

Deep Learning (BEV033DLE)

Lecture 5

Convolutional Neural Networks

Czech Technical University in Prague

◆ Introduction, CNN for Classification

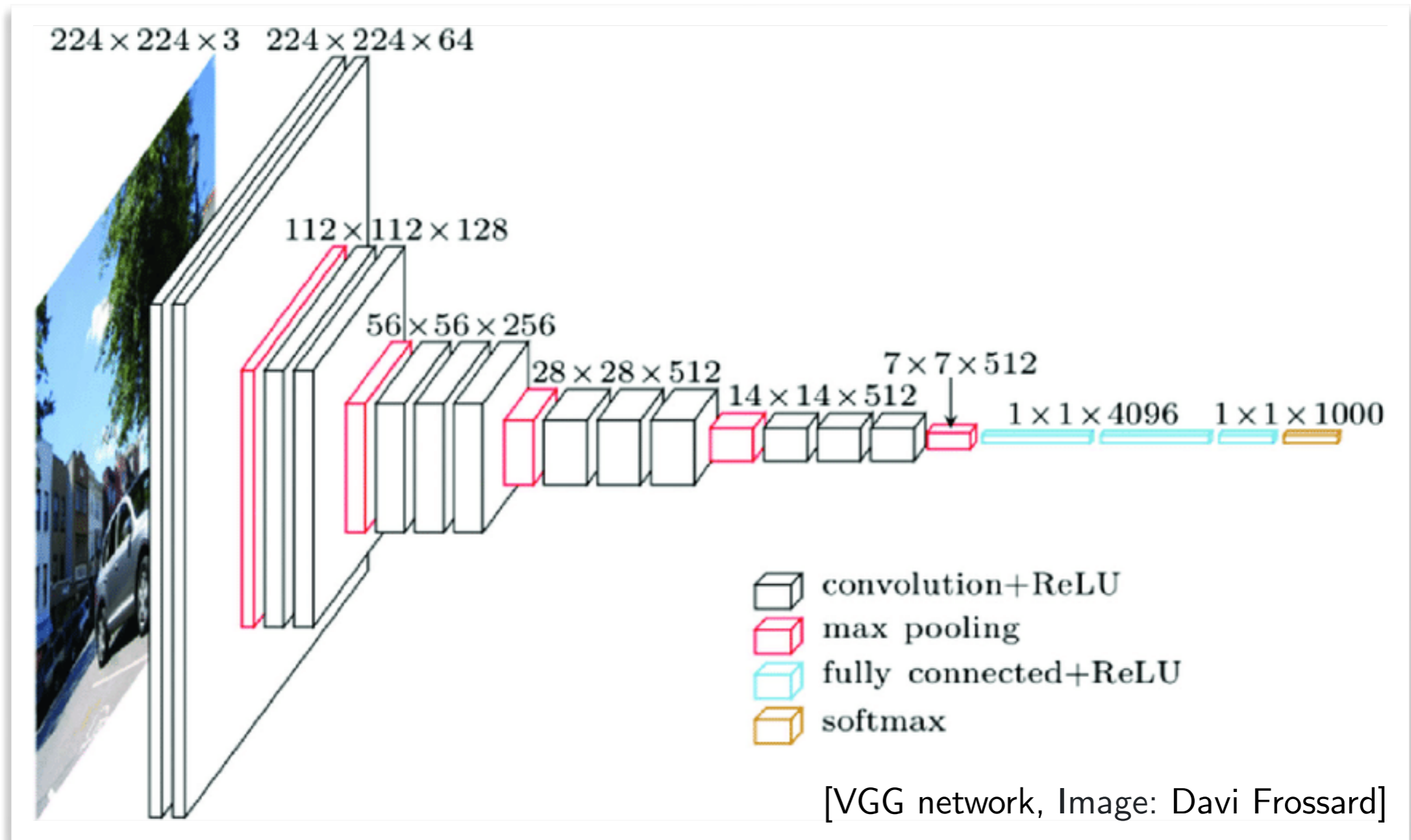
- Correlation filters, translation equivariance, convolution and cross-correlation
- Multi-channel, stride, 1x1
- Pooling, receptive field

◆ More CNNs

- Dilation, transposed

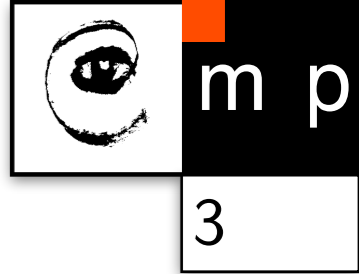
◆ Hierarchy of Parts, Visual Cortex

CNN for Classification



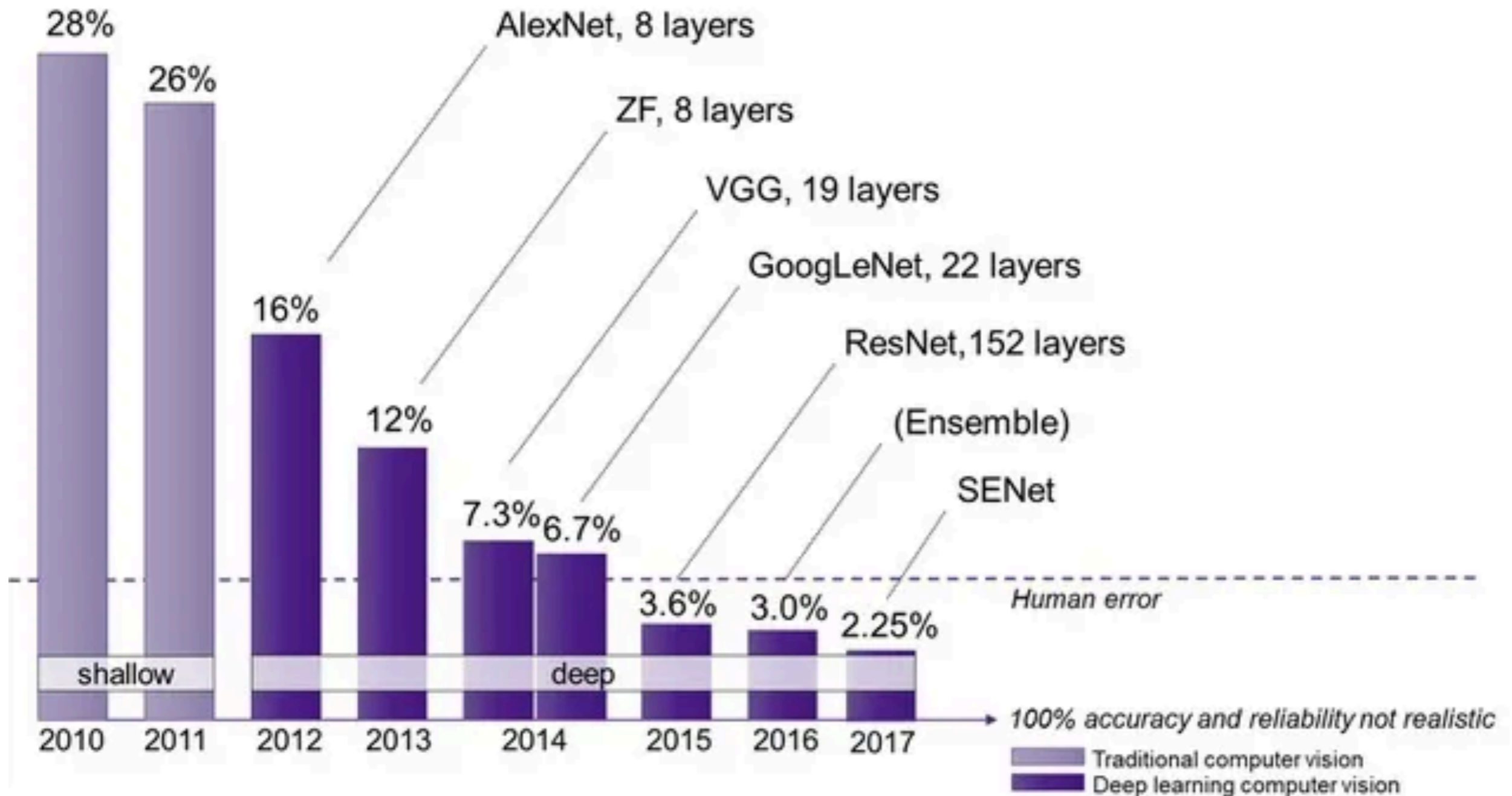
- ◆ We will go step-by-step through this diagram
- ◆ Understand its design principles
- ◆ Consider everything about convolutions in more detail

Big Picture: ImageNet Classification



- ◆ The ImageNet Large Scale Visual Recognition Challenge (2010): 1000 classes, 1.4M images
- ◆ Deep Learning Revolution

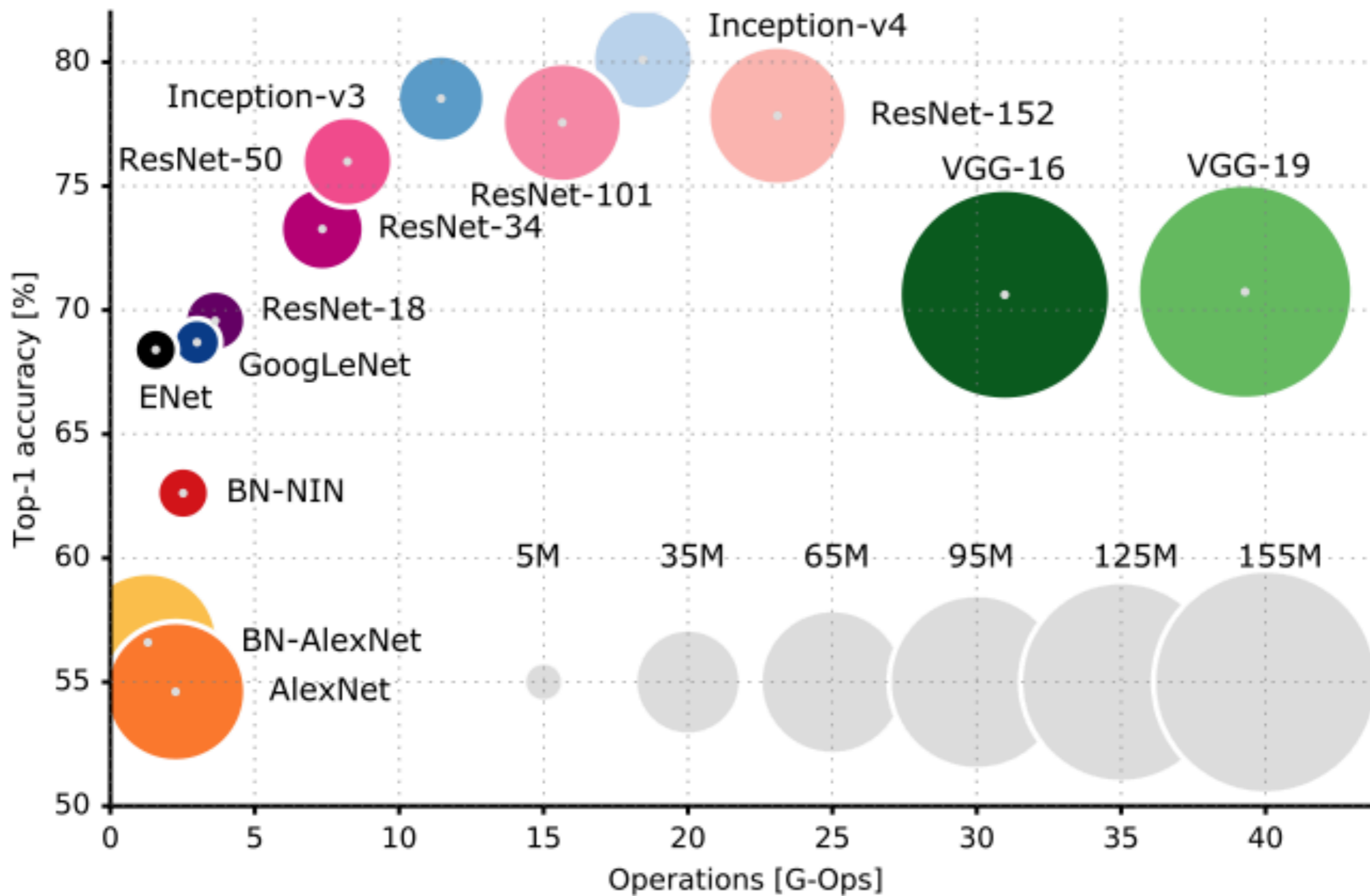
Top-5 accuracy



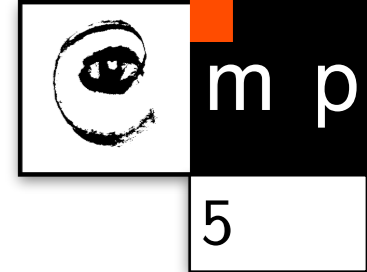
Big Picture: ImageNet Classification



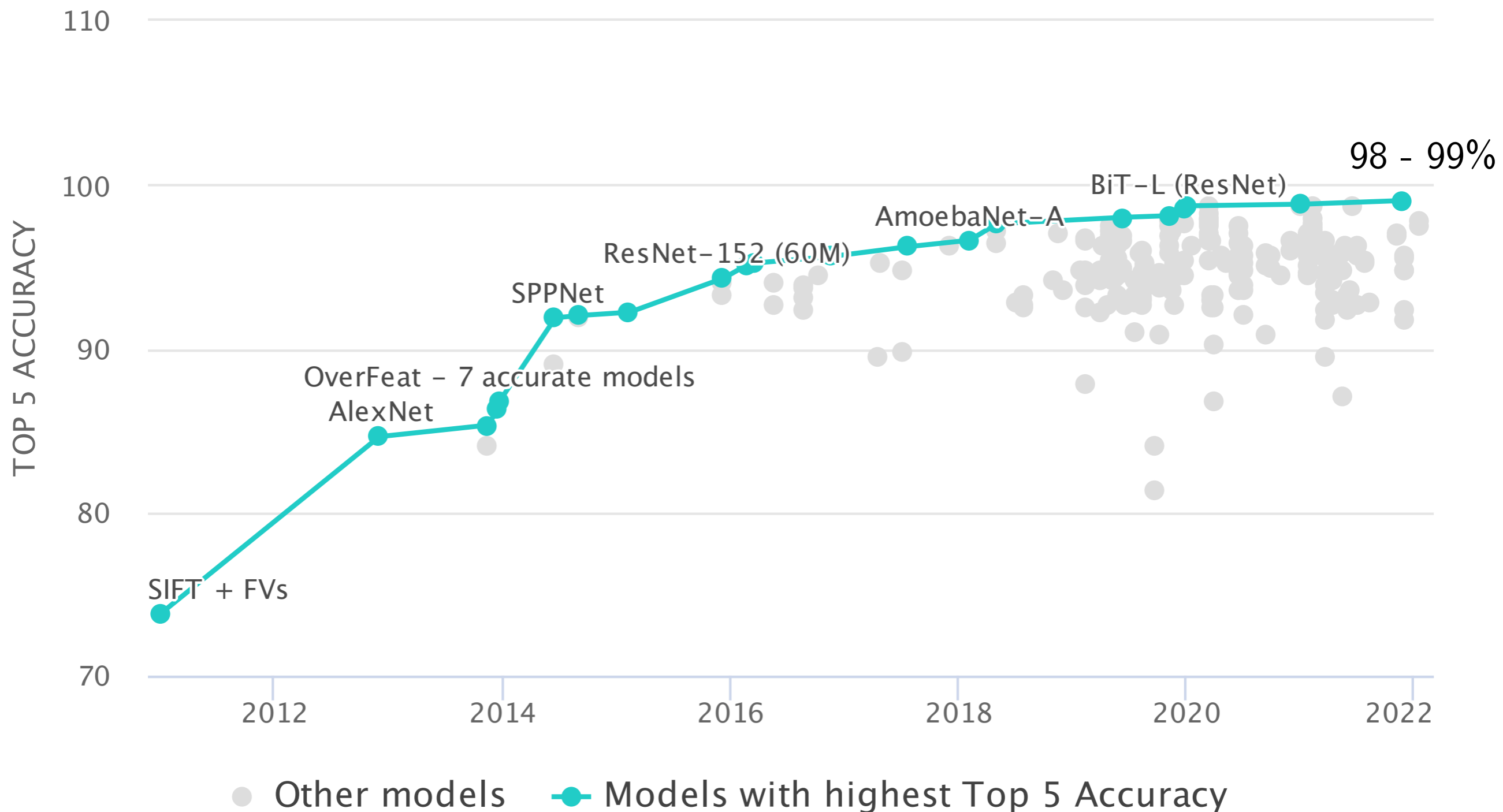
◆ Efficiency Matters



Big Picture: ImageNet Classification



◆ State of the Art (ImageNet 14M images)



Introduction

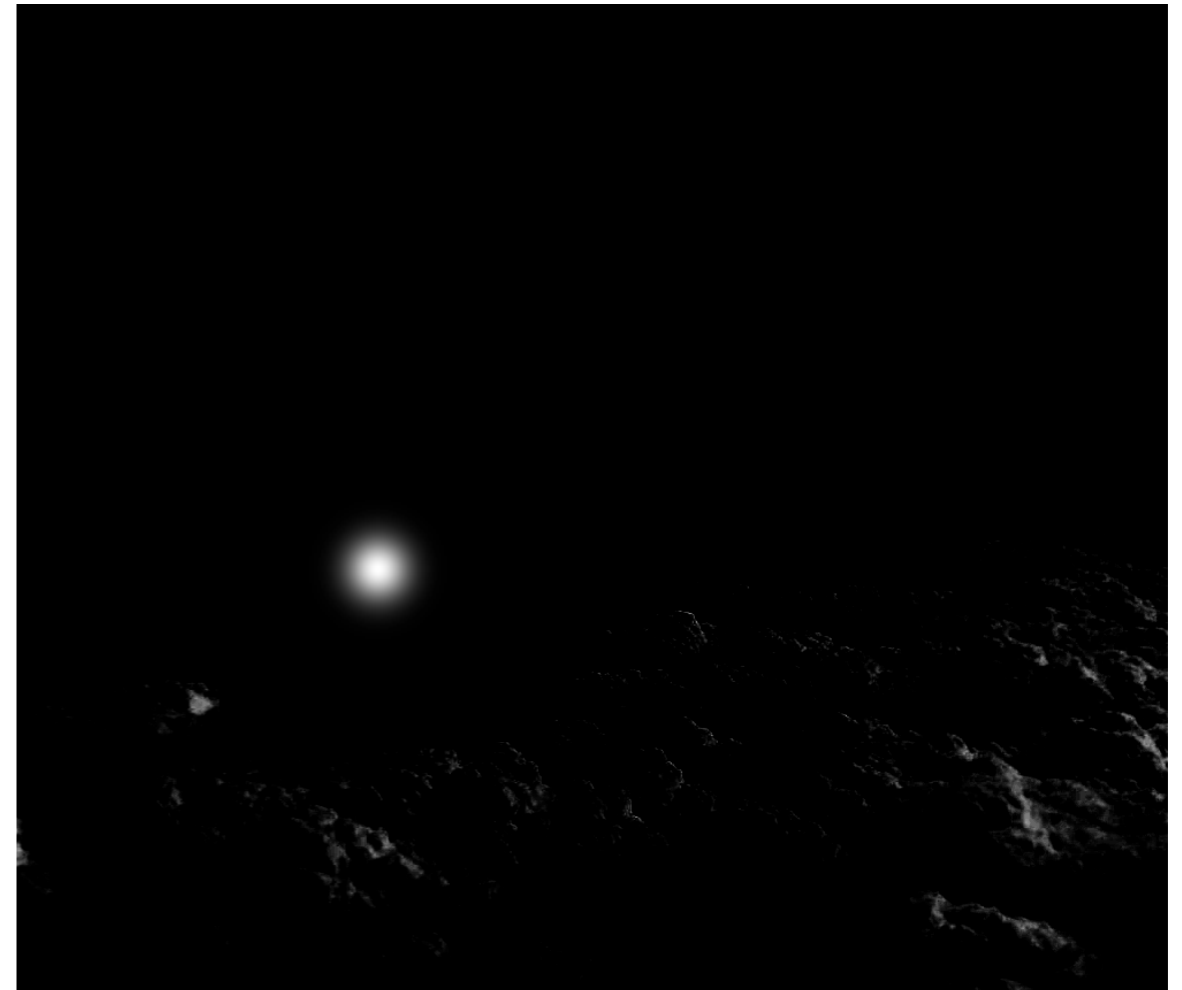
Template



Image



Quality of Match



Introduction

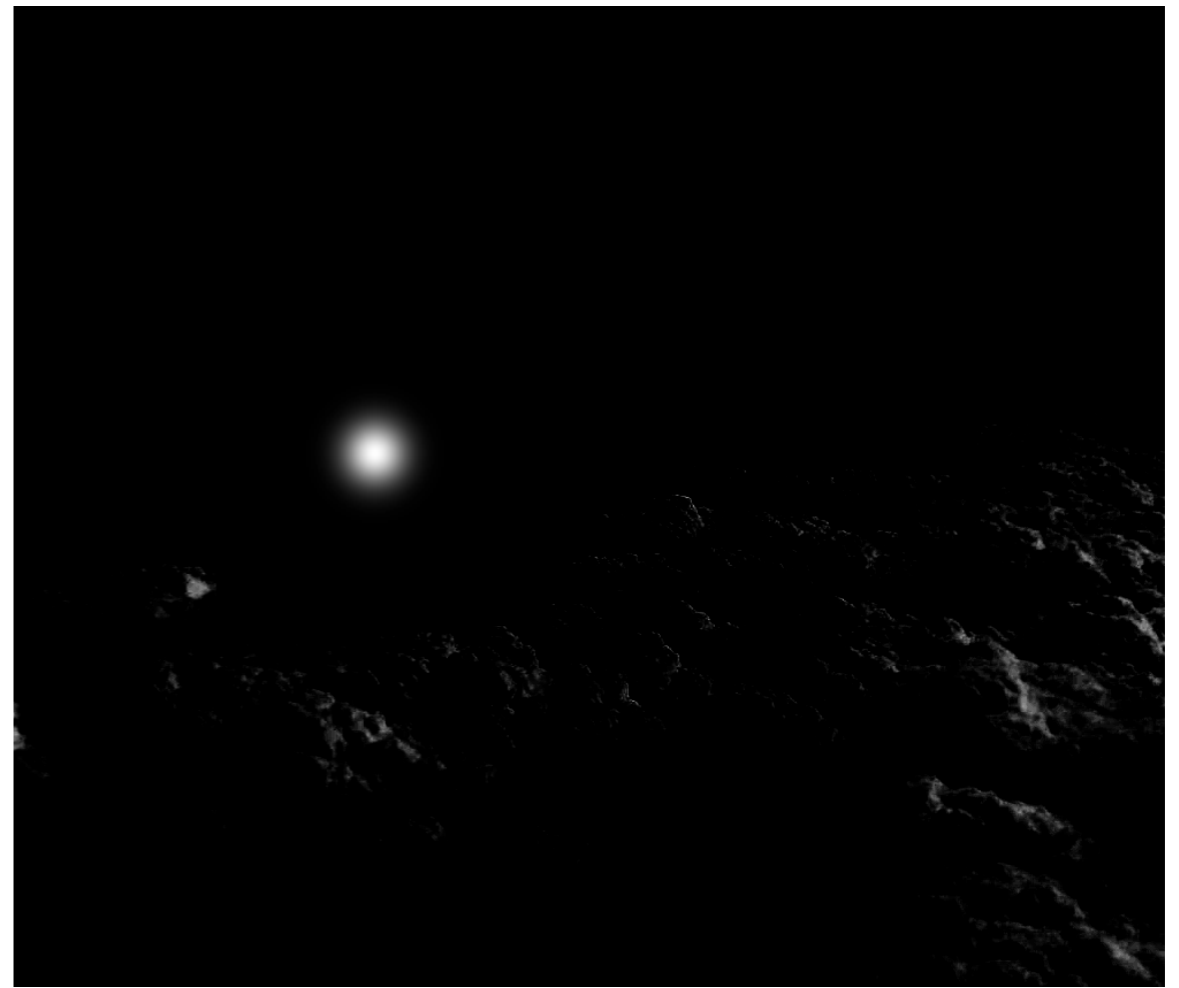
Template



Image



Quality of Match



- ◆ Translational equivariance idea: when the input shifts, the output shifts
 - Would be hard to achieve if the image was given as a general vector — we are using 2D grid structure and require that all locations are treated equally

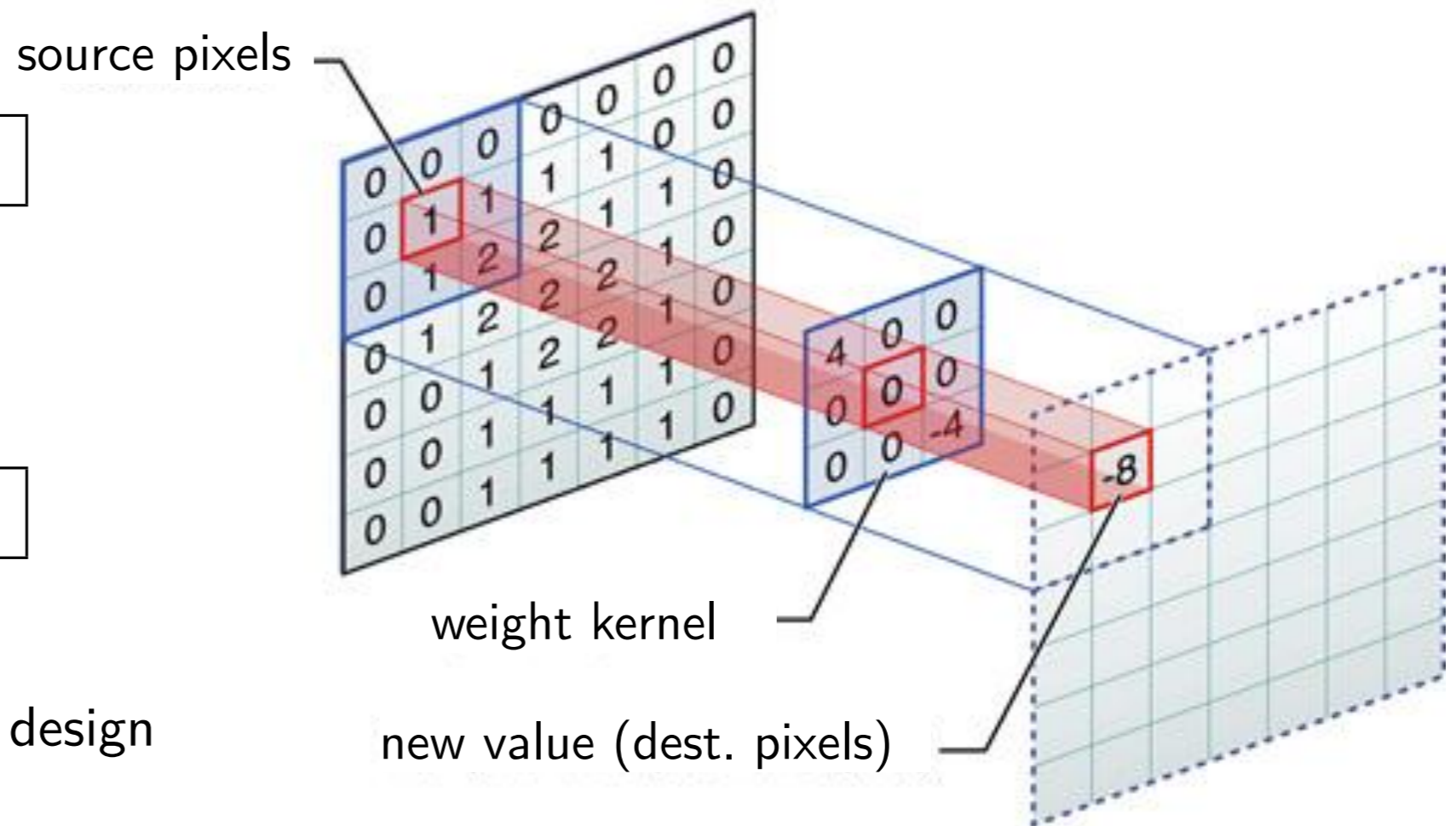
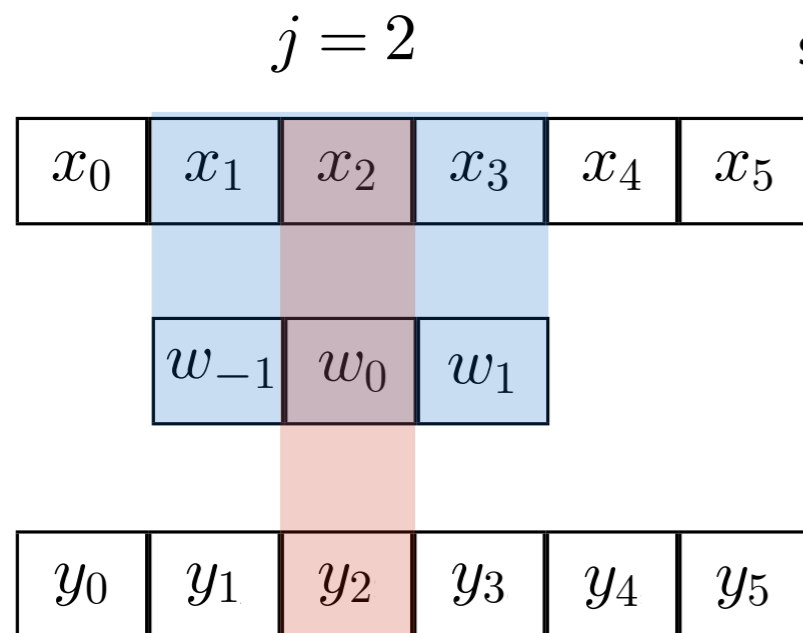
◆ Convolution and Correlation (1D)

- Convolution $y = w * x: y_j = \sum_{k=-h}^h w_k x_{j-k} = \sum_{k=-h}^h w_{-k} x_{j+k}$
- Cross-correlation $y = w \star x: y_j = \sum_{k=-h}^h w_k x_{j+k}$

output weight kernel input

flip of the weight matrix

Easily convertible, more convenient to consider cross-correlation in Deep Learning



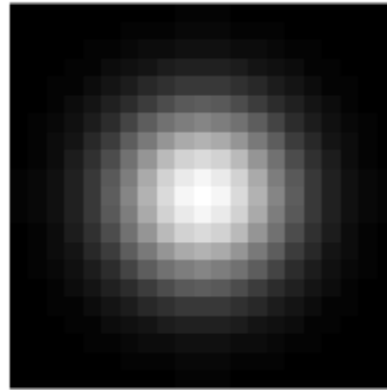
◆ Translation equivariance by design

Examples (Cross-Correlation)

Input



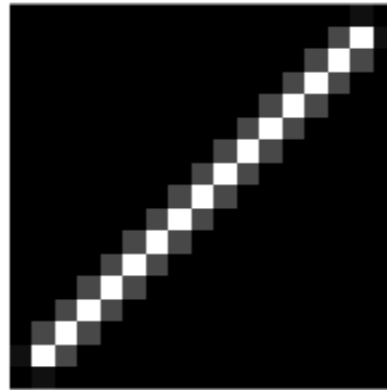
Kernel



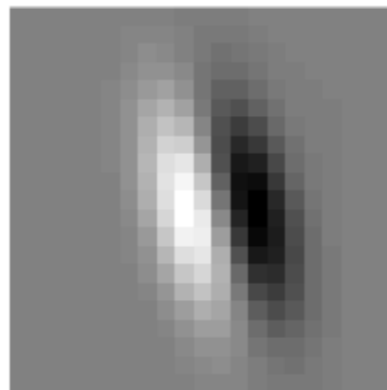
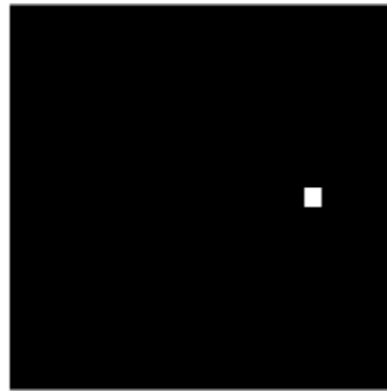
Output



Blur



Motion Blur

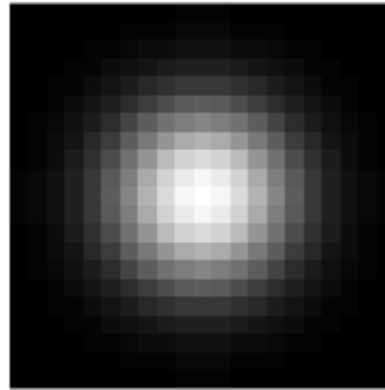


Examples (Cross-Correlation)

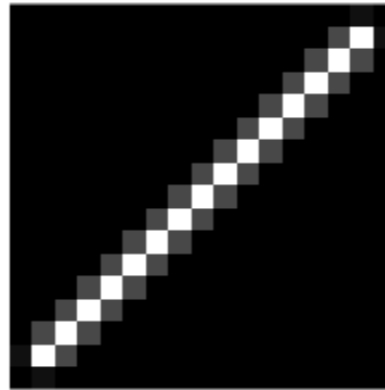
Input

Kernel

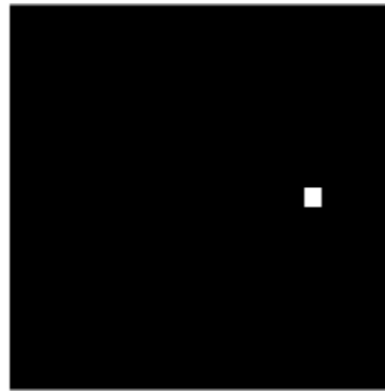
Output



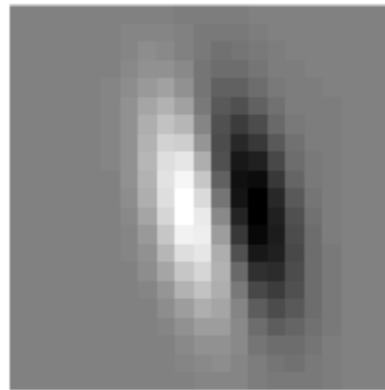
Blur



Motion Blur



Shift

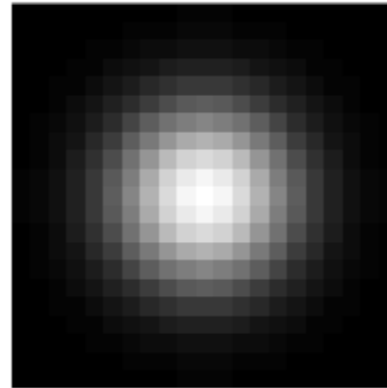


Examples (Cross-Correlation)

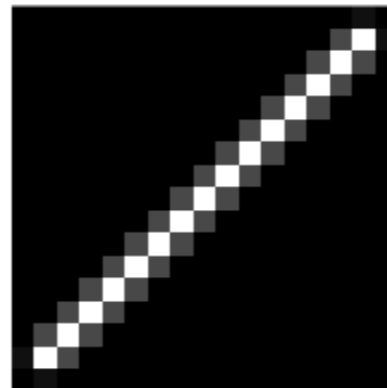
Input

Kernel

Output



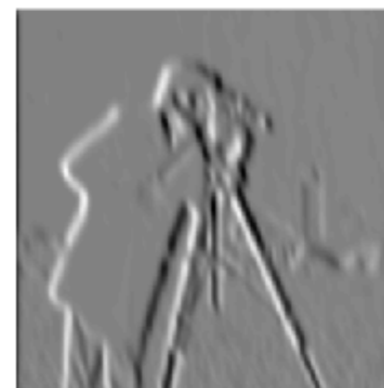
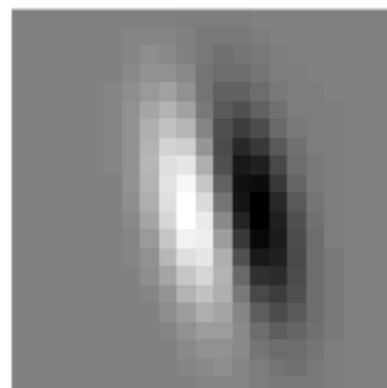
Blur



Motion Blur



Shift



Edge detector

Properties

◆ Cross-Correlation:

- $$y_i = \sum_{k=-h}^h w_k x_{i+k}$$

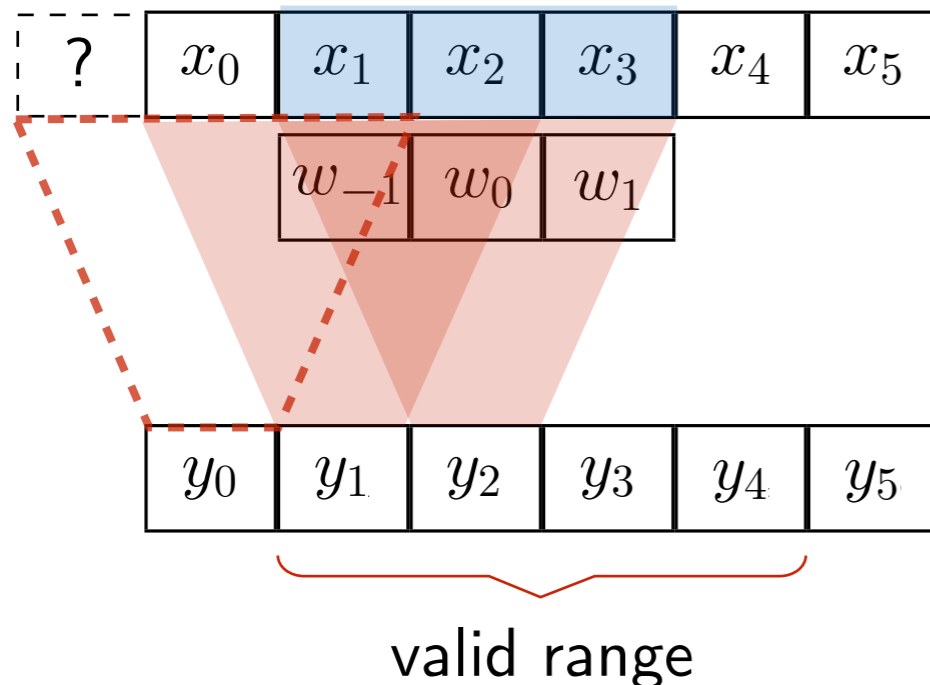
◆ As matrix-vector product: $y_i = \sum_j w_{j-i} x_j = \sum_j W_{ij} x_j$

- Relation: $j = i + k \Rightarrow k = j - i$
- **Compact representation** of certain linear transforms
- Everything that applies to linear transforms applies to convolution and cross-correlation

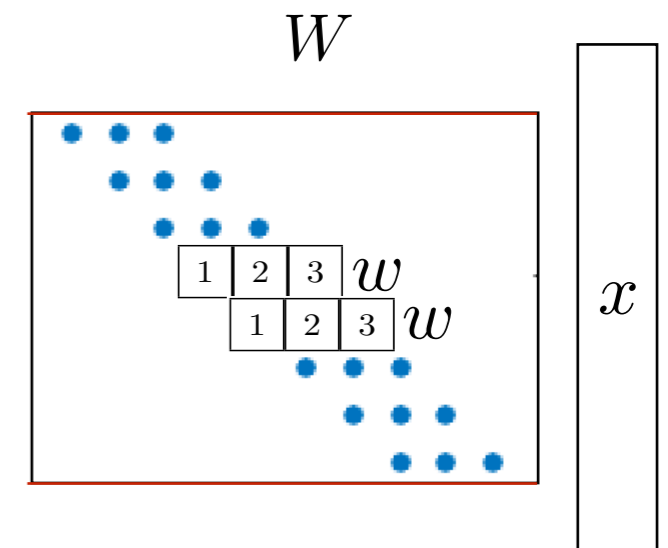
◆ **Valid** range for i :

$$0 \leq j \leq n \Rightarrow 0 \leq i - h, i + h \leq n \Rightarrow h \leq i \leq n - h.$$

- Optionally may **pad** input with zeros to obtain **same** range as unpadding input

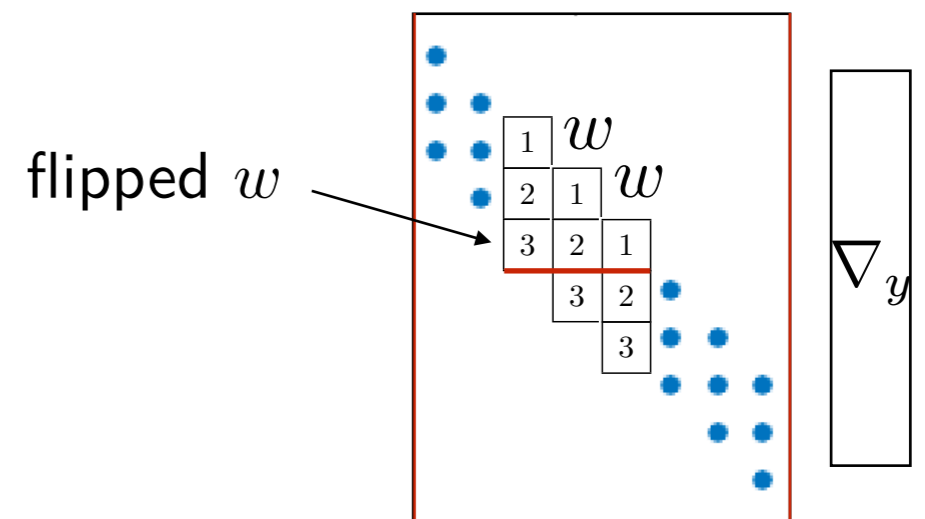


Correlation (valid)



Known backprop

W^T



= Convolution (with padding)

- ◆ As a binary operation $y = w * x$
 - Everything that applies to linear operators, eg. associativity: $u * (w * x) = (u * w) * x$
 - Commutativity for convolutions: $w * x = x * w$:
$$\sum_k w_k x_{i-k} = \sum_j x_j w_{i-j}$$
 - No commutativity for cross-correlation. But $u \star w \star x = w \star u \star x$
- ◆ Examples:
 - $\text{edge_filter}(\text{blur}(\text{image})) = \text{blur}(\text{edge_filter}(\text{image})) = (\text{blur}(\text{edge_filter}))(\text{image})$
 - $\text{filter}(\text{translation}(\text{image})) = \text{translation}(\text{filter}(\text{image}))$
equivariance w.r.t. translation



When the image shifts, the output shifts
Great prior knowledge for learning

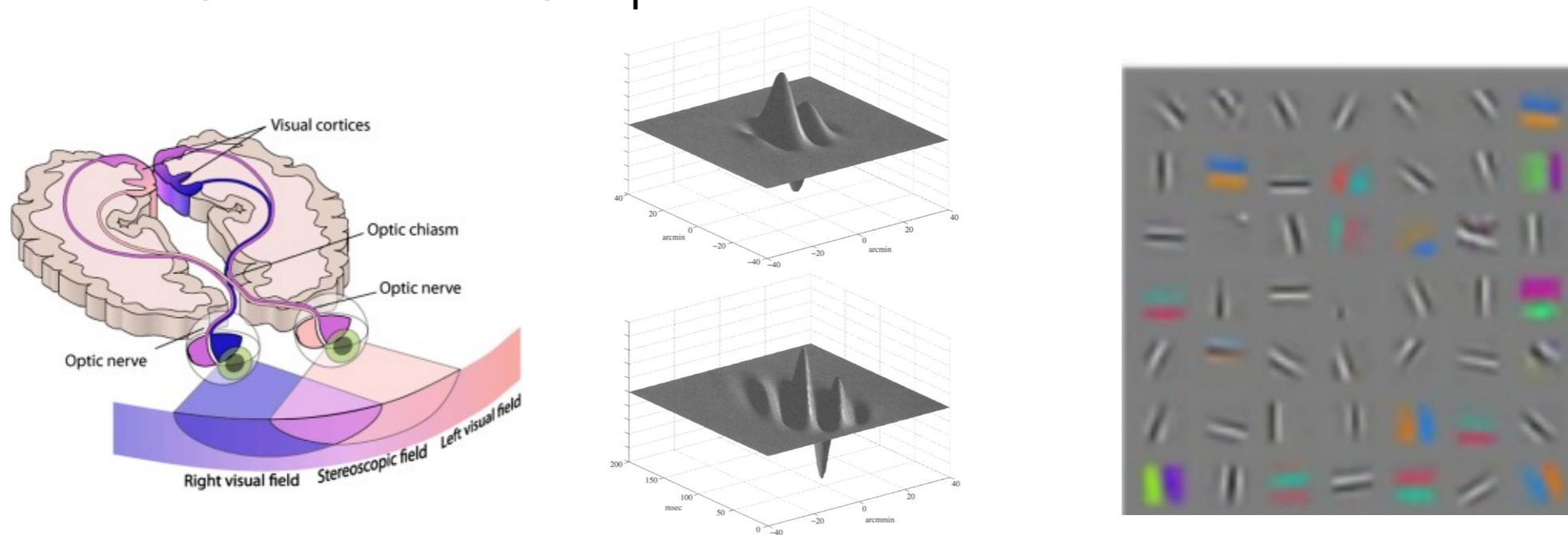
(★) Can you show equivariance of convolution to sub-pixel displacements?

- ◆ In fact, linearity + translation equivariance = convolution

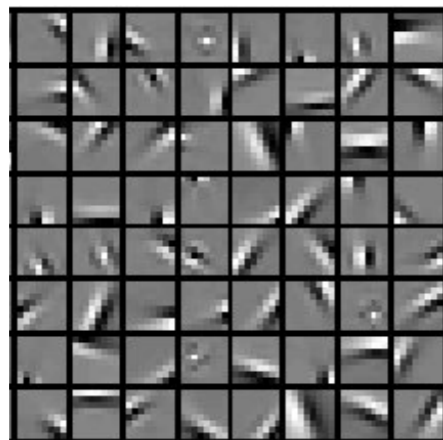
First Convolutional Layer

- ◆ Large filters are not very useful:
 - ◆ Think of viewpoint changes, object deformations, variations within a category
 - ◆ Small filters capture elementary, non task-specific, features

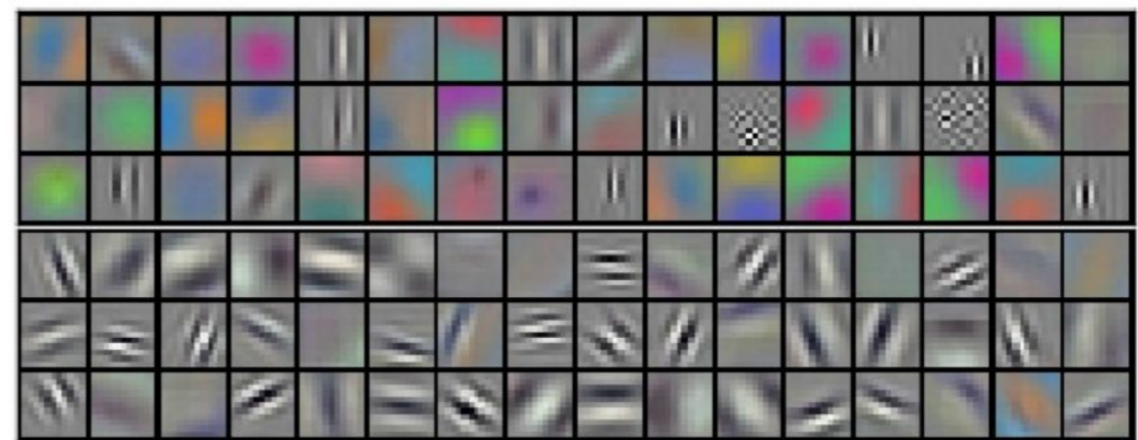
Gabor Filters - Computational model for V1 cells:



PCA of Image Patches



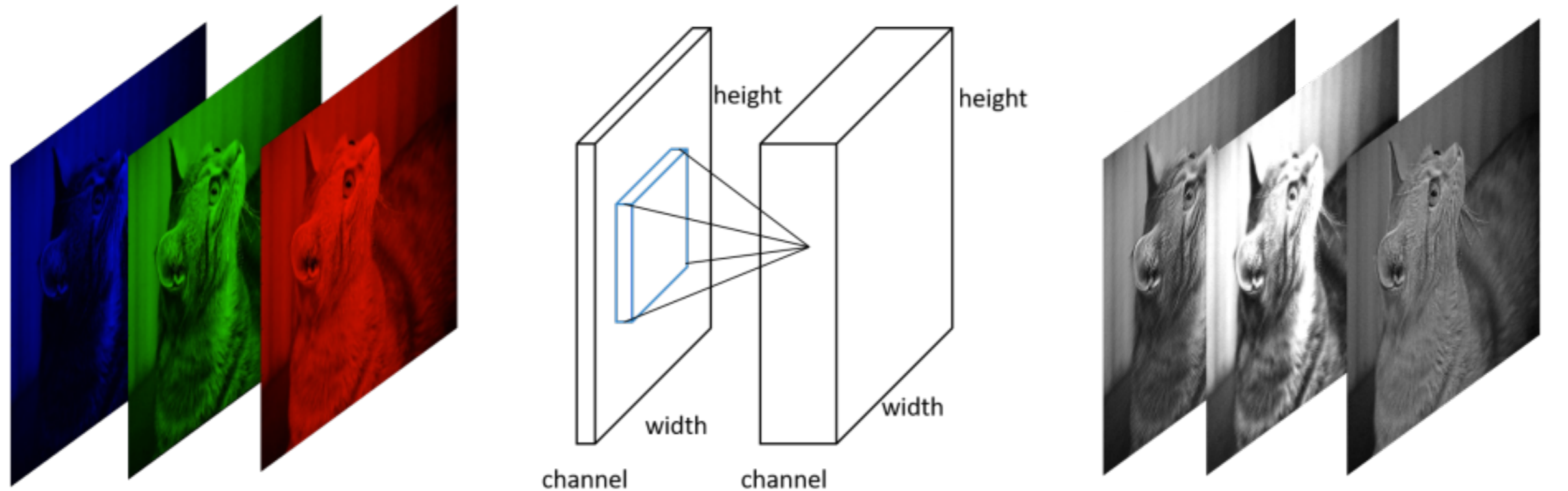
CNN first layer filters (learned)



Multi-Channel Convolution

◆ We just had:

- color input images -> convolution kernel needs to have 3 channels
- stack of filters -> channels of the output



◆ Multi-channel cross-correlation:

$$\sum_c \sum_{\Delta i} \sum_{\Delta j} x_{c,i+\Delta i,j+\Delta j} w_{o,c,\Delta i,\Delta j} = y_{o,i,j}$$

input channel

filter spatial dimensions

output channel

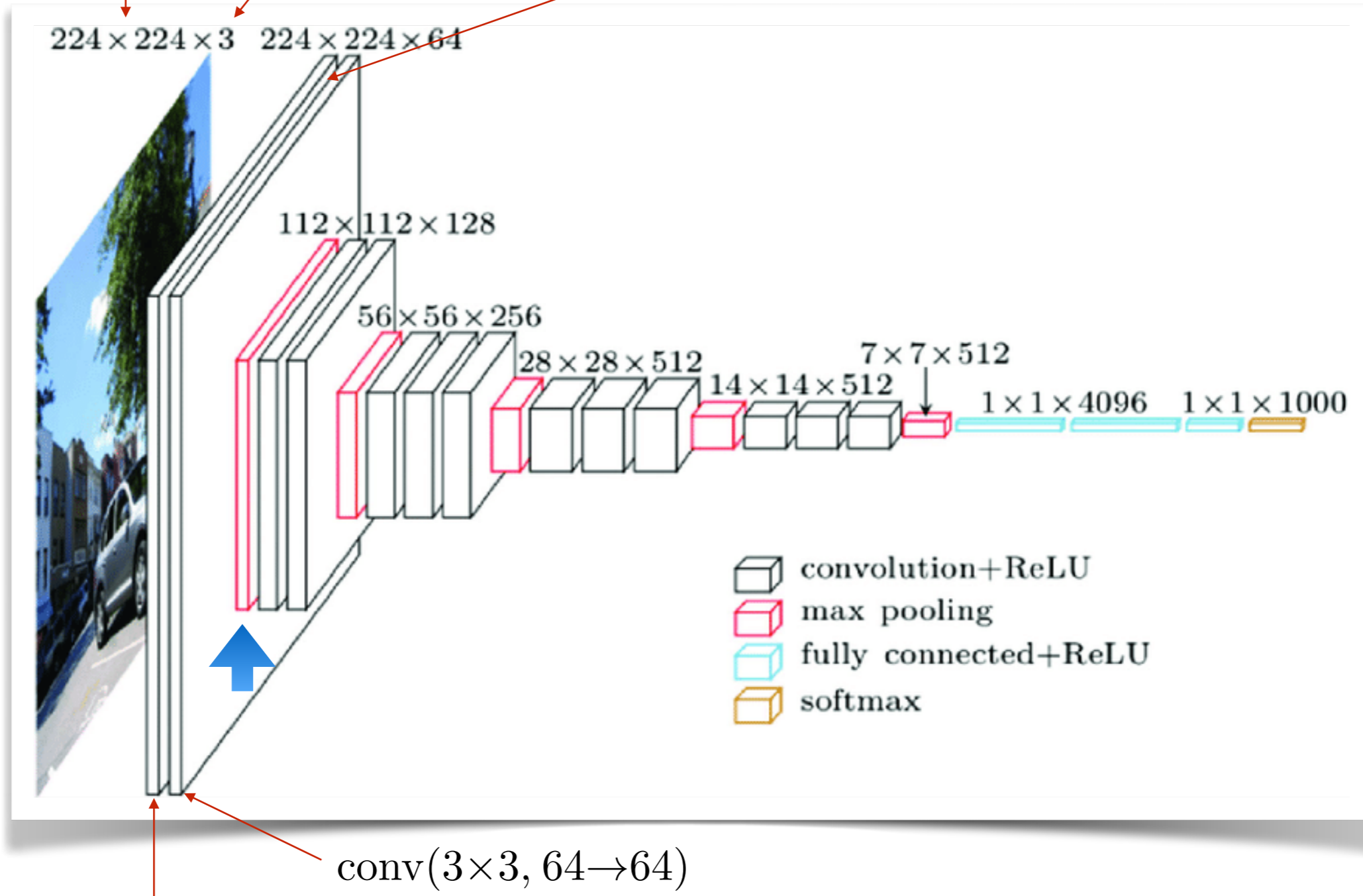
- input is 3D tensor, weight is 4D tensor, output is 3D tensor
- Essentially: a cross-correlation on spatial dims and fully connected on channel dims

Classification CNN

Spatial size of the input image

channels

feature maps



Result of $\text{conv}(K \times K, 3 \rightarrow 64)$ followed by ReLU

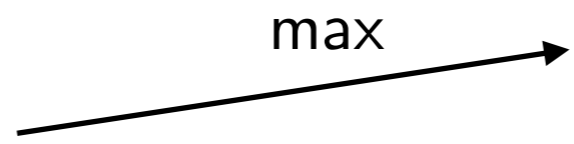
◆ Eventually want to classify -> need to reduce spatial dimensions

Pooling

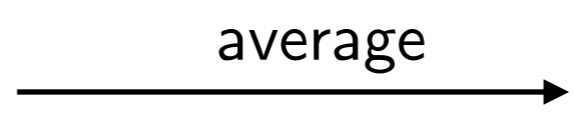
◆ Following approaches are used to reduce the spatial resolution:

- **max pooling**
- **average pooling**
- subsampling -> **convolution with stride**

| | | | |
|----|----|----|----|
| 3 | 13 | 17 | 11 |
| 5 | 3 | 1 | 23 |
| 7 | 1 | 2 | 3 |
| 11 | 17 | 1 | 4 |

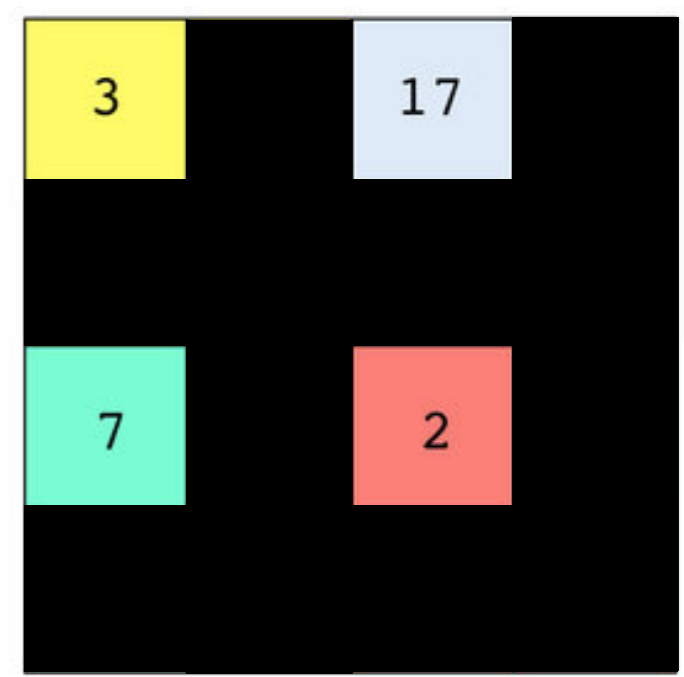
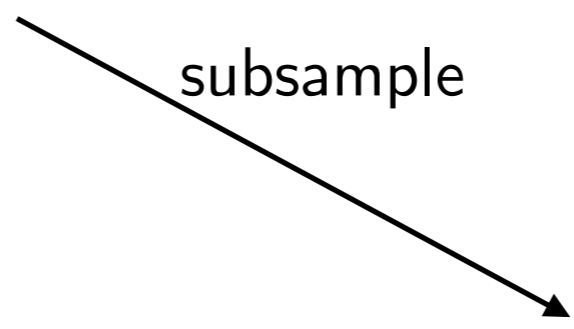


| | |
|----|----|
| 13 | 23 |
| 17 | 4 |



| | |
|---|-----|
| 6 | 13 |
| 9 | 2.5 |

(linear)

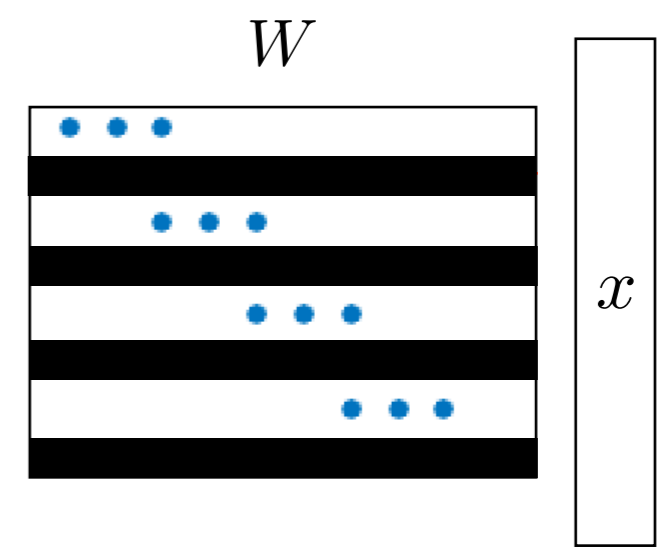
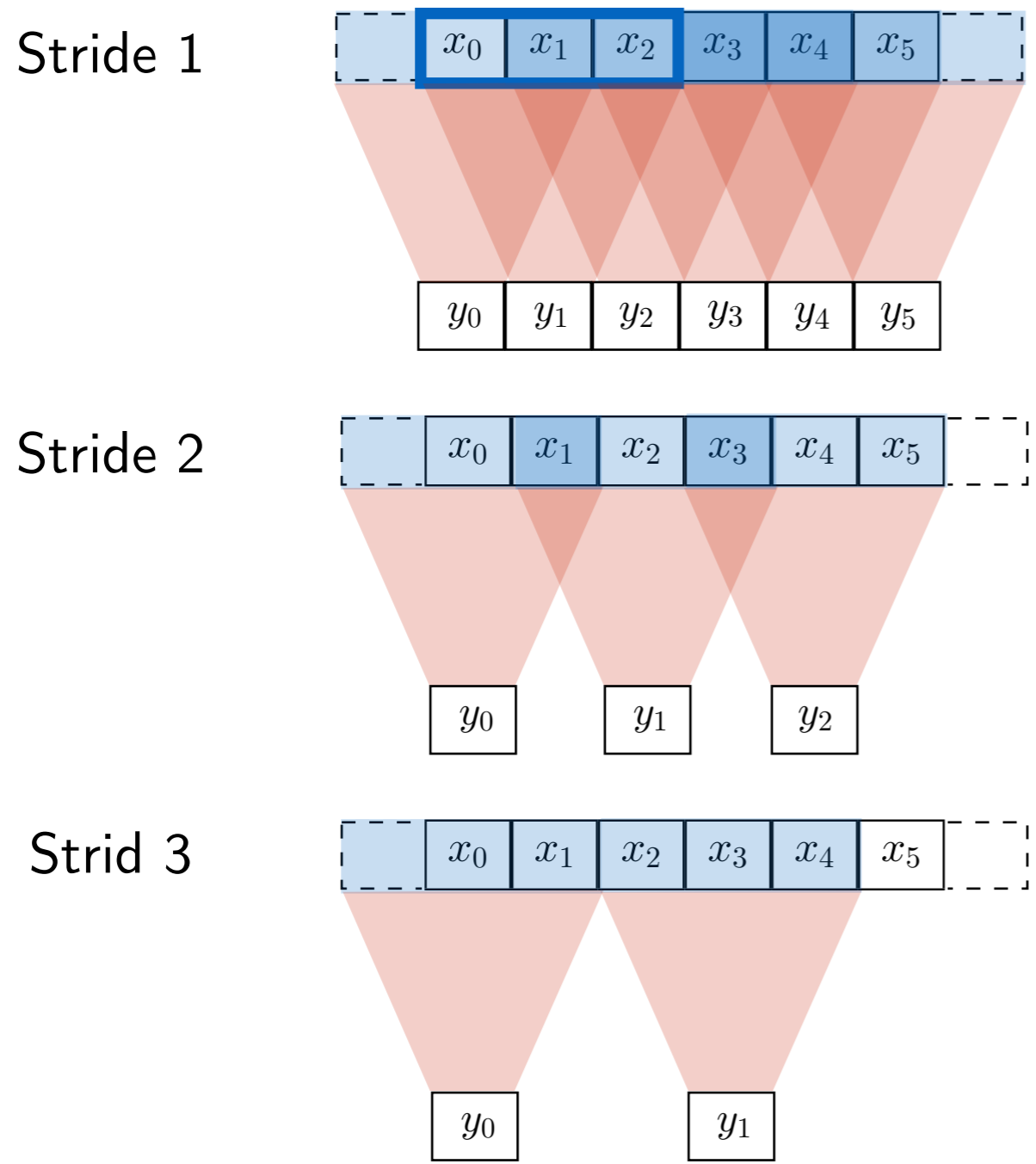


(linear)

- ◆ Somewhat robust to translations
- ◆ Once spacial resolution has been decreased, we can afford to increase the number of channels

Convolution with Stride

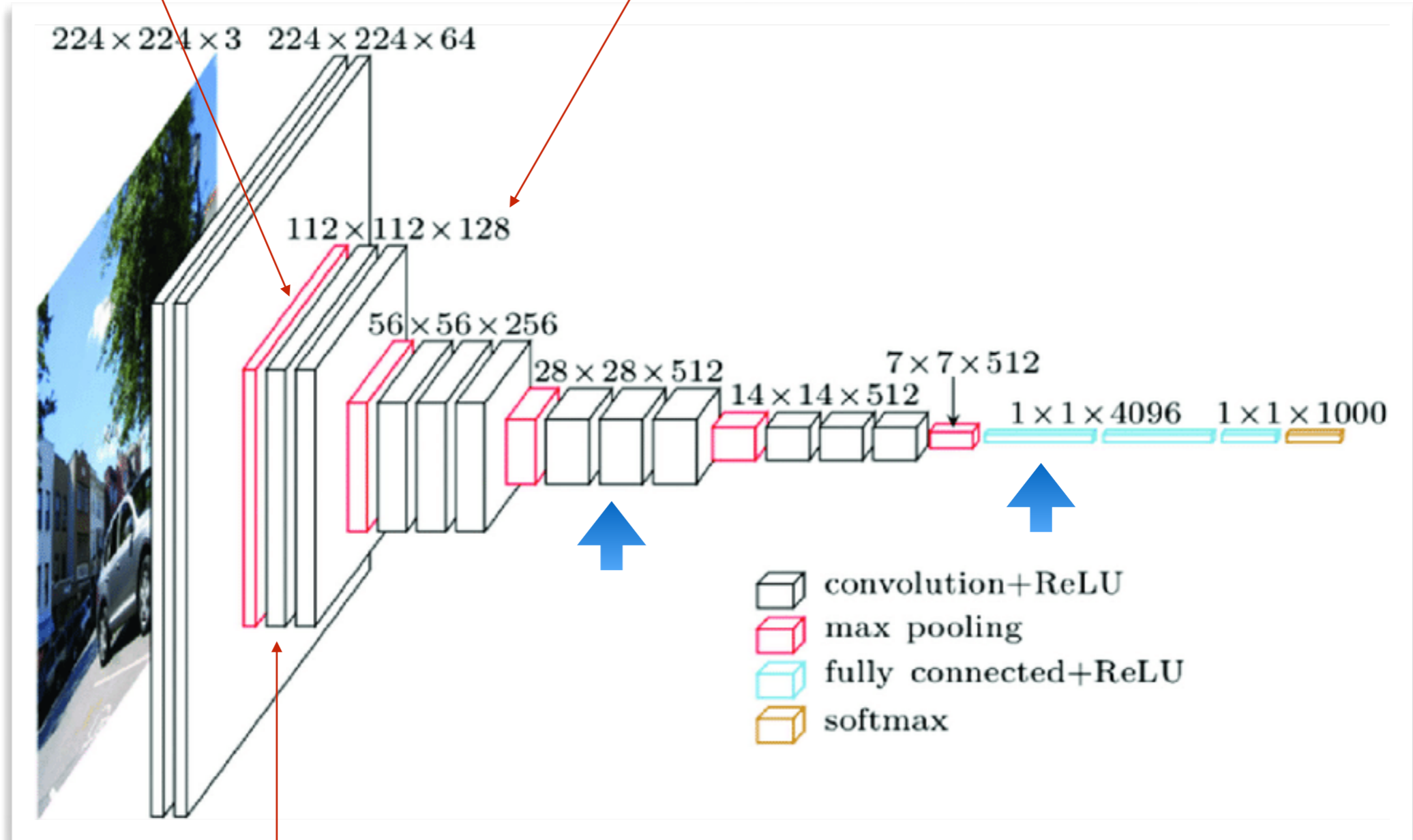
- Full convolution + subsampling is equivalent to calculating the result at the required locations only, stepping with a **stride**



Classification CNN

Reduced spatial size

can afford more channels

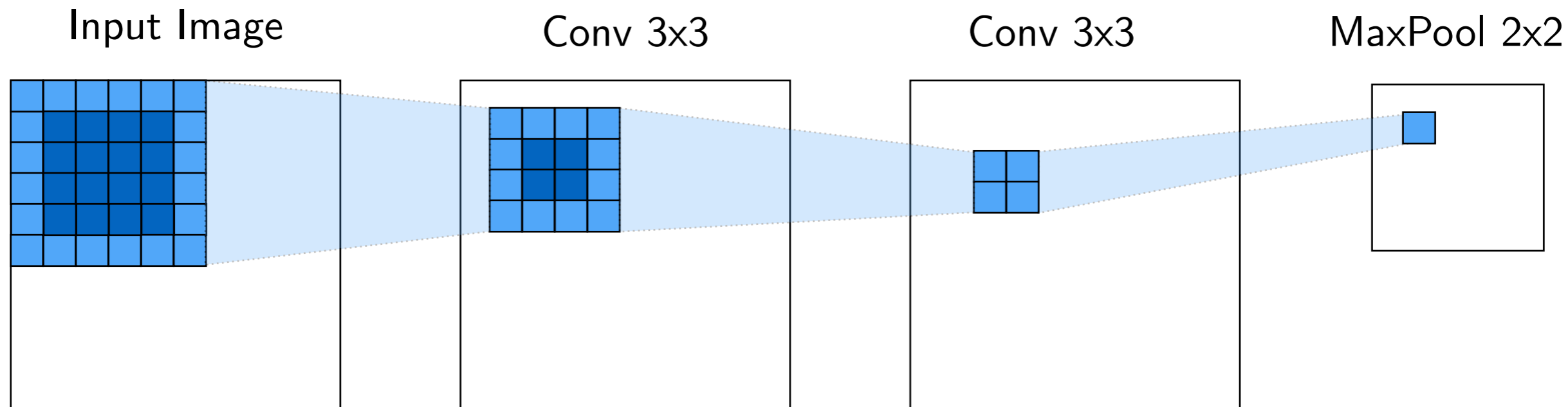


conv(3x3, 64→128)

- ◆ Combining convolutions and spatial pooling increases units receptive field

Receptive Field

- ◆ Receptive Field = pixels in the input which contribute to the specific output



- ◆ Small convolutions are not sufficient to create large enough receptive fields.
Example:
 - Want to classify images of size 256x256
 - Each 3x3 convolution increases the receptive field by 2 pixels
 - Would take 128 convolutional layers
- ◆ Need pooling / strides / larger filters

Weight Kernel Sizes



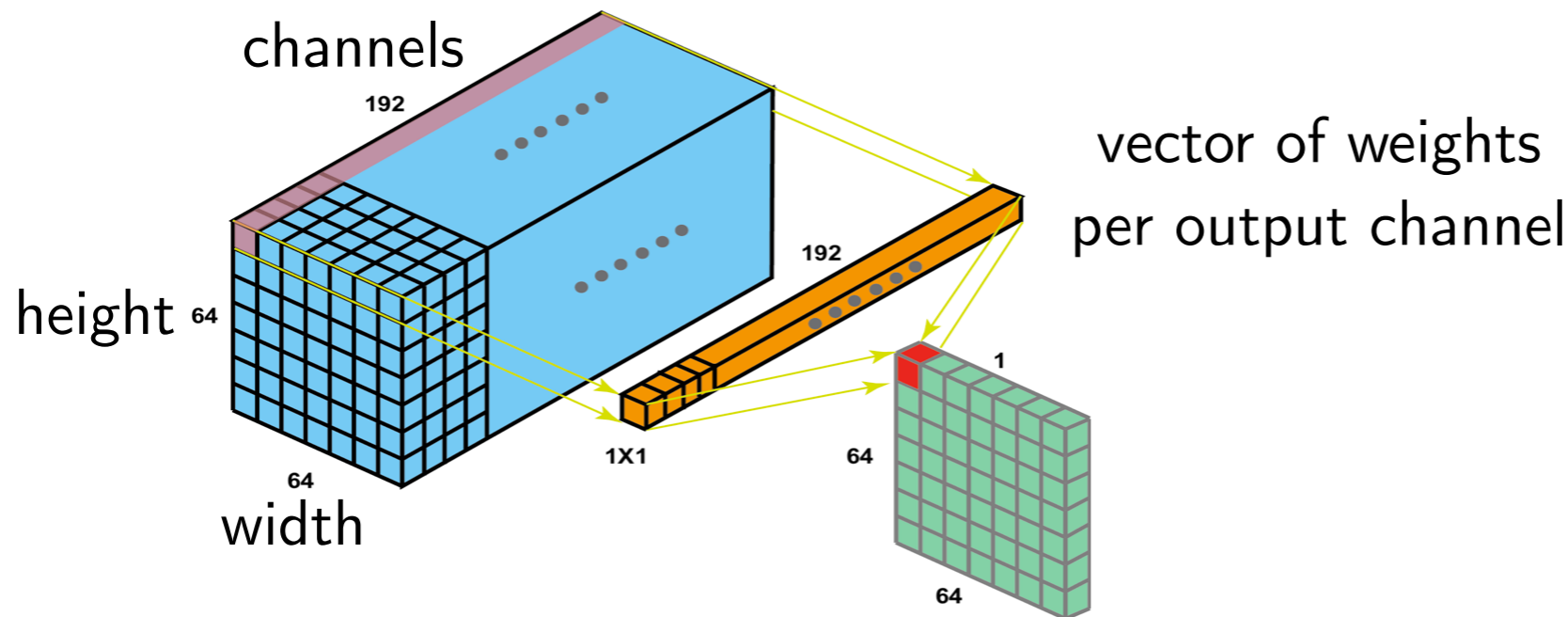
- ◆ With pooling we reduced the size of feature maps. What about filter kernels?
 - First layer: $(7 \times 7, 3 \rightarrow 64) \approx 10^3$ – can afford large filter size
 - Second layer: $(3 \times 3, 64 \rightarrow 64) \approx 3 \cdot 10^4$ – small filter size preferable
 - Layers with more channels: $(3 \times 3, 256 \rightarrow 256) \approx 5 \cdot 10^5$ – become expensive
- ◆ Need further efficient parametrization techniques
 - Depth-wise separable convolutions:
spatial convolution same for all channels composed with a general linear transform on the channels (1x1 convolution)
 - Something in between:
 $\text{conv}(K \times K, S \rightarrow S)$ composed with $\text{conv}(1 \times 1, C \rightarrow S)$, $S < C$

1x1 Convolution

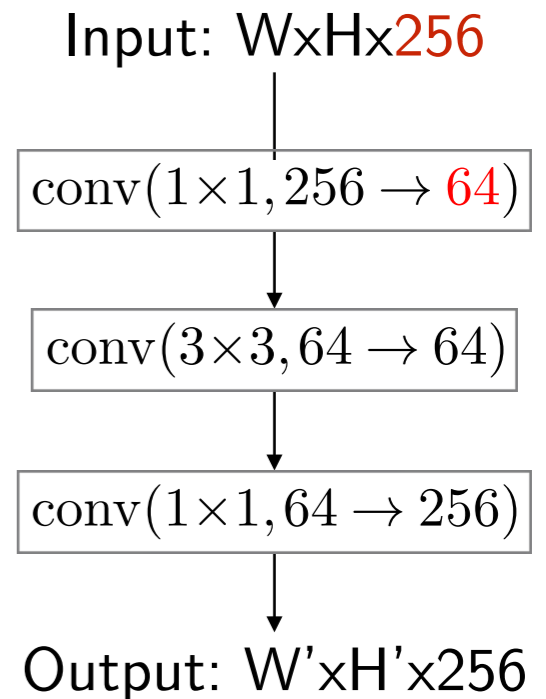
- ◆ Kernel size 1x1:

$$\begin{aligned}
 y_{o,i,j} &= \sum_c \sum_{\Delta i=0} \sum_{\Delta j=0} w_{o,c,\Delta i,\Delta j} x_{c,i+\Delta i,j+\Delta j} \\
 &= \sum_c w_{o,c,0,0} x_{c,i,j}
 \end{aligned}$$

- ◆ For all i, j a linear transformation on channels with a matrix $w_{o,c,0,0}$

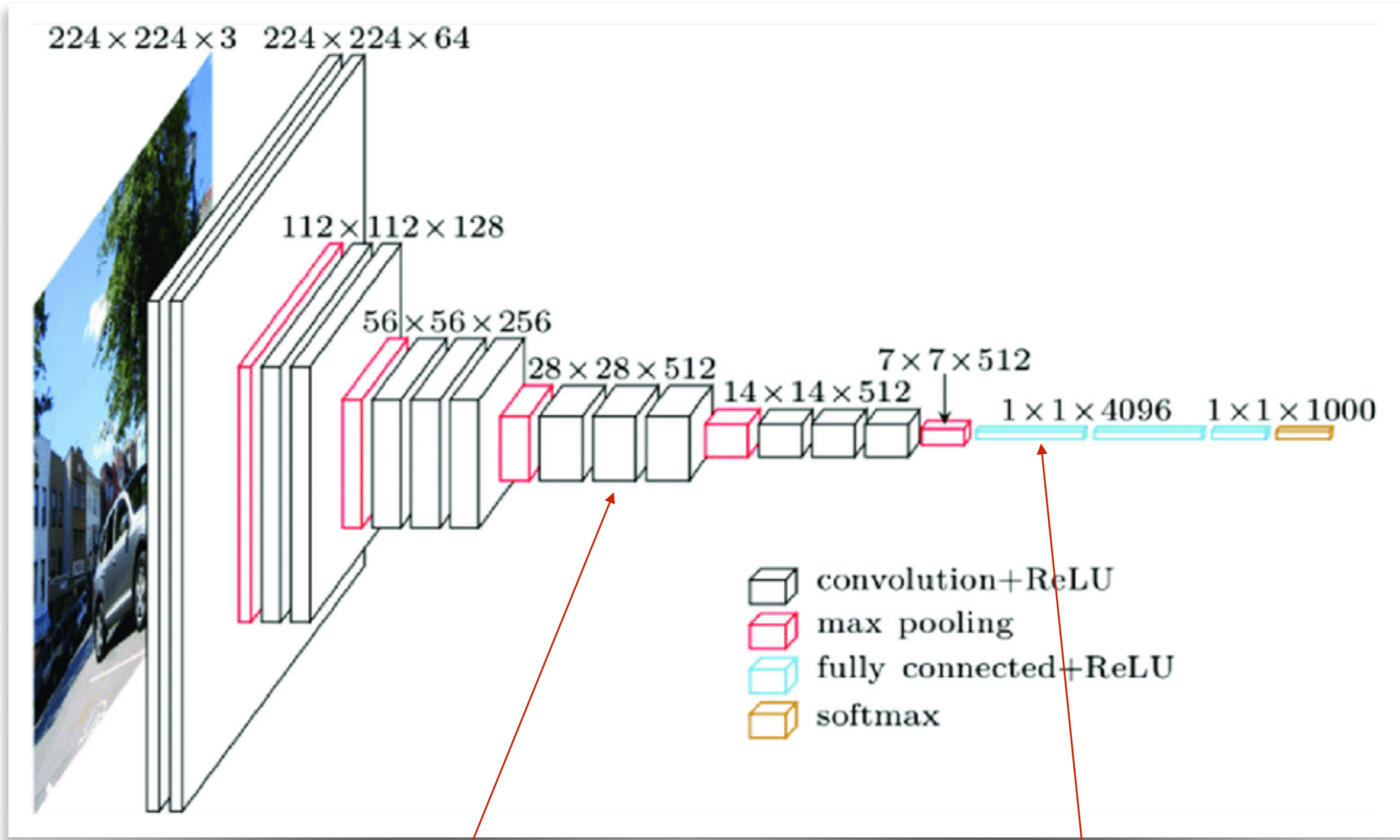


Example 3x3, 256→256,
is too expensive, simplify:



- ◆ Useful to perform operations along channels dimension:
 - Increase /decrease number of channels
 - Normalization operations
 - In combination with purely spatial convolution = separable transform

Classification CNN



Could use efficient module here

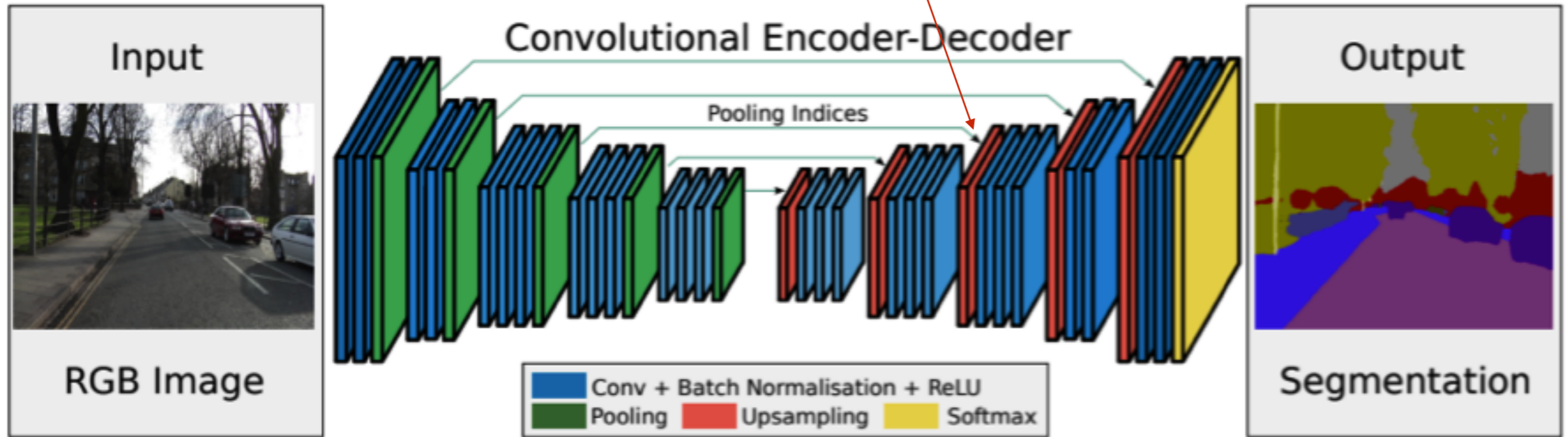
1x1 convolution for input of size 1x1 is equivalent to fully connected

◆ Second last layer has $4096 \times 4096 = 16\text{M}$ parameters!

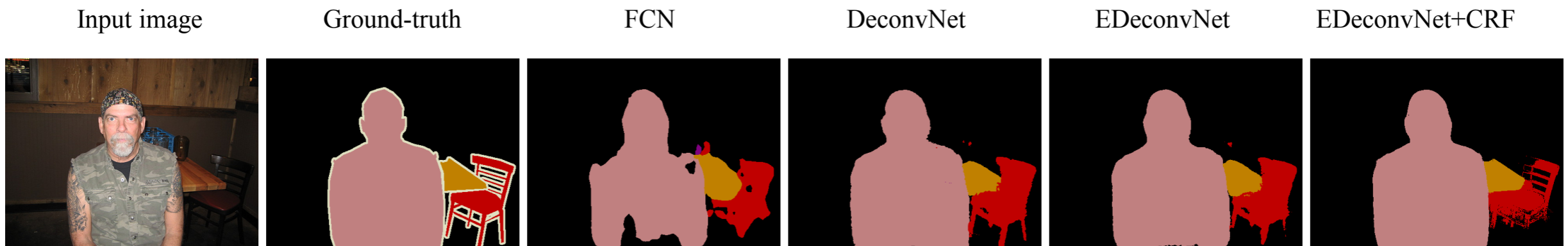
More Convolutions in DL

Deconvolution

Semantic Segmentation Architectures need unpooling / upsampling



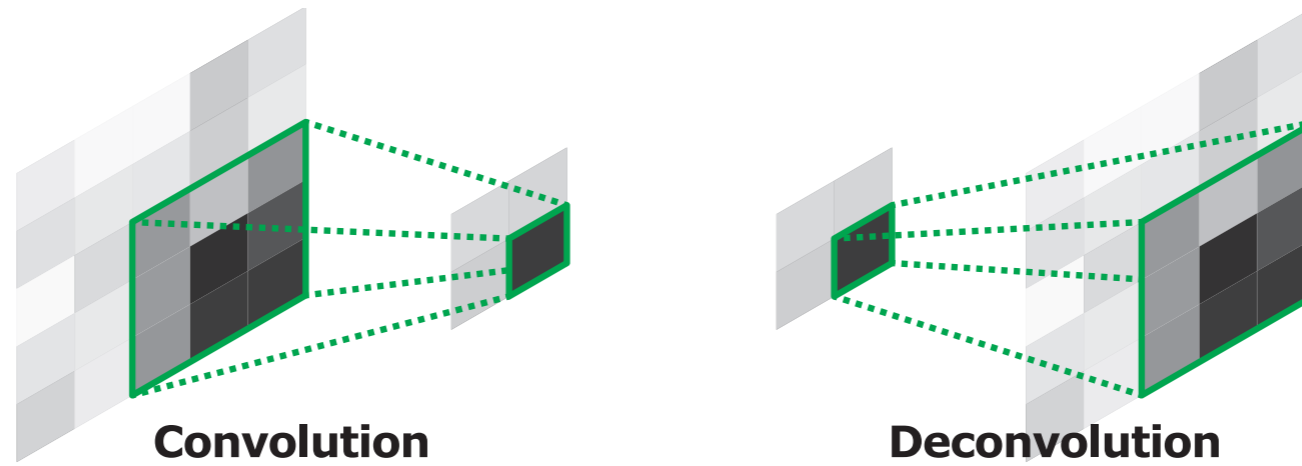
We will look at up-sampling with “transposed” convolution (“deconvolution”)



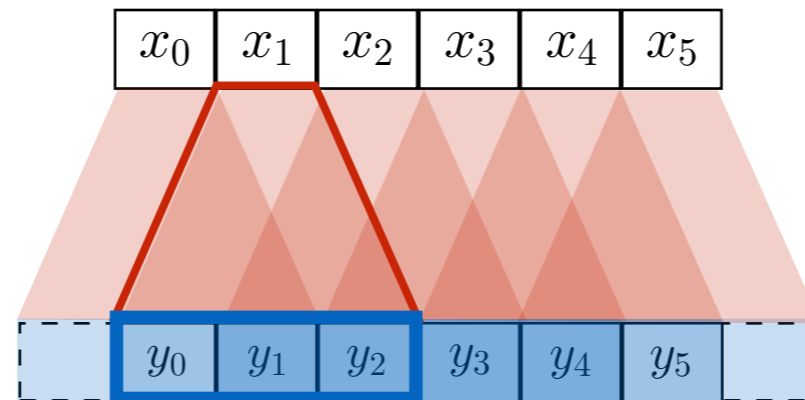
[Noh et al. (2015) Learning Deconvolution Network for Semantic Segmentation]

Transposed Convolution

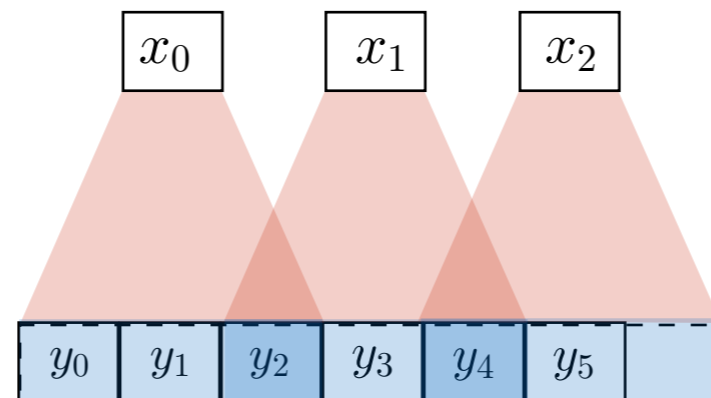
◆ Deconvolution = Transposed convolution = backprop of convolution



Stride 1



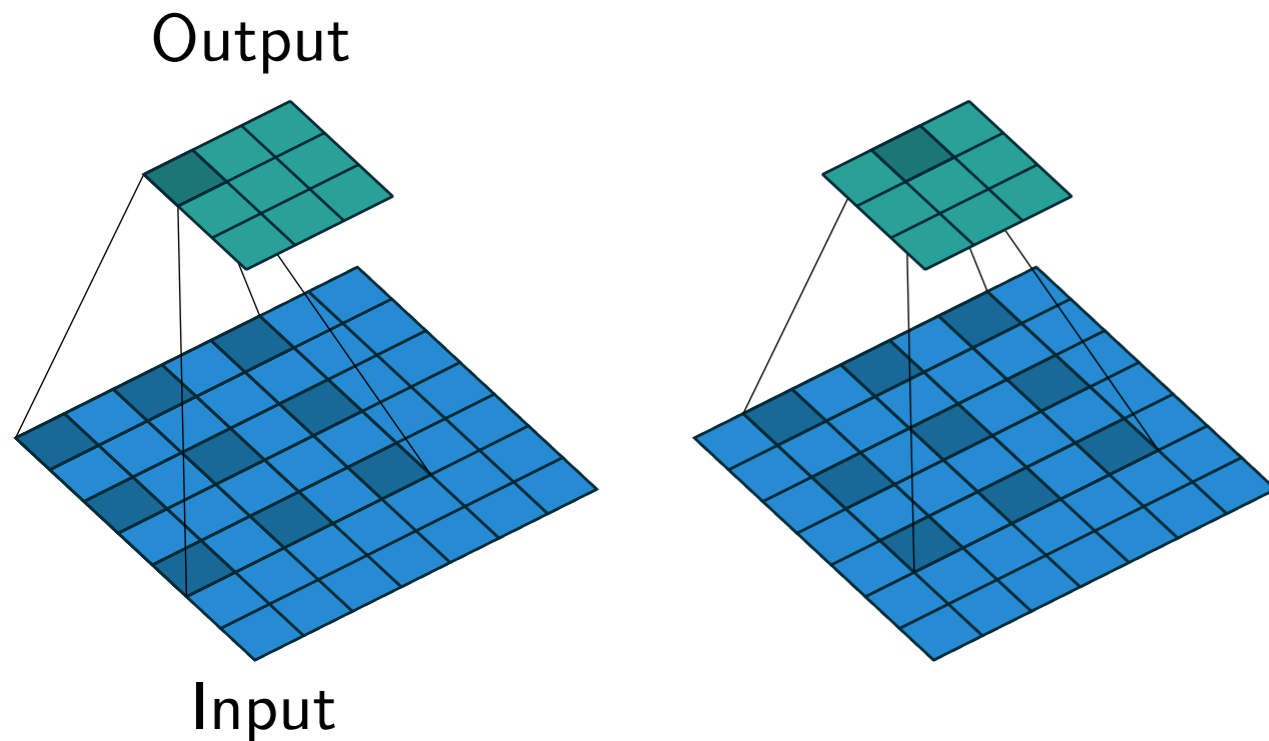
Stride 2



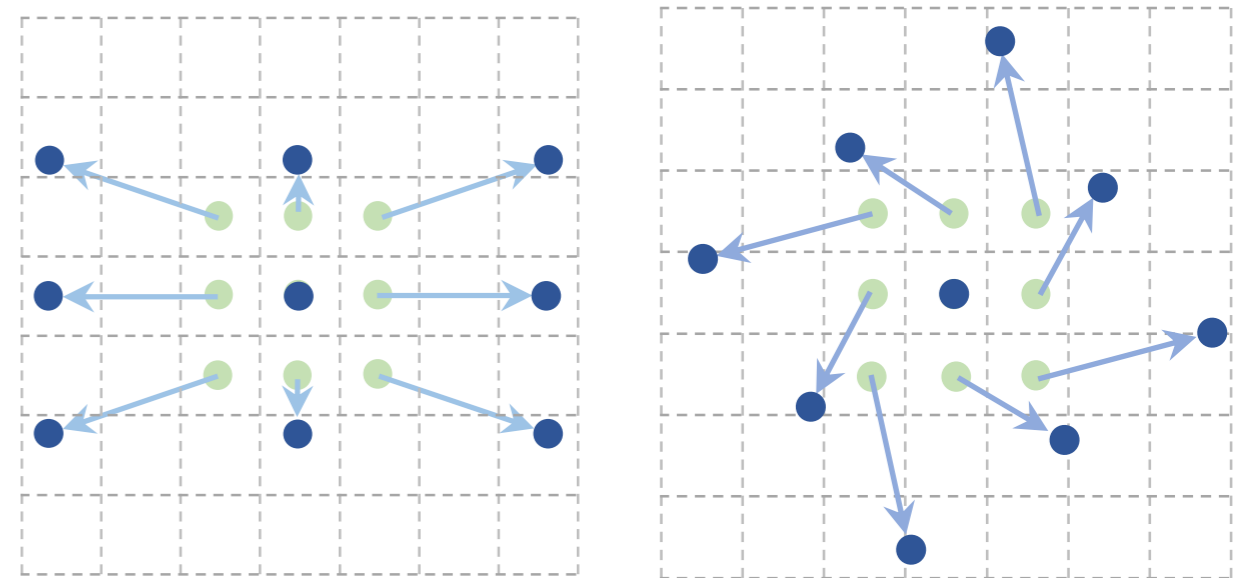
Sparse Convolutions

- ◆ Want to increase receptive field size
 - without decreasing spatial resolution and having too many layers
 - Can increase kernel size, but it was also costly
 - Can use a sparse mask for the kernel

Dilated convolutions

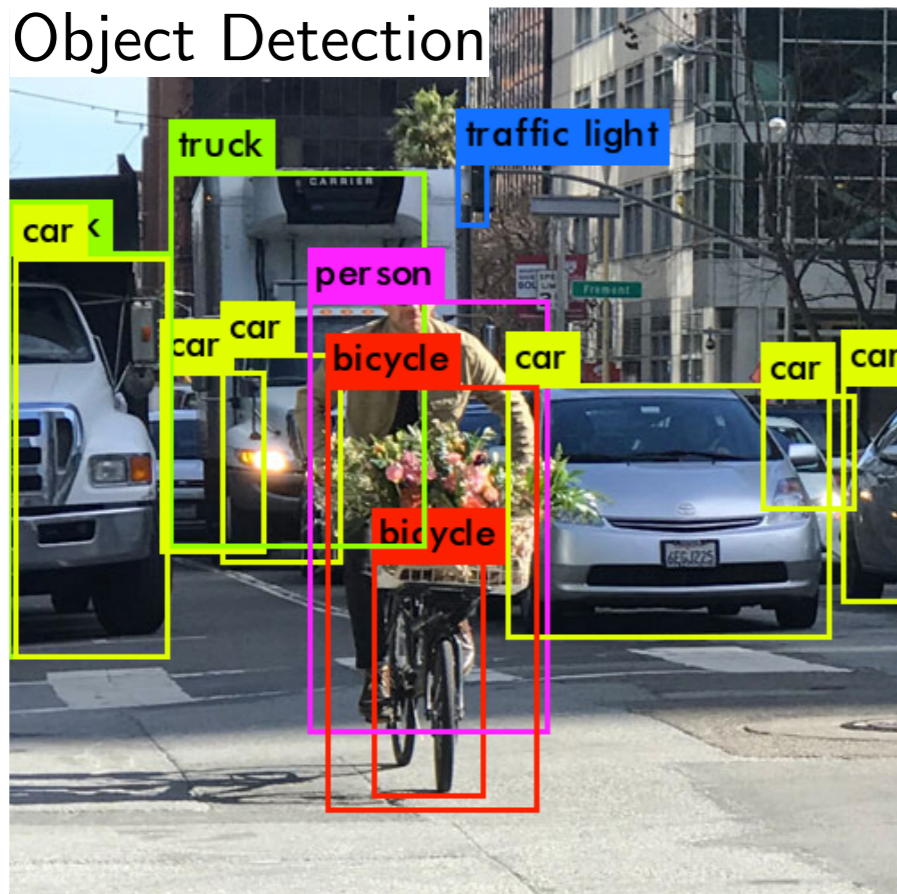


Can even learn sparse locations —
deformable convolutions

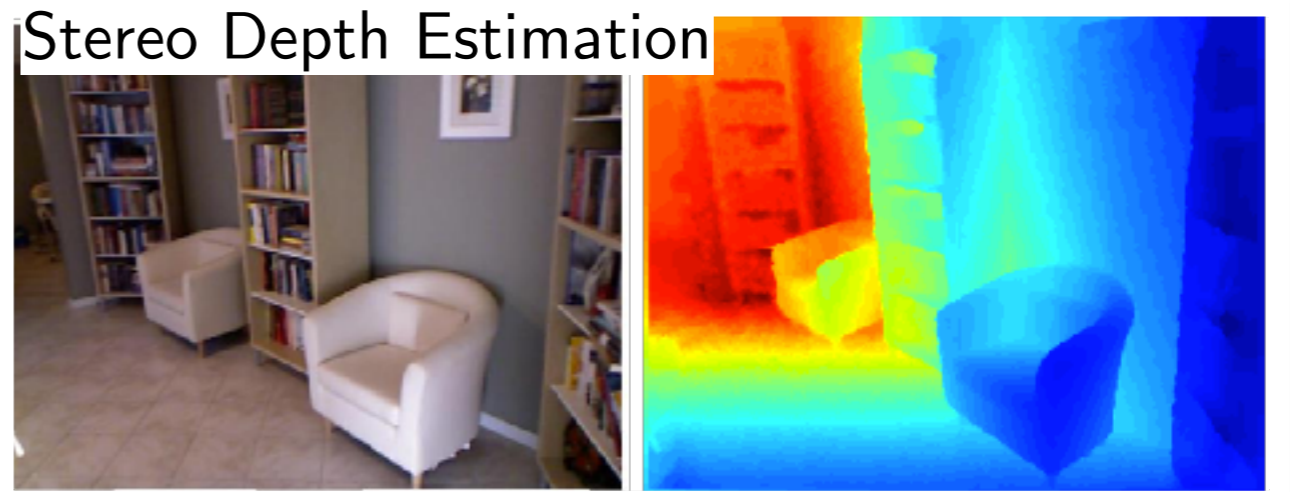


Many More Examples and Smart Architectures

Object Detection



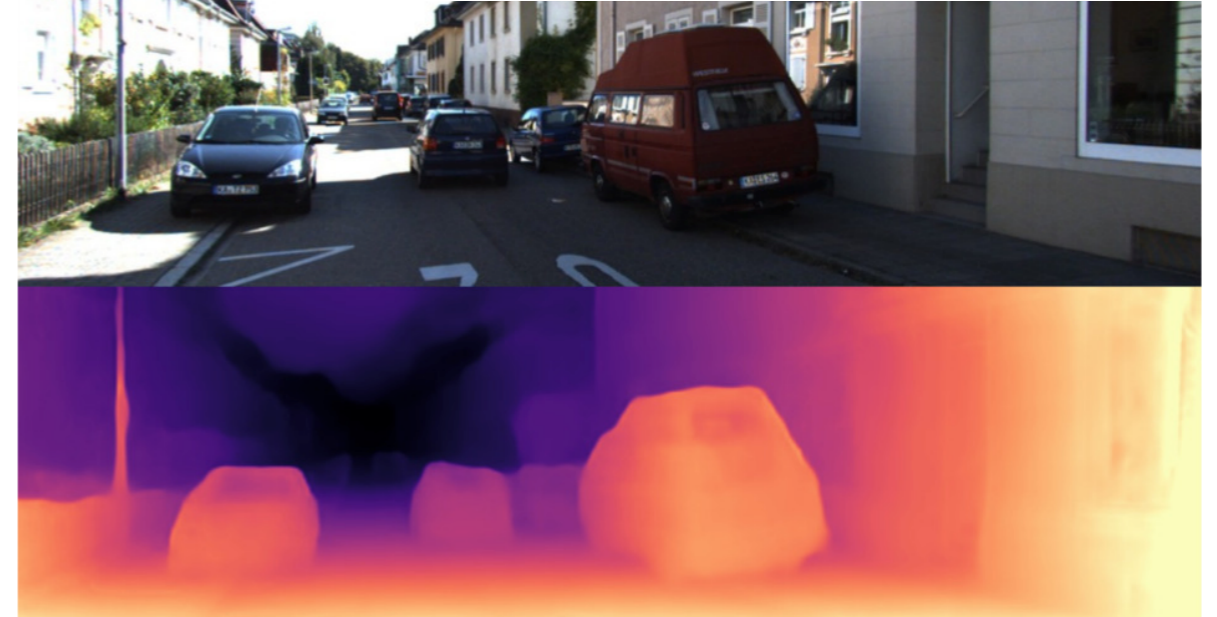
Stereo Depth Estimation



Instance Segmentation



Monocular Depth Estimation



- ◆ Computer Vision Methods (BE4M33MPV, Spring)
 - Lectures 1, 2: overview of vision architectures, examples
 - Lecture 9: deep retrieval

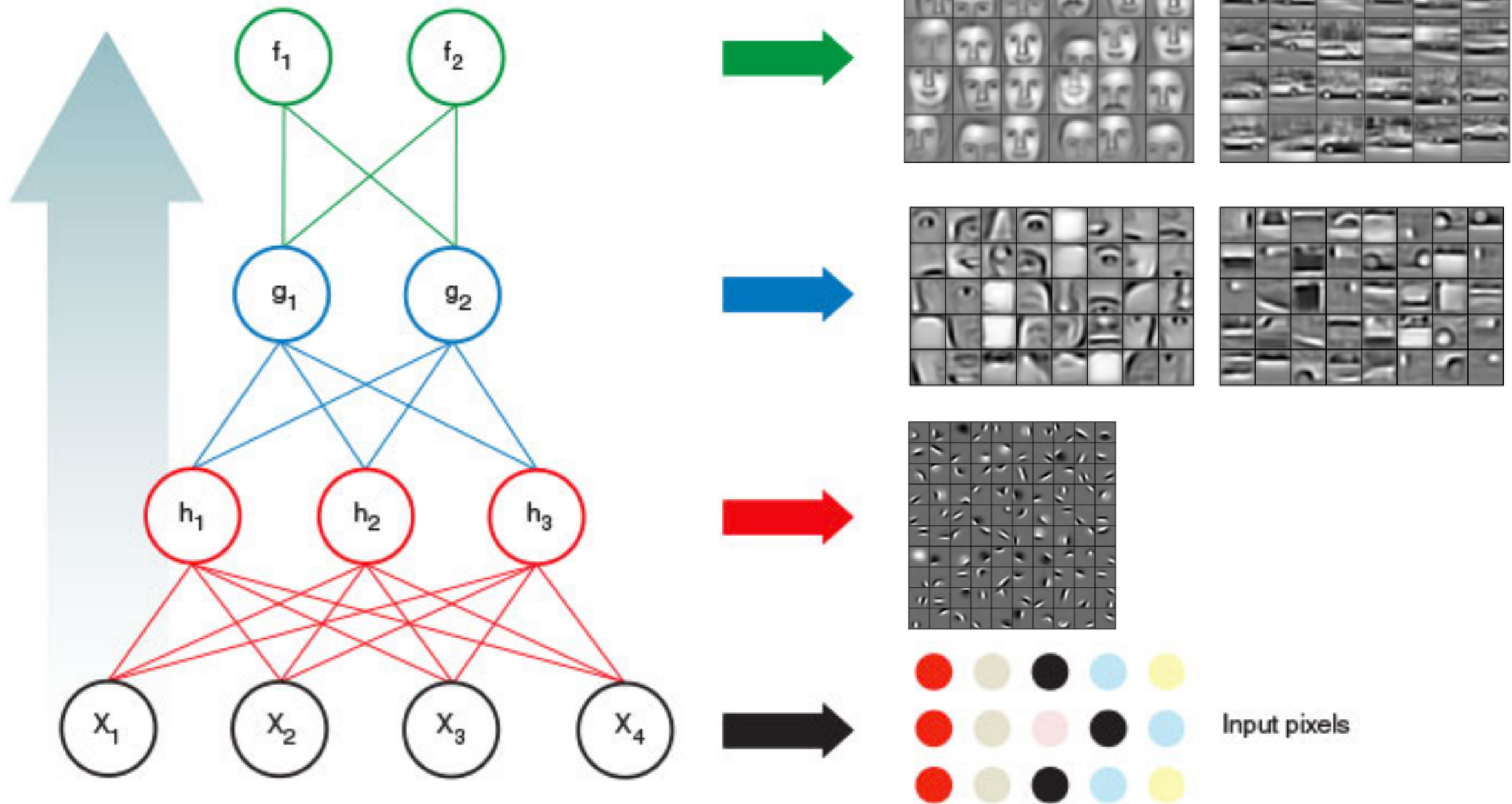
- ◆ Vision for Robotics (B3B33VIR, Fall)
 - Lecture 6,8: (architectures)
 - Lecture 7: self-supervision, weak supervision
 - Lecture 9: Convolutions in 1D, 2D, 3D, graphs
 - Lecture 10, 11: Deep reinforcement learning
 - Lecture 12: Generative adversarial networks

Does the Analogy with Human Vision Extend Further?

Hierarchy of Parts Phenomenon

◆ In networks trained for different complex problems

- some intermediate layers activations correspond object parts



Hierarchy of Parts Phenomenon

◆ In networks trained for different complex problems

- some intermediate layers activations correspond object parts

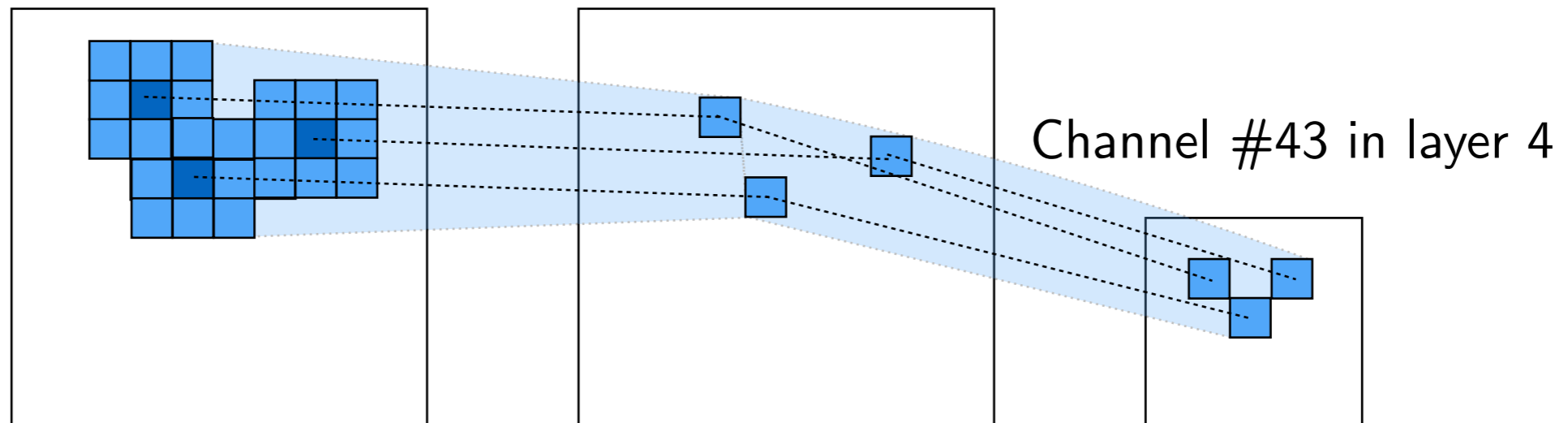
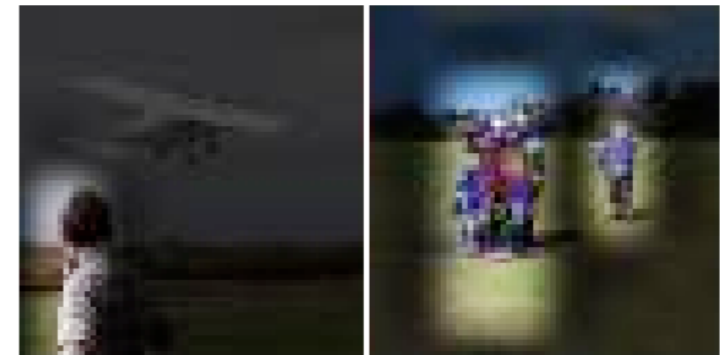
lamps in places net



wheels in object net

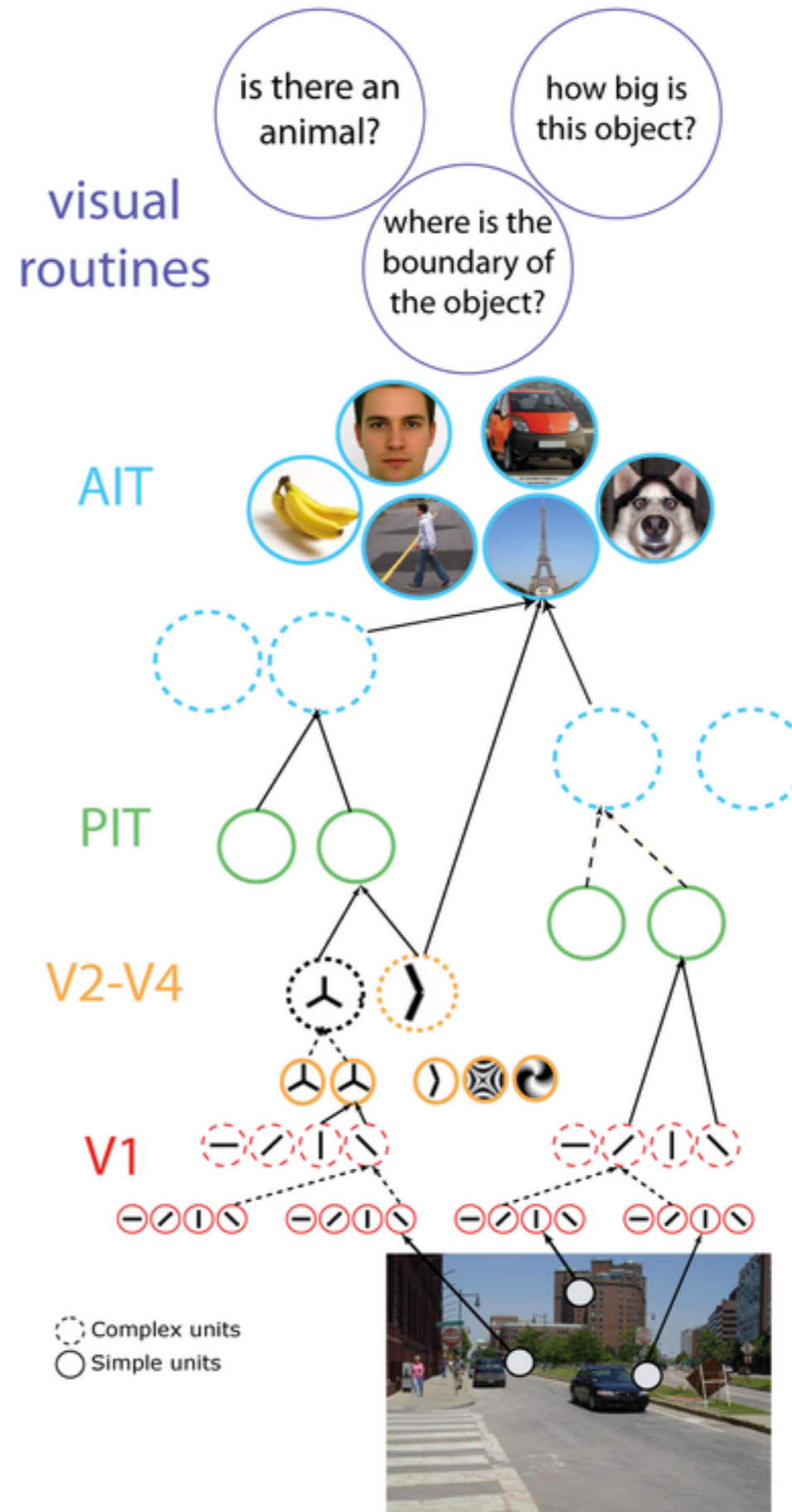


people in video net

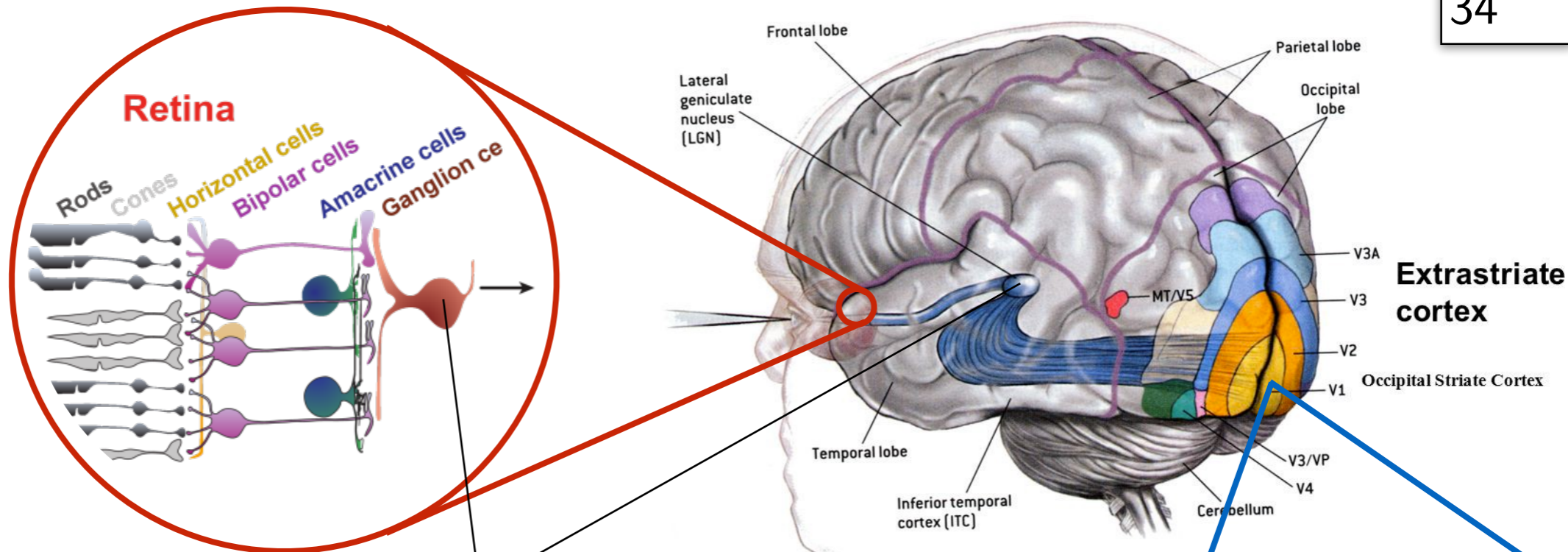


Computational Model of the Brain

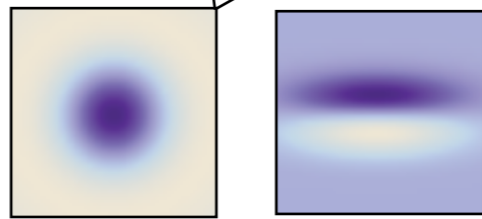
- ◆ Complex tasks build upon the capabilities of simpler tasks



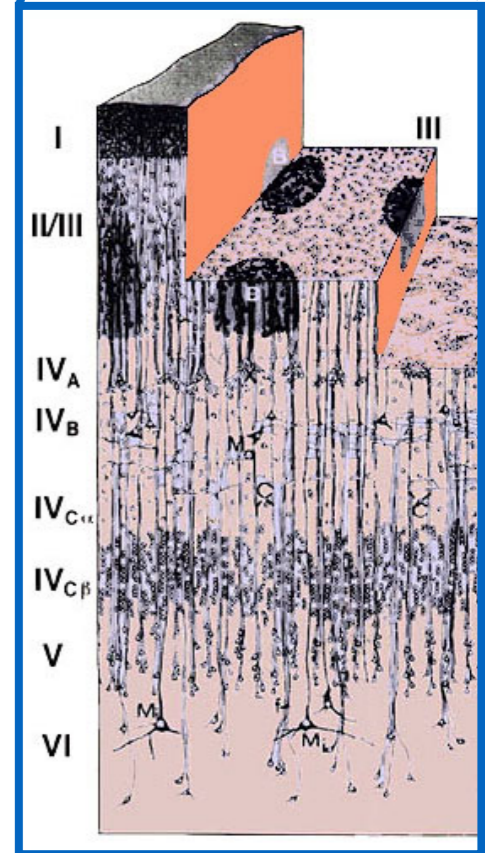
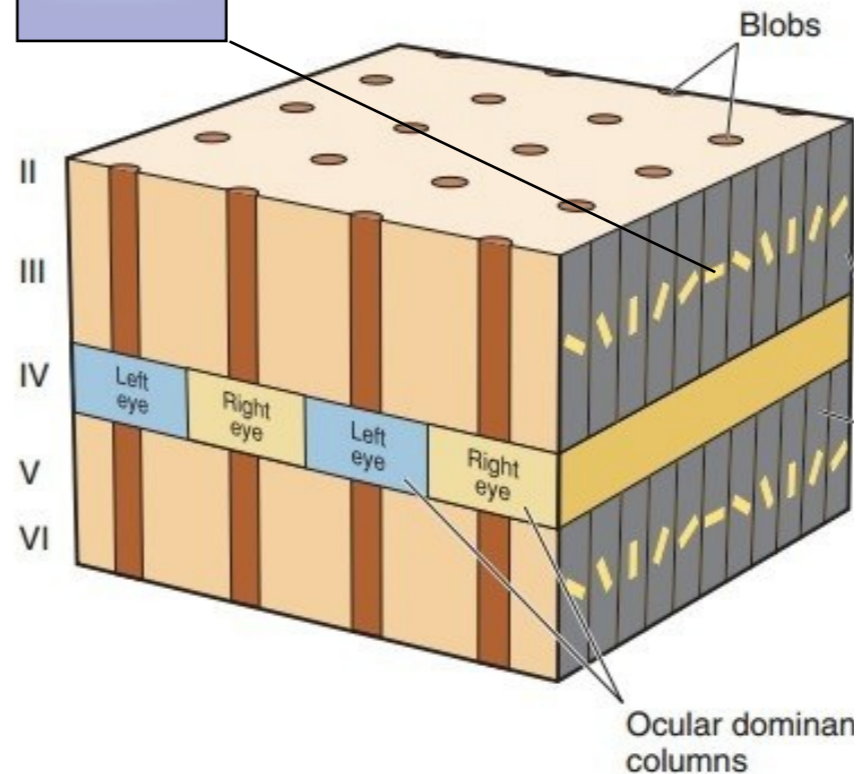
Parallels with Visual Cortex



- ◆ LGN:
 - no orientation preference
 - space-time separable



- ◆ V1 packing in 2D problem:
 - location in the view (retinotopy)
 - orientation
 - ocular dominance
 - motion
- ◆ feedback connections



50000 neurons / mm³