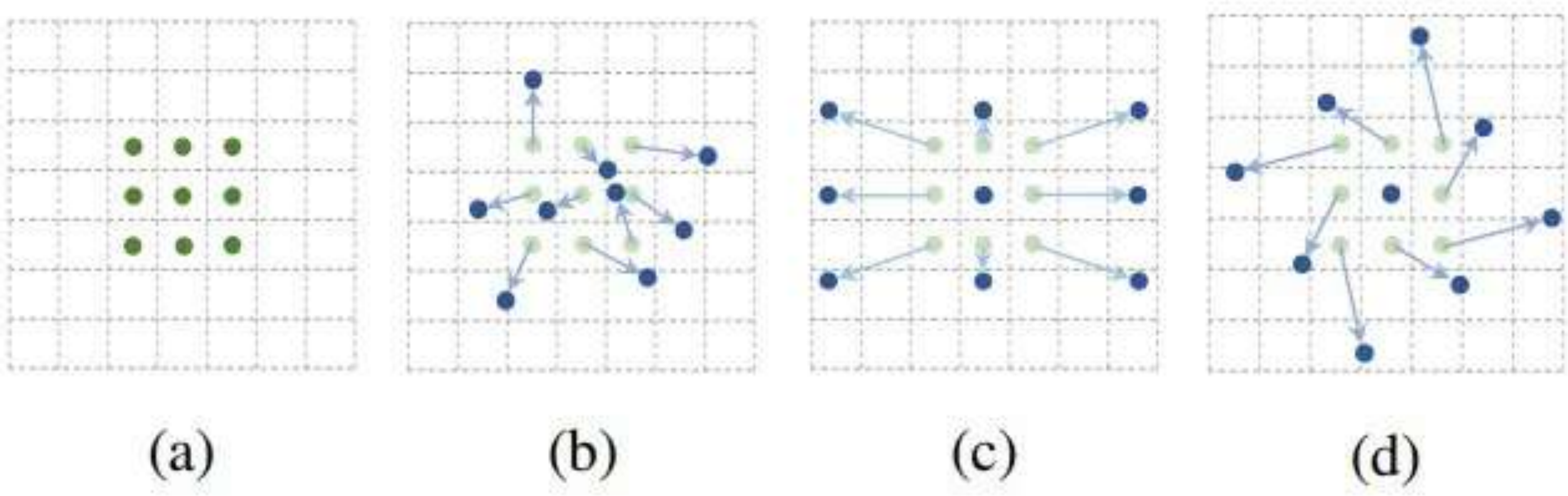


illustration from https://towardsdatascience.com/deformable-convolutions-demystified-2a77498699e8

…and much More!

- https://paperswithcode.com/methods/category/convolutions

Structured Outputs

- Convolutional networks can output a high-dimensional structured object (e.g., a tensor)
- For example, one could output a 3D tensor with the probability of a given set of classes at each pixel
- This could be used for segmentation or pixel-wise labeling

Data Types

- Input data can be in different formats
- 1D: Audio waveforms (single channel) and skeleton animation data/motion (multi-channel)
- 2D: Audio data preprocessed via Fourier (single channel), color image data (multi-channel)
- 3D: Volumetric data such as CT scans (single channel), color video data (multi-channel)

Data Types

- Convolutional networks allow dealing with images of different sizes
- Also, we might design networks whose output is size-varying (the loss function must be able to handle it) — e.g., per pixel labeling

Efficient Implementations

- Convolutions can be efficiently implemented via parallel computing (GPU)
- One fast implementation (when kernels are large) is via the FFT (the convolution is a dot product in Fourier space)
- Another fast implementation is when a d-dimensional kernel is separable (it can be written as the outer product of d 1D kernels)

Examples of Prominent CNN Architectures

- AlexNet
- VGG
- ResNet
- UNet
- (EfficientNet)
- 3D conv architectures
- Graph CNNs