Deep Learning (BEV033DLE)
Lecture 5
Convolutional Neural Networks

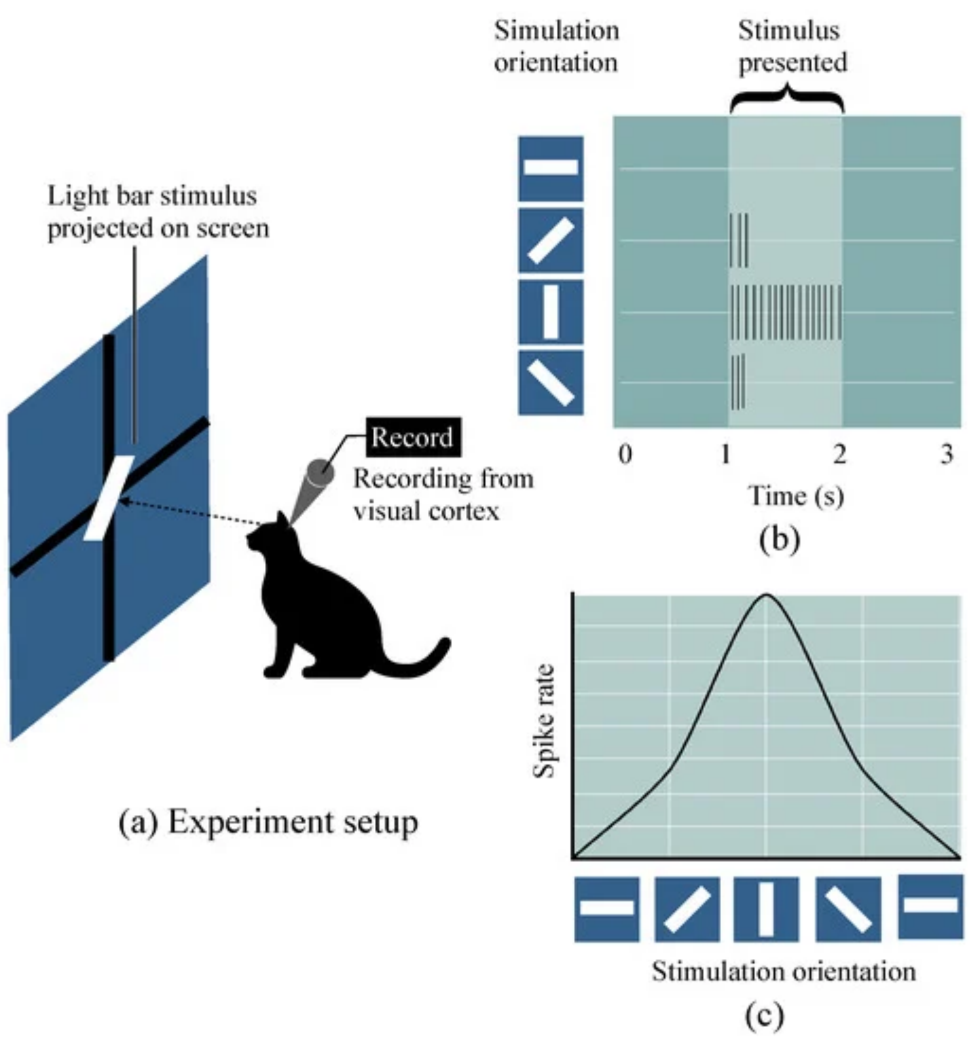
Czech Technical University in Prague

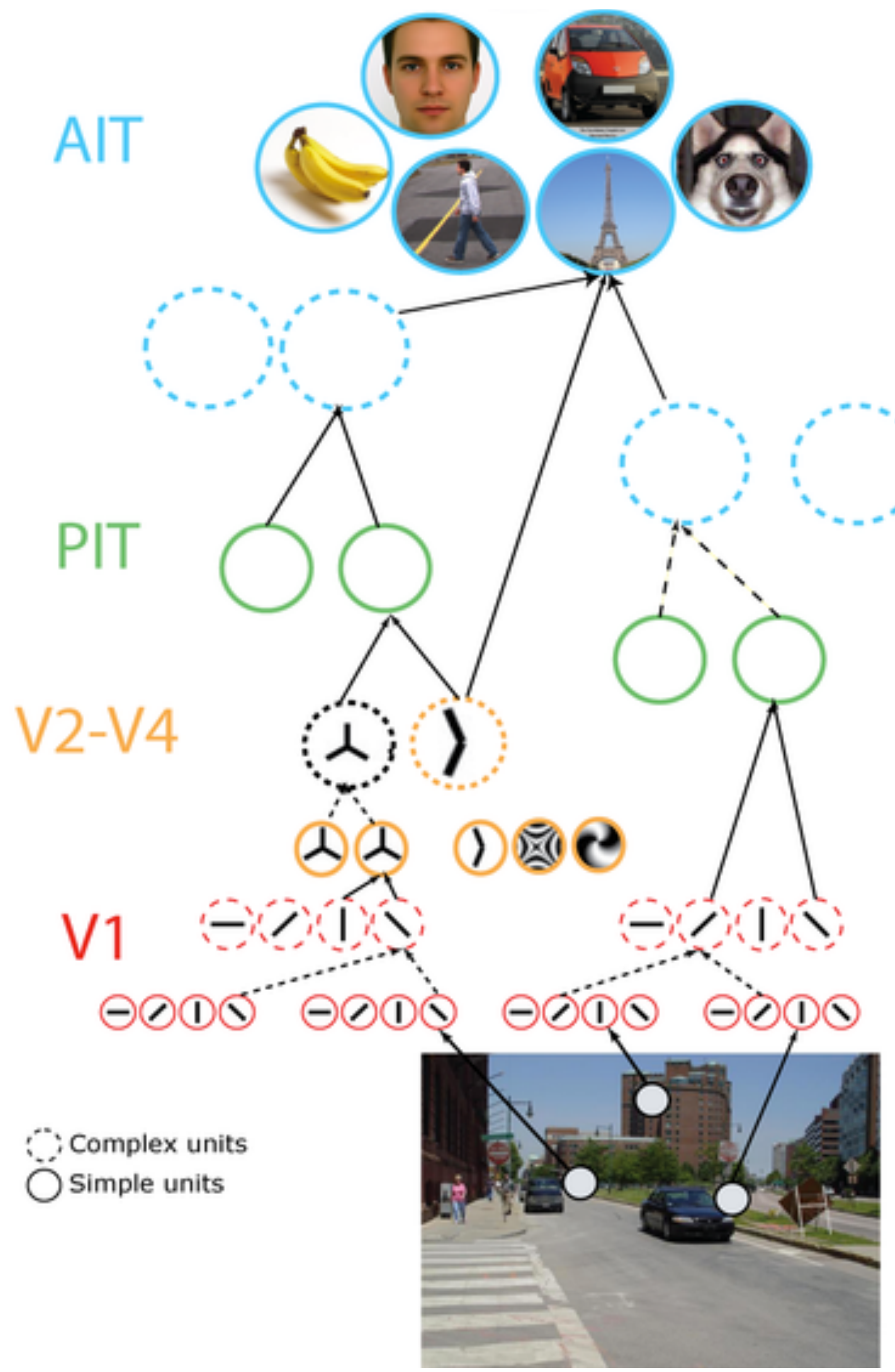
- Overview and Rationales of CNN architecture design

Inspiration from Neuroscience

- Hube and Wiesel (1959): Receptive fields of single neurones in the cat's striate cortex

- the organisation appeared to be hierarchical: responses of '*simple cells*' were aggregated by '*complex cells*,'

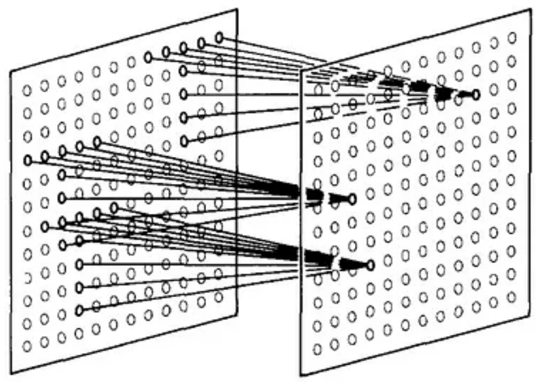


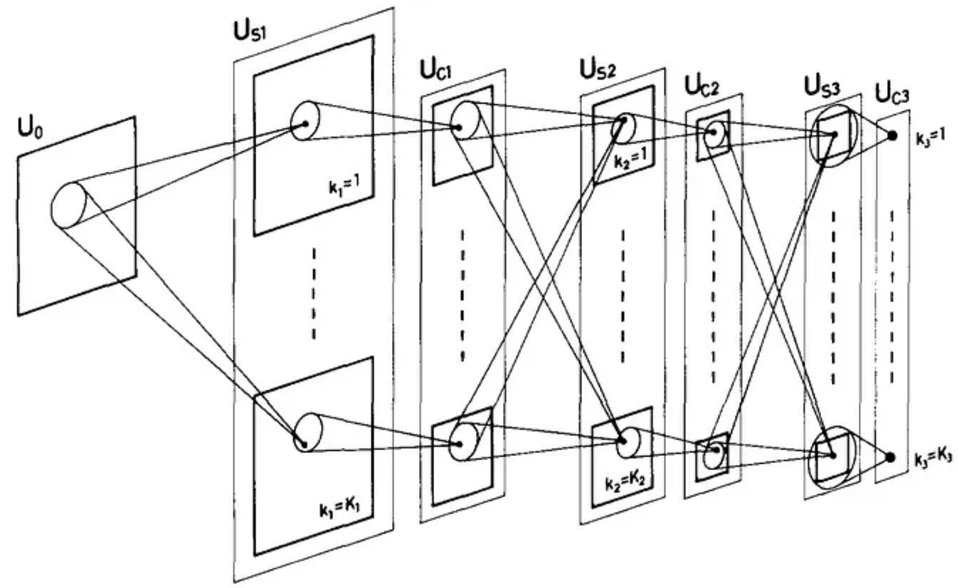


Inspiration e.g. for SIFT descriptors

Neocognitron

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





K. Fukushima

Inspired by neuroscience (Hubel and Wiesel's observation of response to local patterns and idea of hierarchical organization, excitation-inhibition mechanism)

A 7-layer network! Local receptive files with shared weights, pooling layers, ReLU activations.

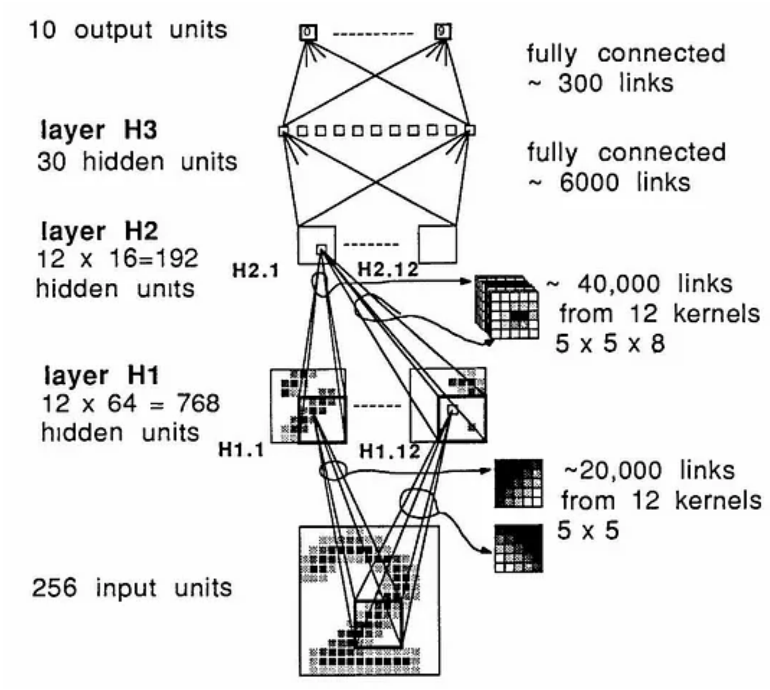
Trained in an unsupervised manner...

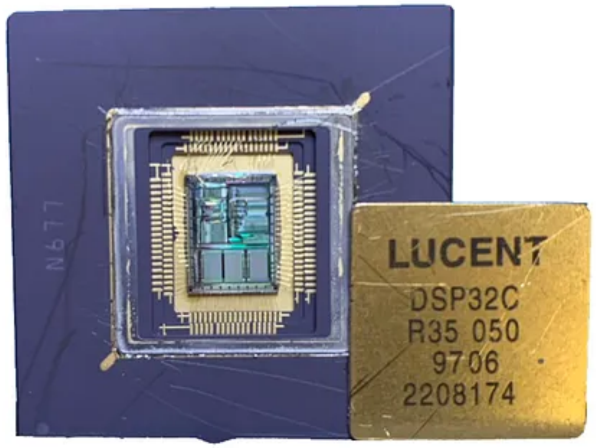
By using local receptive fields at each level we achieve more flexibility to geometric variations

Convolutional Neural Networks

LeNet 3 (1989)

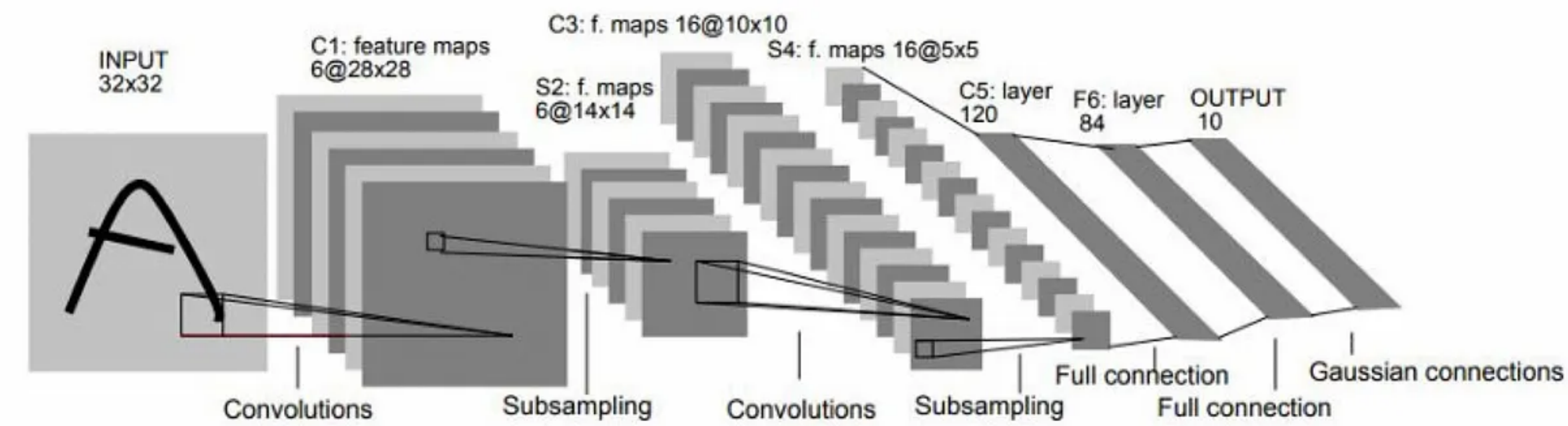
Linear filters, backprop





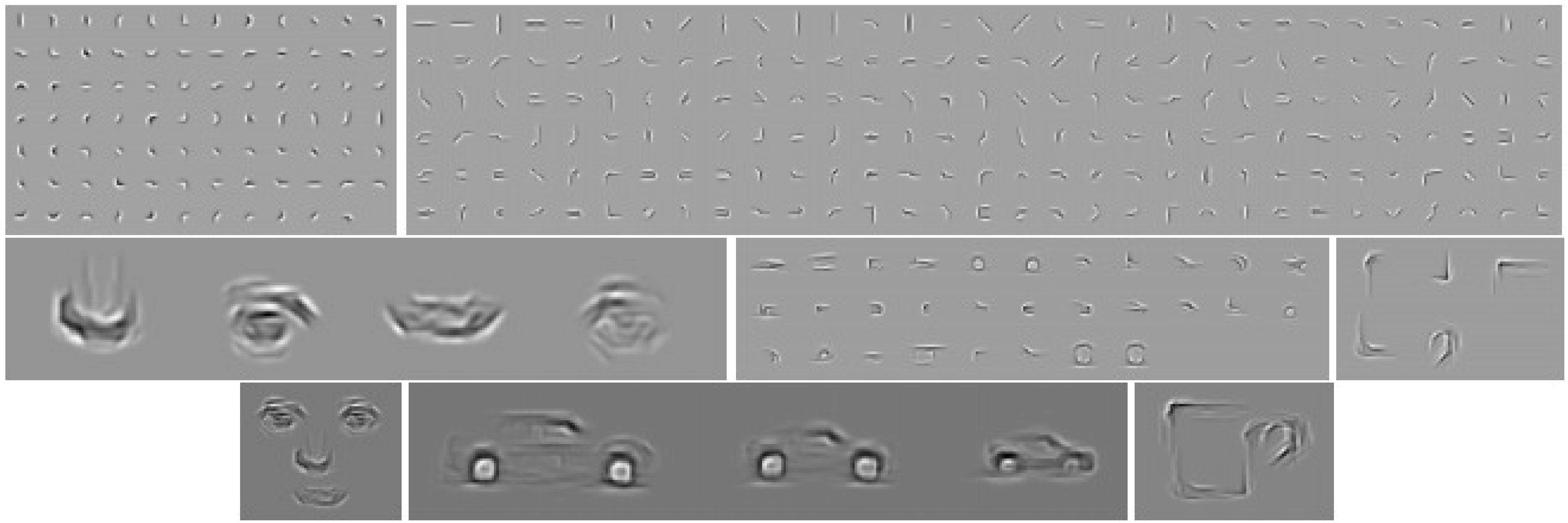
Y. LeCun

LeNet 5 (1998)

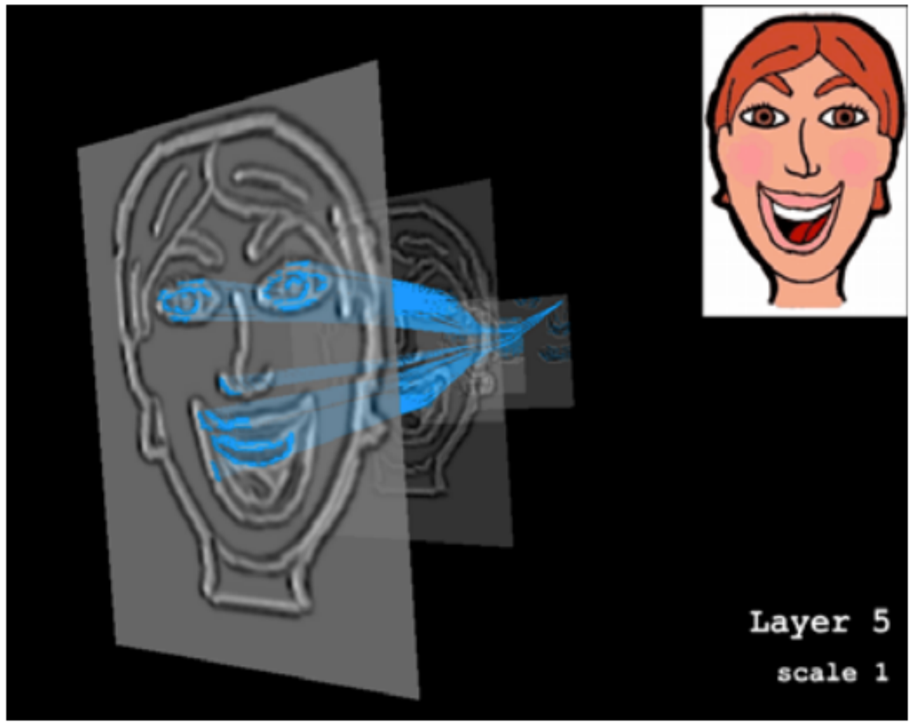


Non-NN Hierarchy of Parts Models

- [Fidler and Leonardis (2007): "Towards Scalable Representations of Object Categories: Learning a Hierarchy of Parts"]



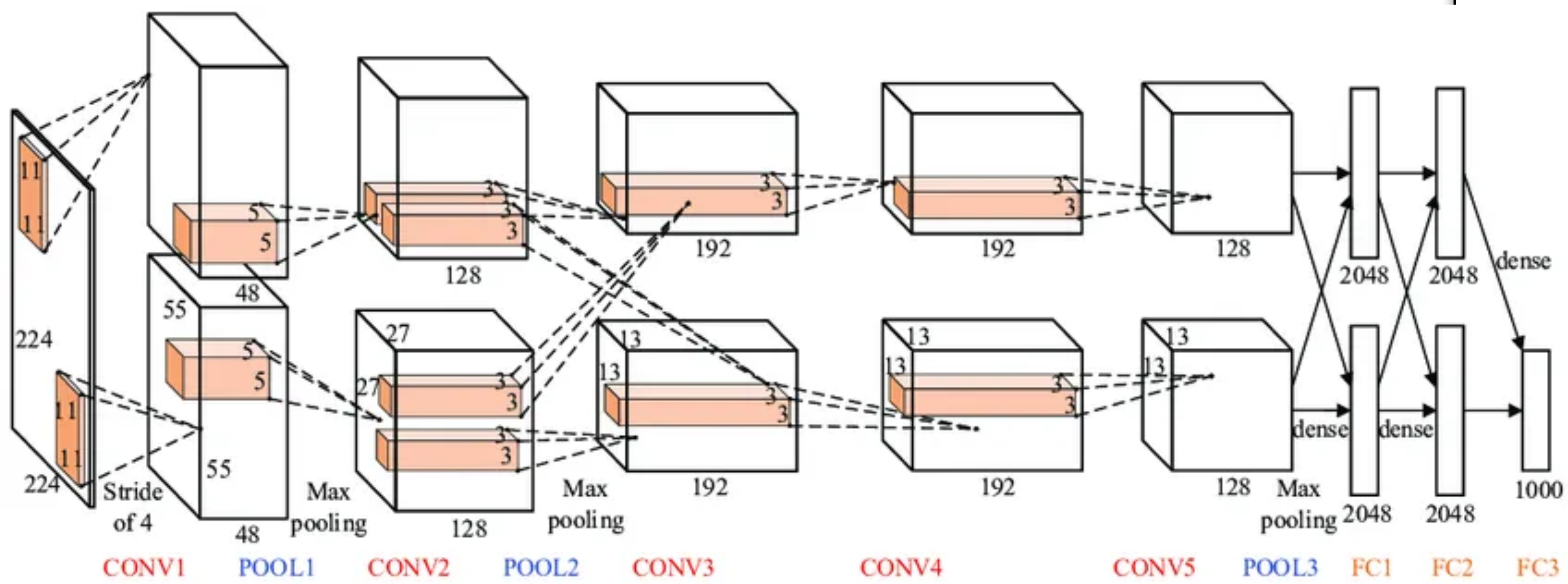
Learning layer-by-layer based on statistics and selection



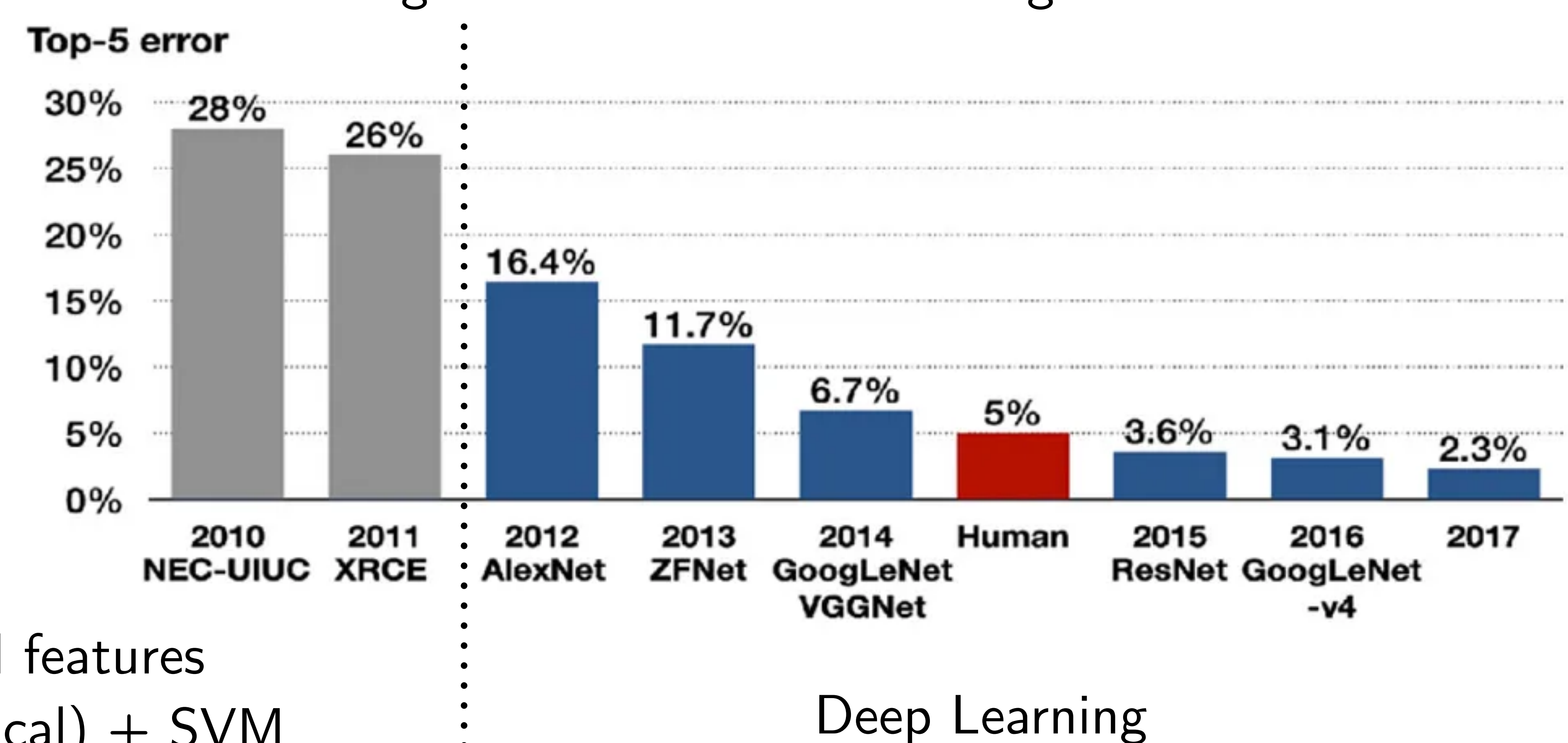
Convolutional Neural Networks

AlexNet (2012)

More data & weights, GPU



ImageNet classification challange

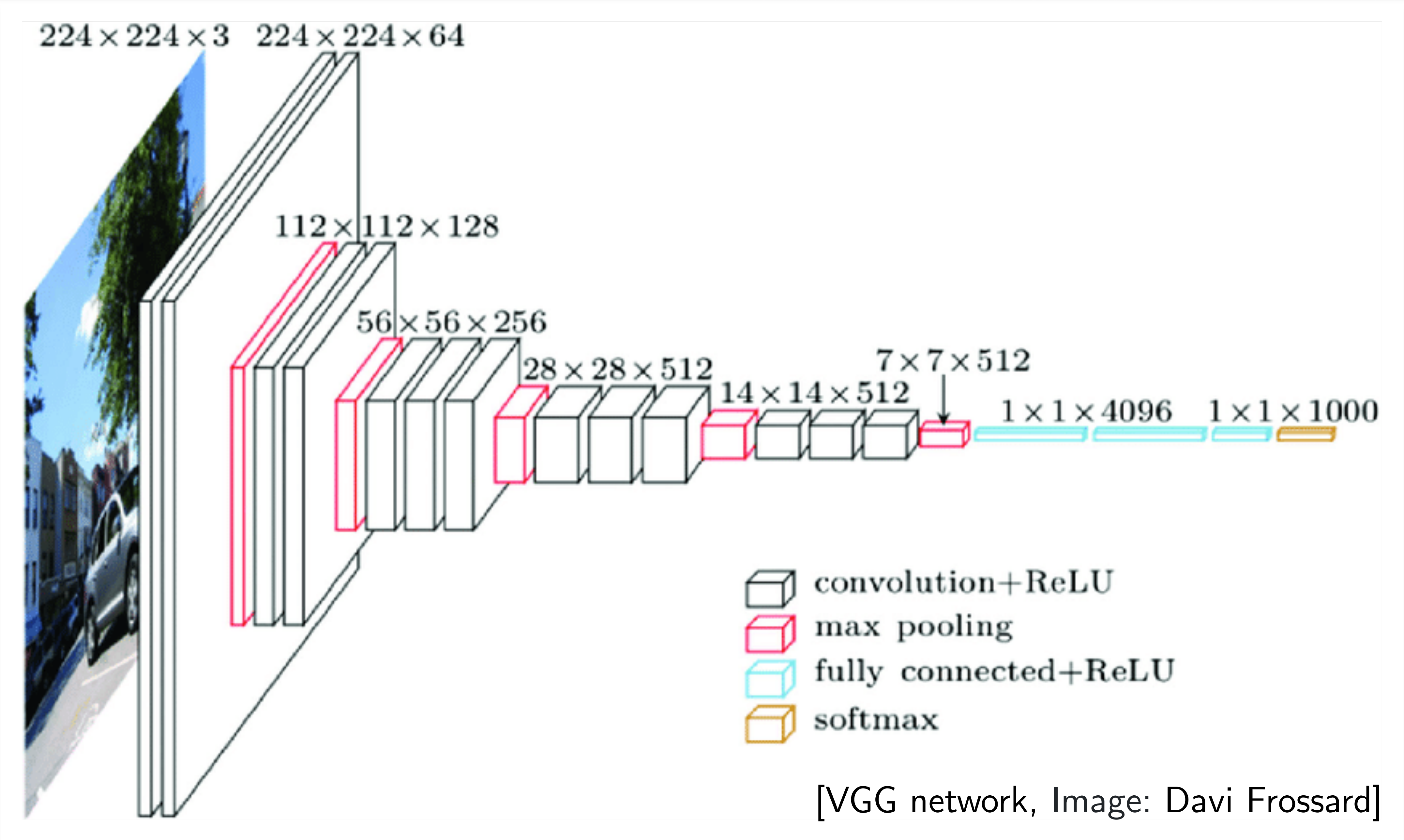


Engineered features (incl. hierarchical) + SVM

Deep Learning

VGG

[Simonyan and Zisserman (2014): Very Deep Convolutional Networks for Large-Scale Image Recognition]



[VGG network, Image: Davi Frossard]

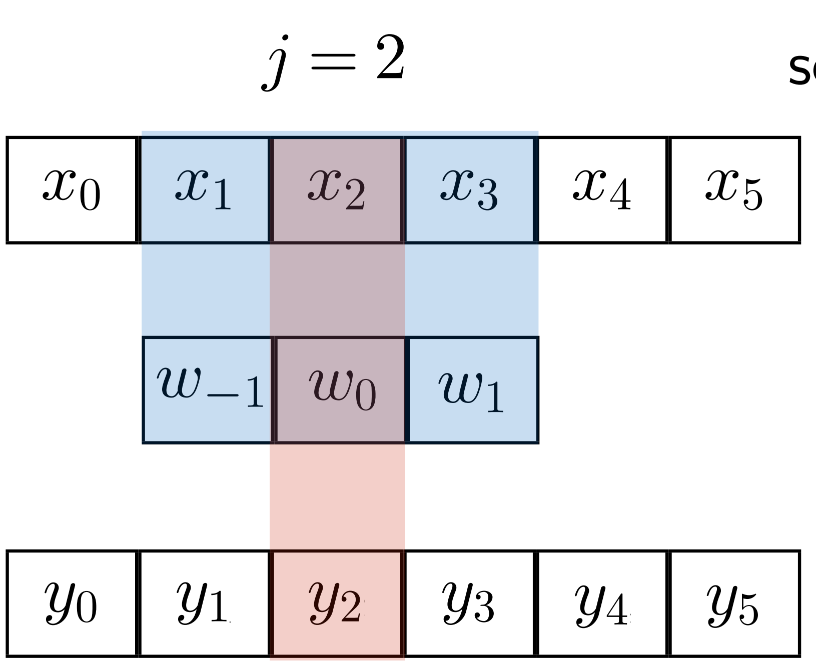
- Goal: understand building blocks and design principles

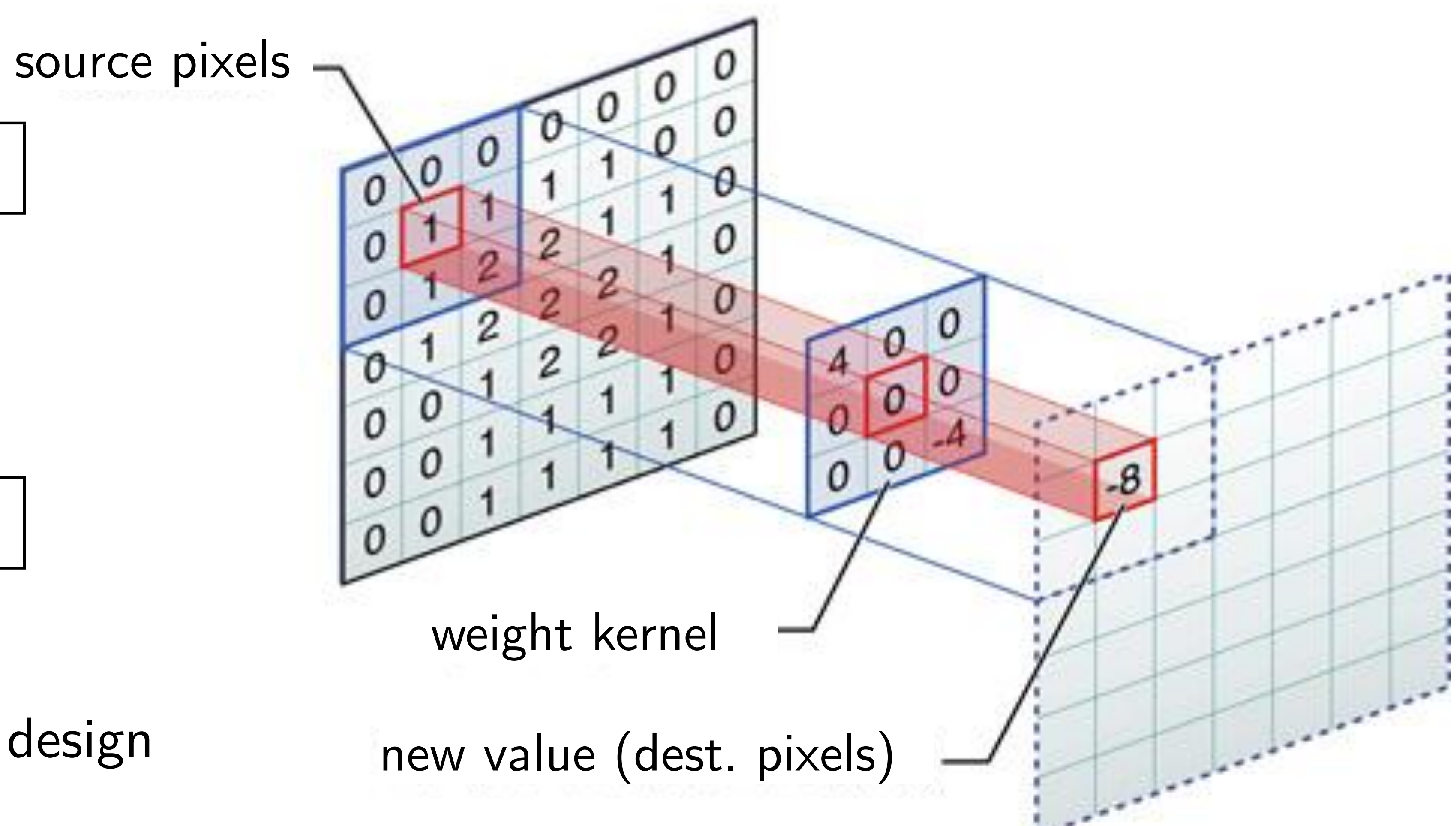
Convolution and Cross-Correlation

- Convolution and Correlation (1D)
  - Convolution $y = w * x$: $y_j = \sum_{k=-h}^{h} w_k x_{j-k} = \sum_{k=-h}^{h} w_{-k} x_{j+k}$ (flip of the weight matrix)
  - Cross-correlation $y = w \star x$: $y_j = \sum_{k=-h}^{h} w_k x_{j+k}$

output (y_j), weight kernel (w_k), input (x_{j+k})

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

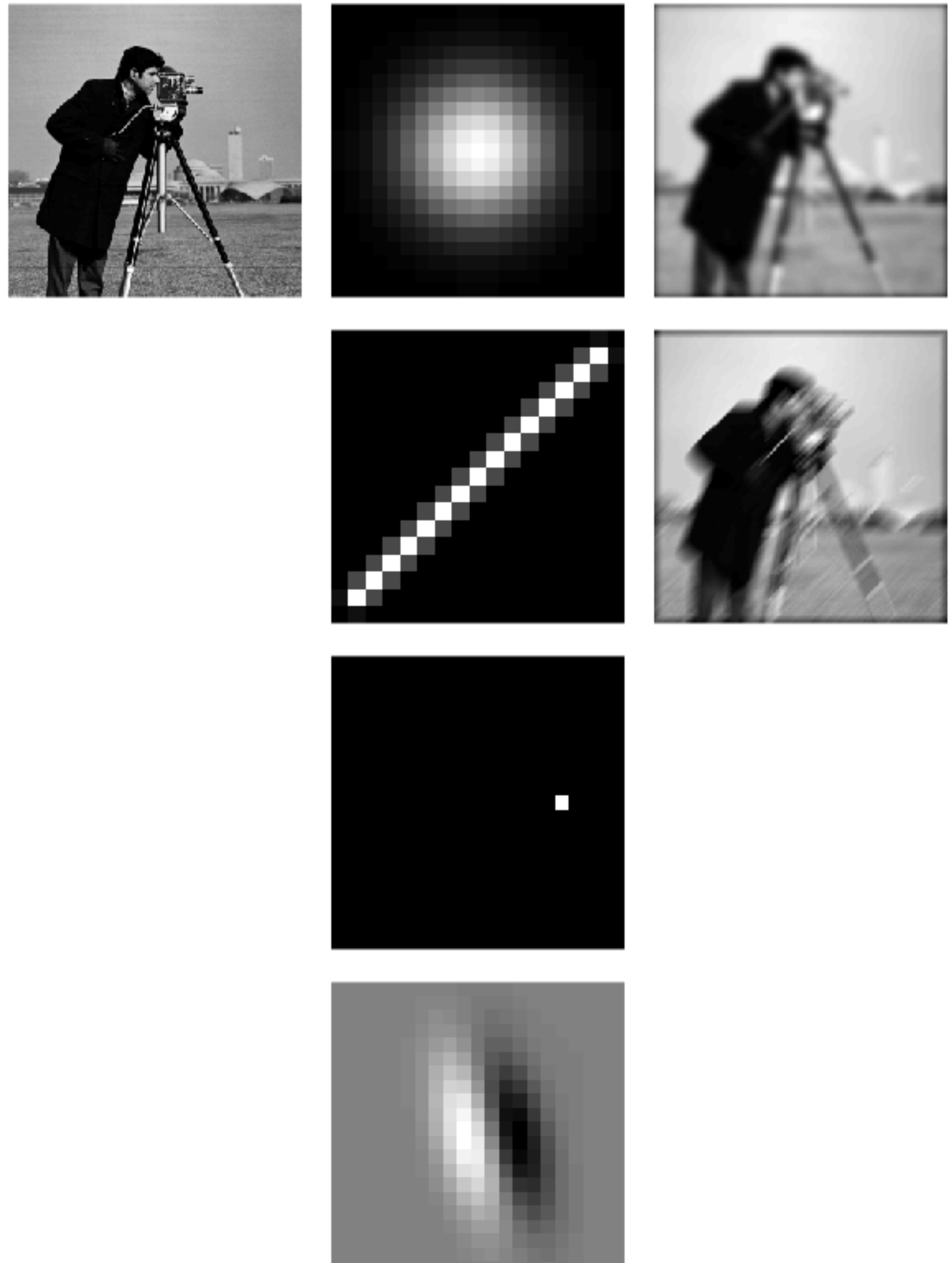




- Translation equivariance by design

Examples (Cross-Correlation)

Input Kernel Output

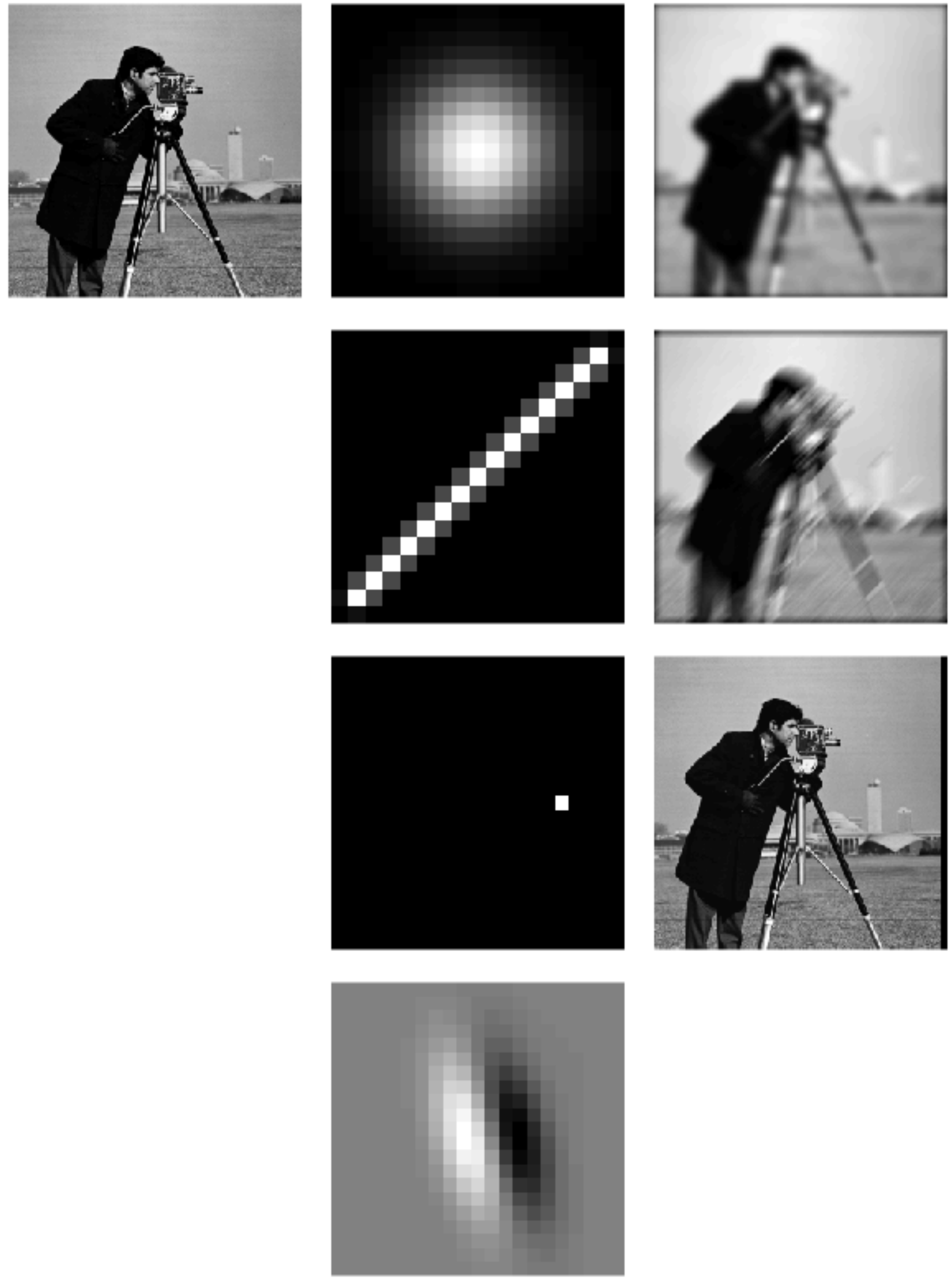


Blur

Motion Blur

Examples (Cross-Correlation)

Input | Kernel | Output

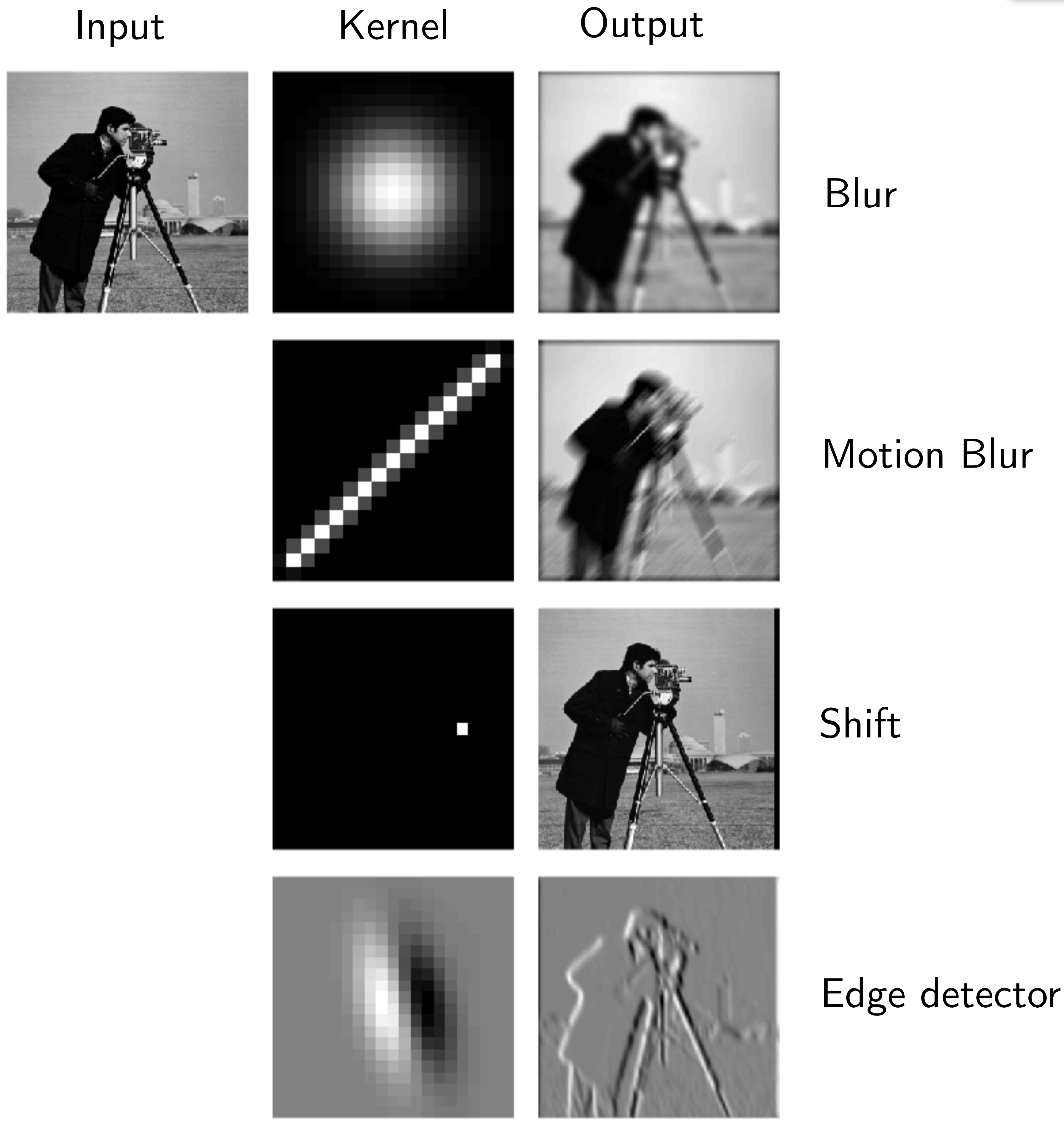


Blur

Motion Blur

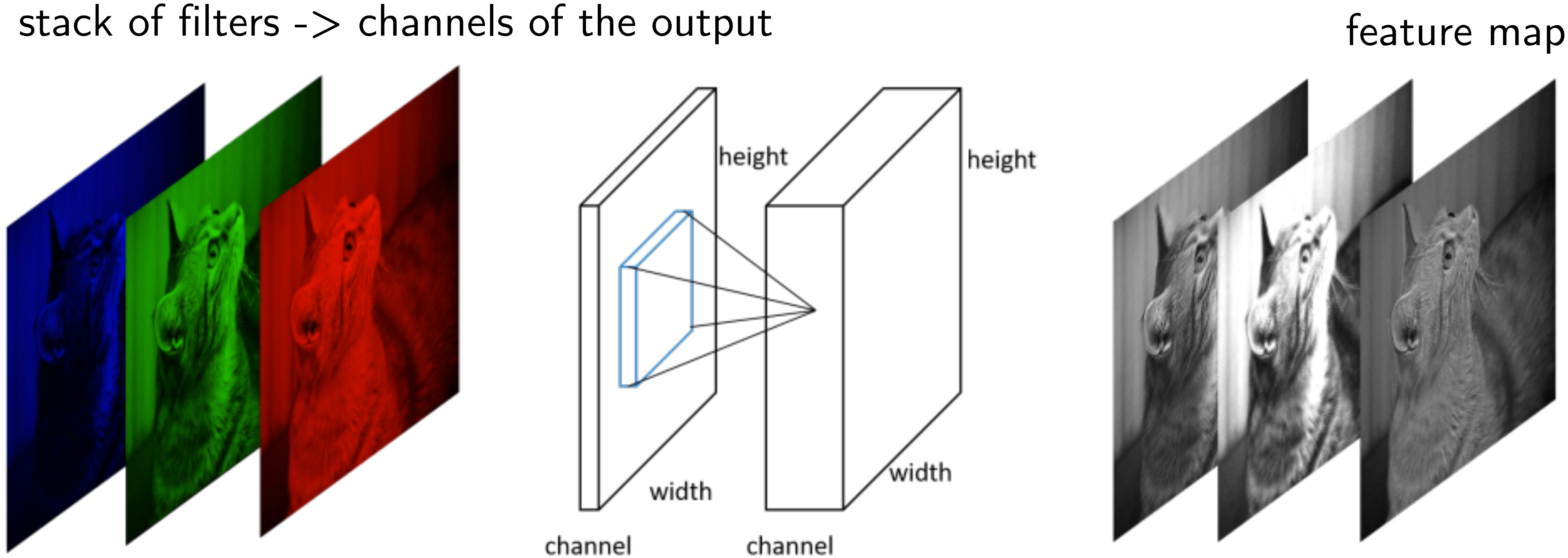
Shift

Examples (Cross-Correlation)



Multi-Channel Convolution

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



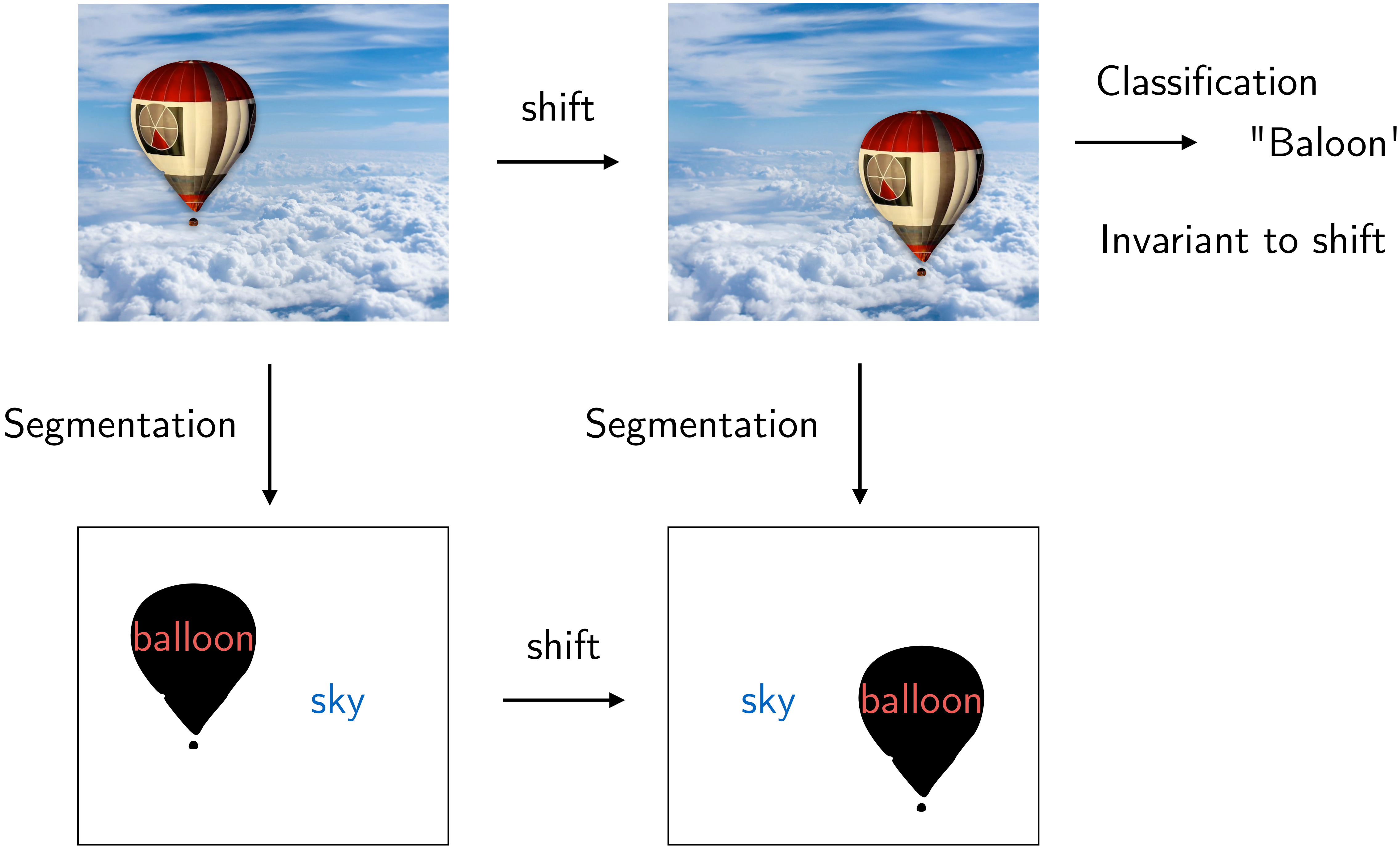
- Multi-channel cross-correlation:

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

input channel (c) — filter spatial dimensions (Δi, Δj) — output channel (o)

- 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

Invariance and Equivariance



Equivariant to shift (commutes with shift)

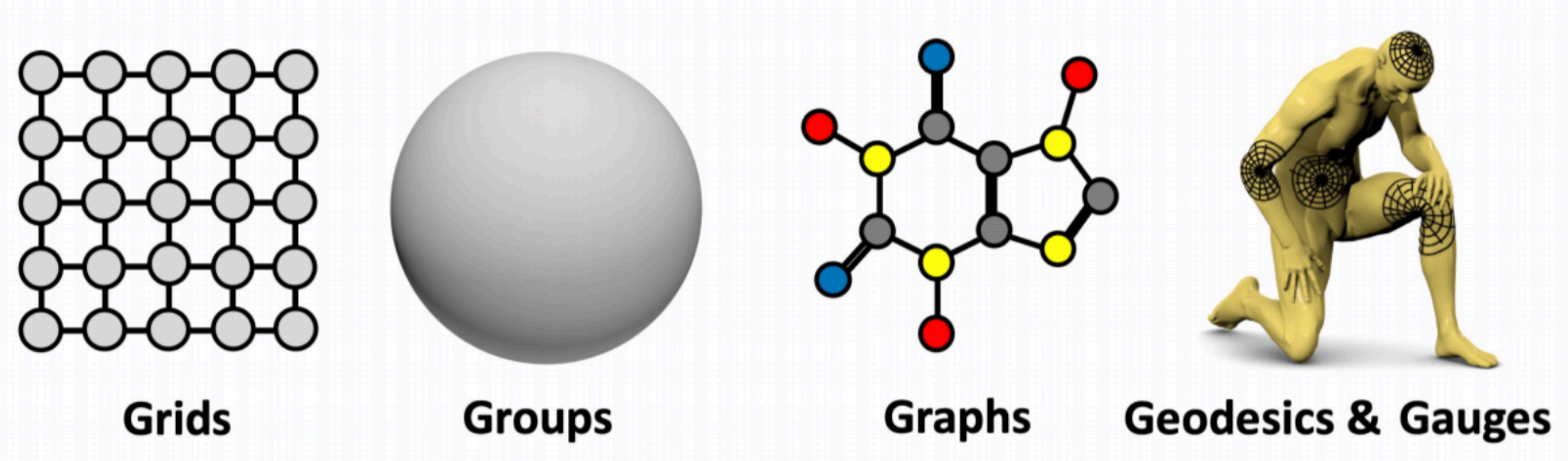
Would be hard to achieve if the image was given as a general vector — we are using 2D grid structure and require that all locations are treated equally

Geometric Deep Learning

[M. Bronstein et al.: "Geometric Deep Learning"]

✦ Concept: systematization of geometries as study of equivariances

- Apply this principle to systematize the zoo of NN architectures



✦ Key message: neural networks for processing geometric data should respect the structure of the domain

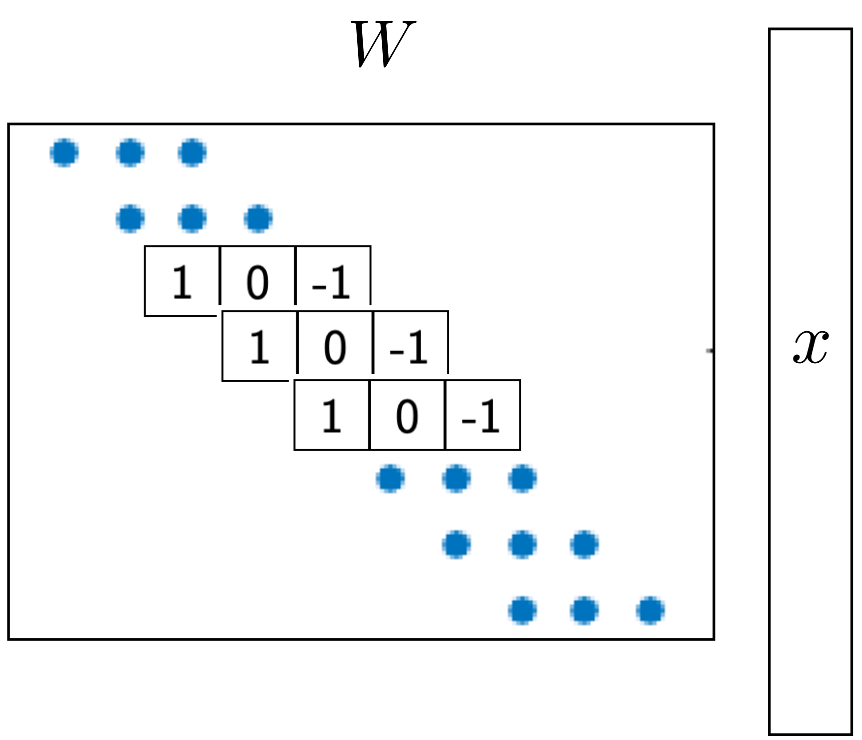
Convolution as a Linear Operator

- Convolution:
  - $y_j = \sum_{k=-h}^{h} w_k x_{j-k}$
- As matrix-vector product:
  - Denote: $i = j - k$, then $k = j - i$ and $y_j = \sum_i w_{j-i} x_i$
  - Denote $W_{i,j} = w_{i-j}$
  - Then $y = Wx$

- Convolution is a linear transform of a special structure:

Example: $w_k = k$

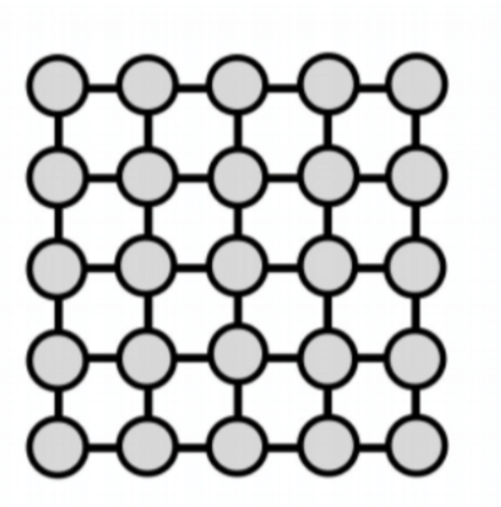
-1	0	1



- Cross-correlation has flipped w, resulting in transposed W
  - Backprop of convolution is cross-correlation and vice-versa

Convolution from Equivariance

Grid

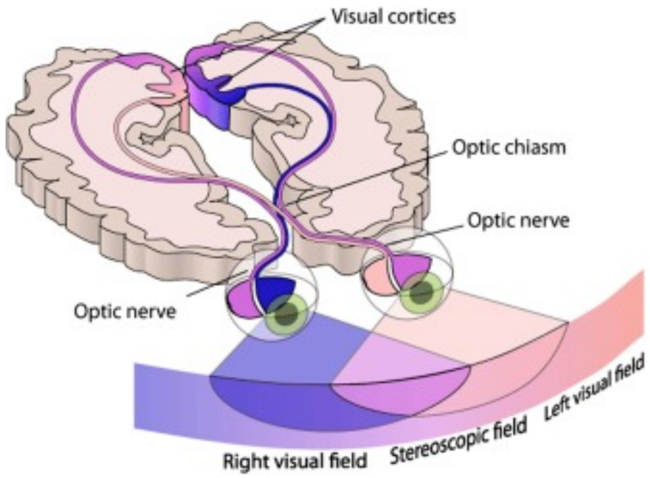


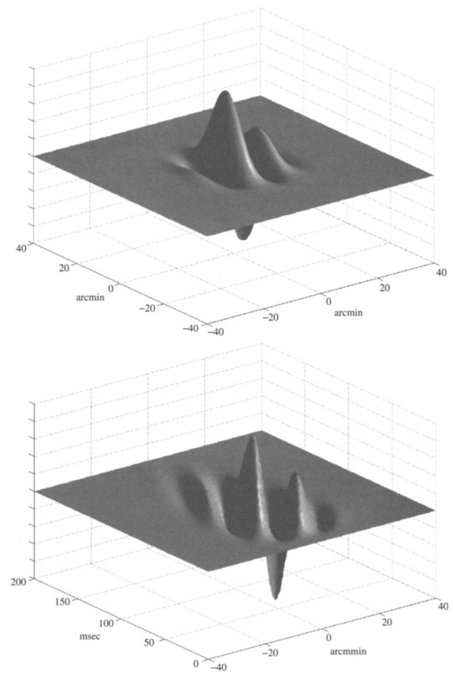
Shift operator (1D)

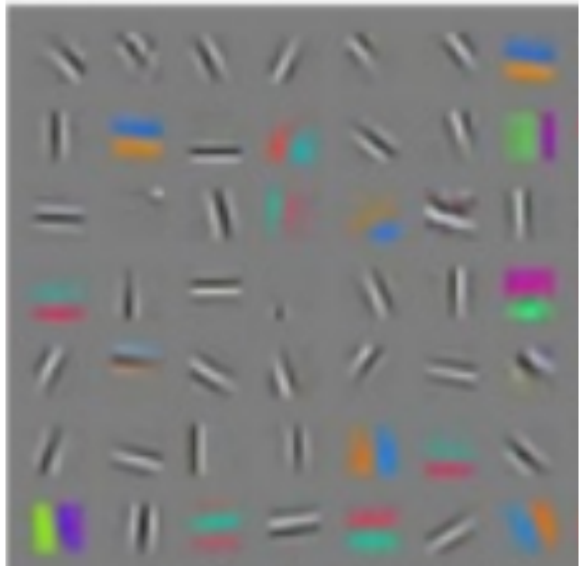
- Characterize linear transforms which are equivariant to shift:
  - let $S: \mathbb{R}^n \to \mathbb{R}^n$ be a shift matrix: $S_{i,j} = [\![j = i+1]\!]$
  - let $A: \mathbb{R}^n \to \mathbb{R}^n$
  - equivariance: $ASx = SAx$ for all x
  - implies $AS = SA$
  - implies $A_{i,j+1} = A_{i+1,j}$ for all i, j – "circulant" matrix (convolution)
- **Conclusion**: a matrix is circulant if and only if it commutes with shift
- Further properties:
  - Matrices satisfying $AS = SA$ will have same eigenvectors
  - Eigenvectors of shift S are the Fourier basis functions Φ
  - All convolutions can be represented as $A = \Phi \Lambda(w) \Phi^\mathsf{T}$, where $\Lambda(w) = \mathrm{diag}(\Phi^\mathsf{T} w)$
  - Convolution Theorem: $Ax = \Phi\Big((\Phi^\mathsf{T} w) \odot (\Phi^\mathsf{T} x) \Big)$

Learned vs Engineered Filters

✦ Gabor Filters (and generalizations) - mathematical model for V1 cells:



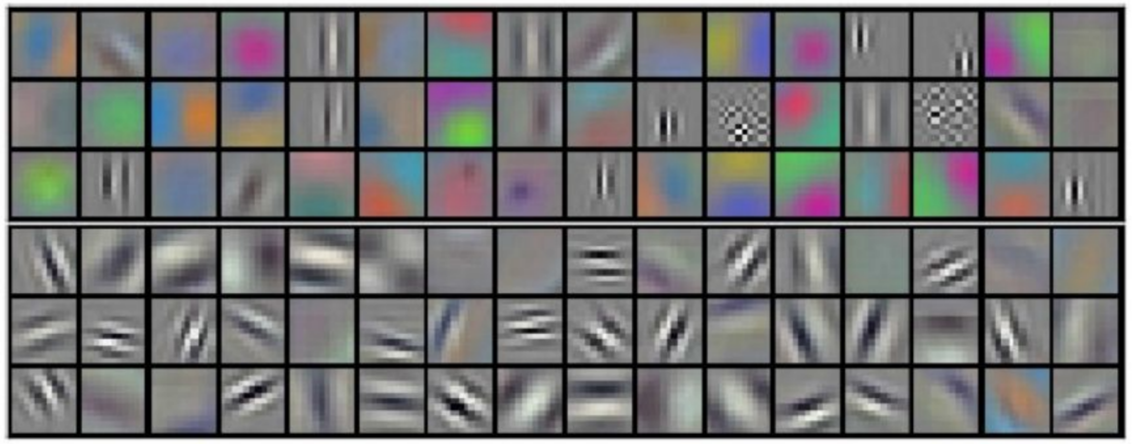




Equivariance to transformations + more design principles w.r.t. scale-space

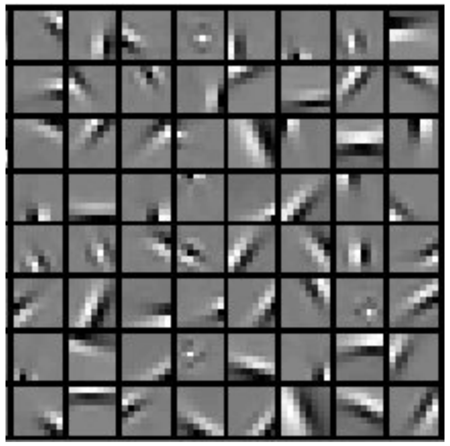
Active research on symmetry as a guiding principle in artificial and brain neural networks

✦ CNN first layer filters (learned)



equivariance + learning

✦ PCA of Image Patches



Data statistics + regularization

Geometric Deep Learning on other Domains

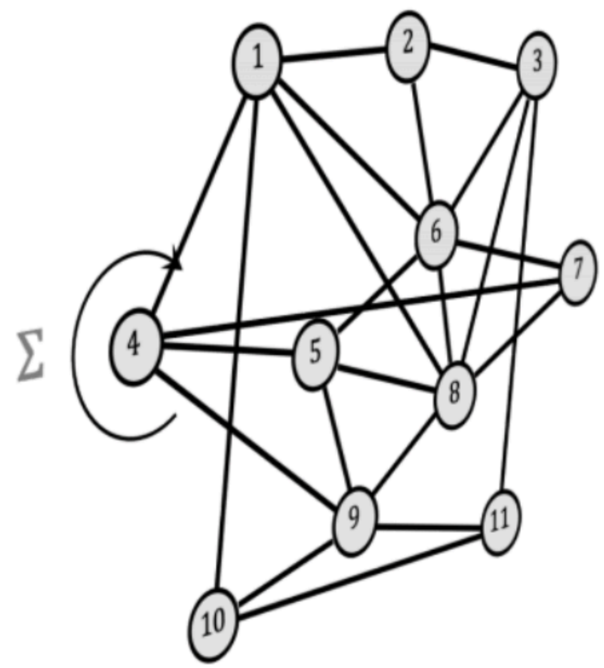
✦ Popular architectures as instances of GDL blueprint

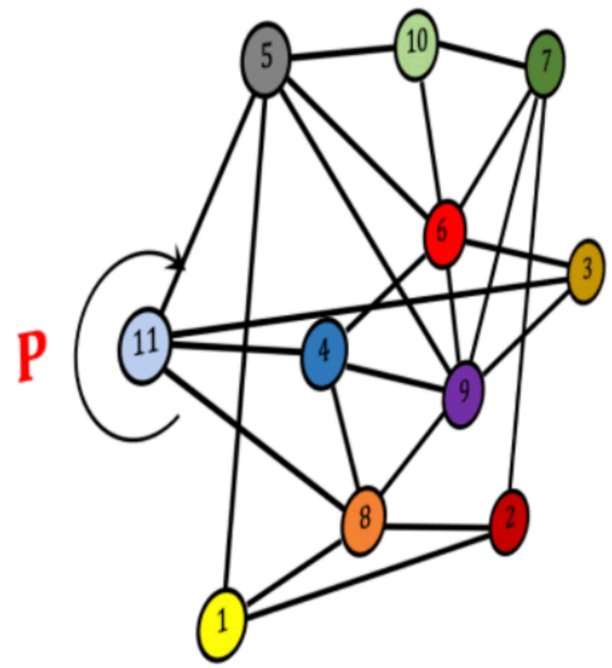
Architecture	**Domain Ω**	**Symmetry Group $\mathfrak{G}$**
CNN	Grid	Translation
Spherical CNN	Sphere / $SO(3)$	Rotation $SO(3)$
Intrinsic / Mesh CNN	Manifold	Isometry $Iso(\Omega)$ / Gauge Symmetry $SO(2)$
GNN	Graph	Permutation Σ_n
Deep Sets	Set	Permutation Σ_n
Transformer	Complete Graph	Permutation Σ_n
LSTM	1D Grid	Time warping

[M. Bronstein et al.: "Geometric Deep Learning"]

Geometric Deep Learning: Graphs

Graph $G = (V, E)$	Node features $\mathcal{X}(G)$	functions $\mathcal{F}(\mathcal{X}(\Omega))$





Permutation group Σ_n

Permutation matrix P

$$\mathbf{PX} = \left(x_{\pi^{-1}(i),j}\right)$$

Message passing

$$\mathbf{F}\left(\mathbf{PX}, \mathbf{PAP}^\top\right) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$$

Convolutional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

Attentional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

Message-passing

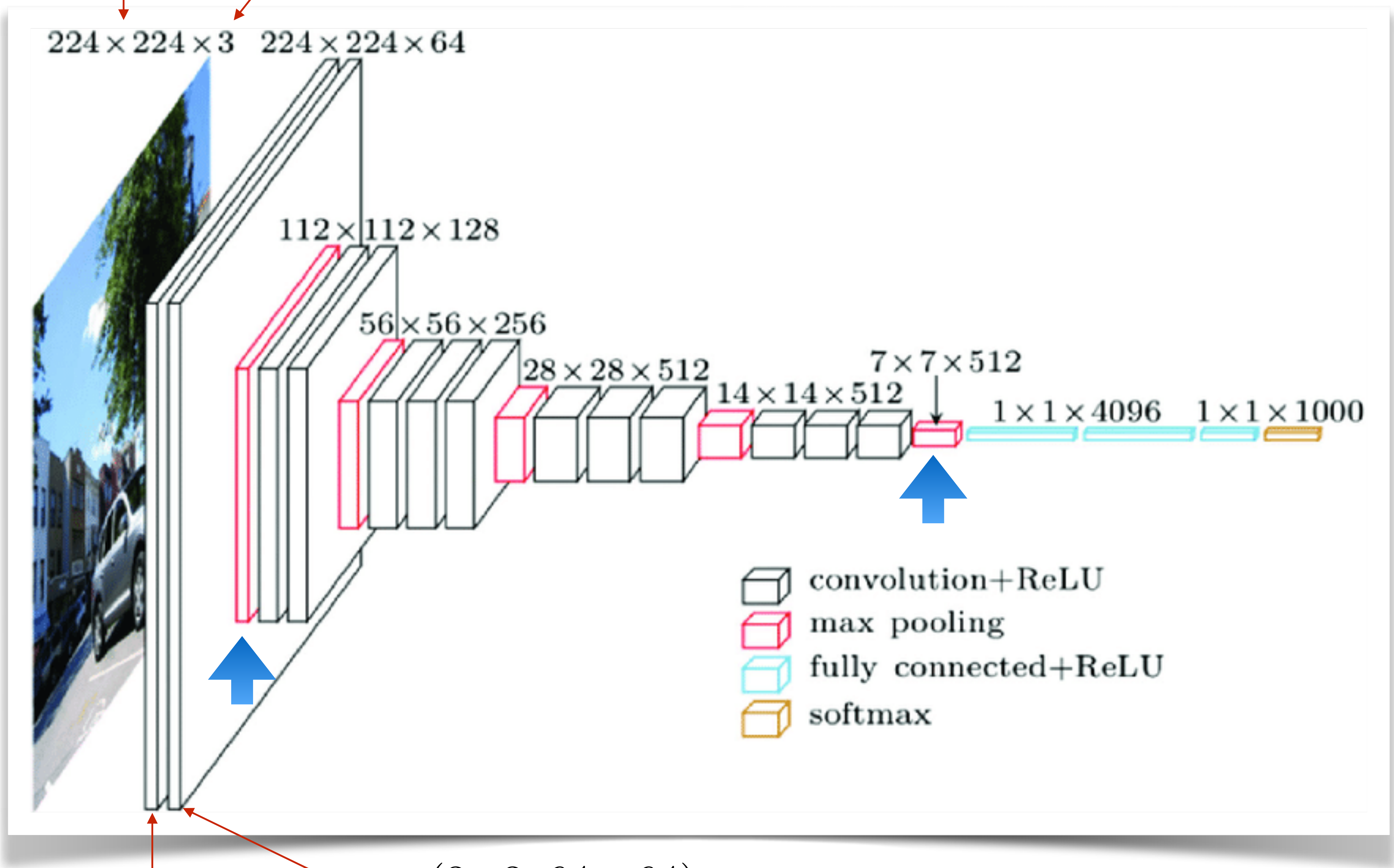
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

[M. Bronstein et al.: "Geometric Deep Learning"]

Pooling

Spatial size of the input image

channels



conv(3×3, 64→64)

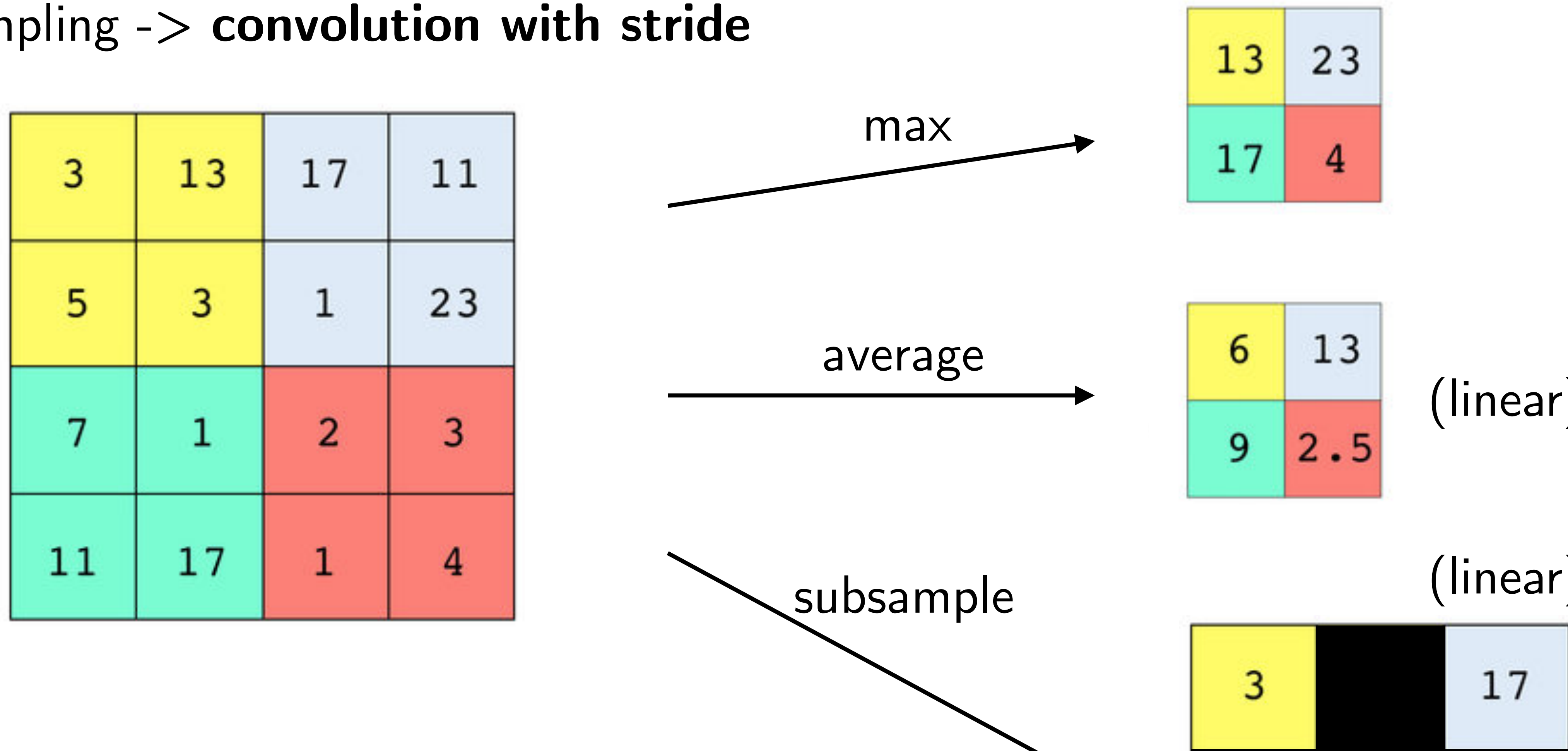
Result of $\mathrm{conv}(K\times K, 3\to 64)$ followed by ReLU

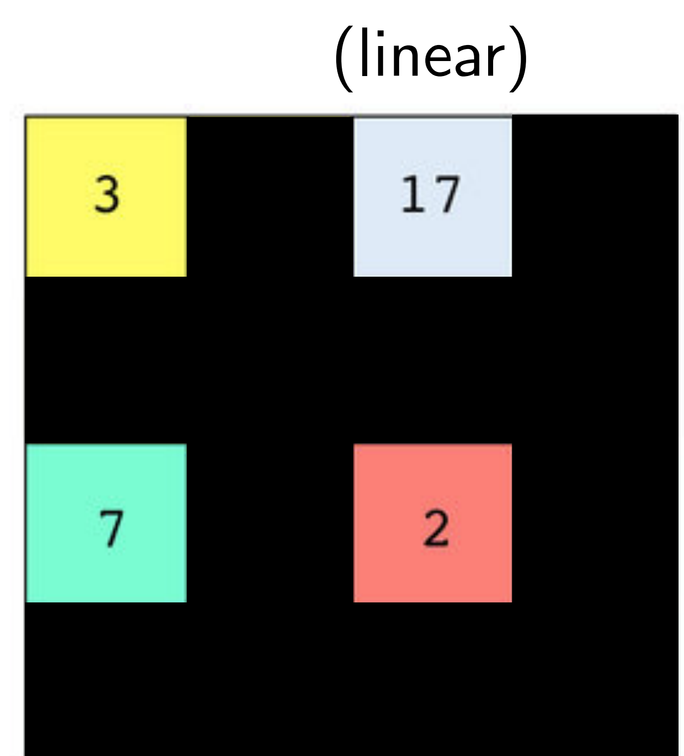
- Eventually want to classify -> need to reduce spatial dimensions

Pooling

✦ Following approaches are used to reduce the spatial resolution:

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



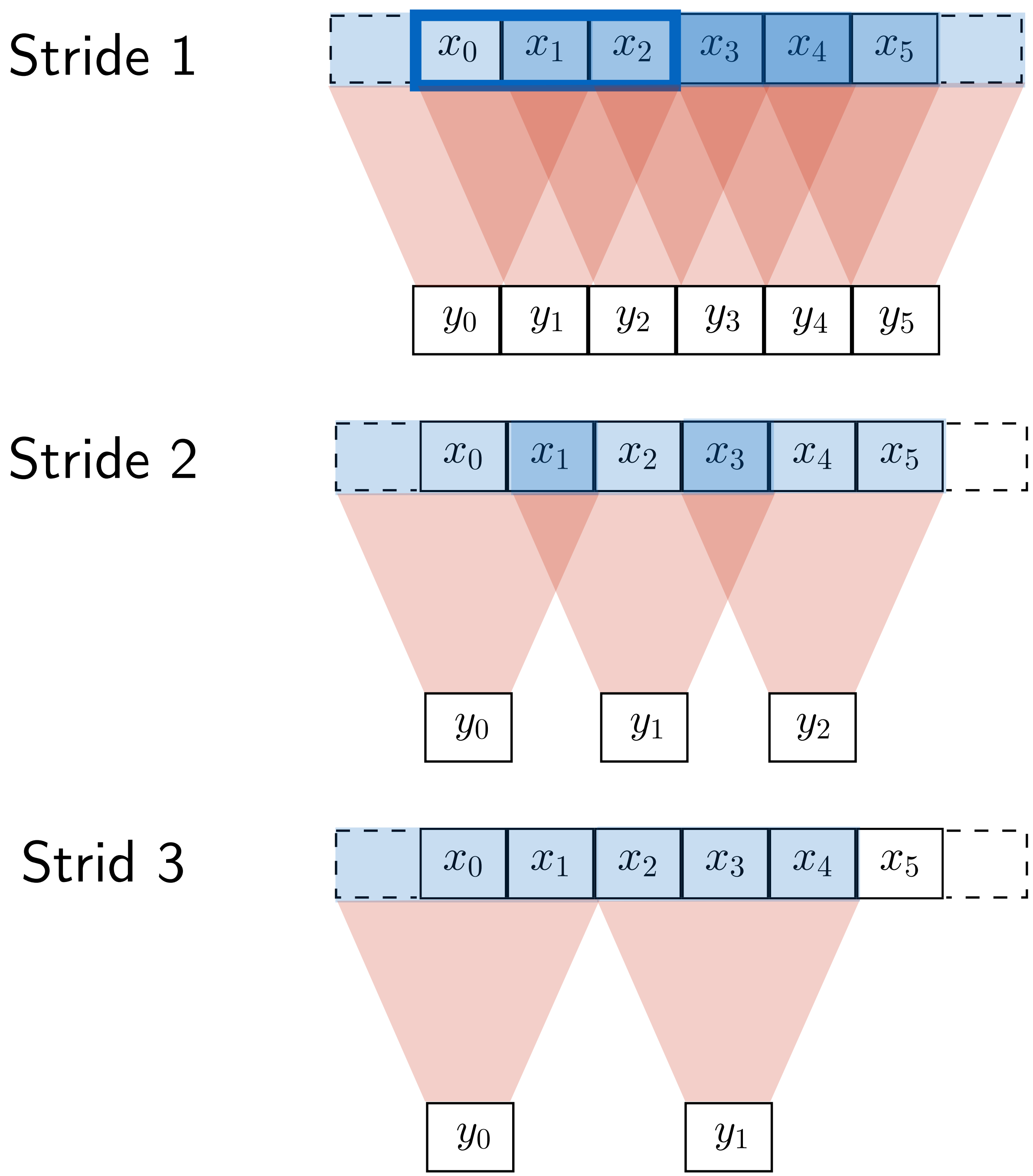


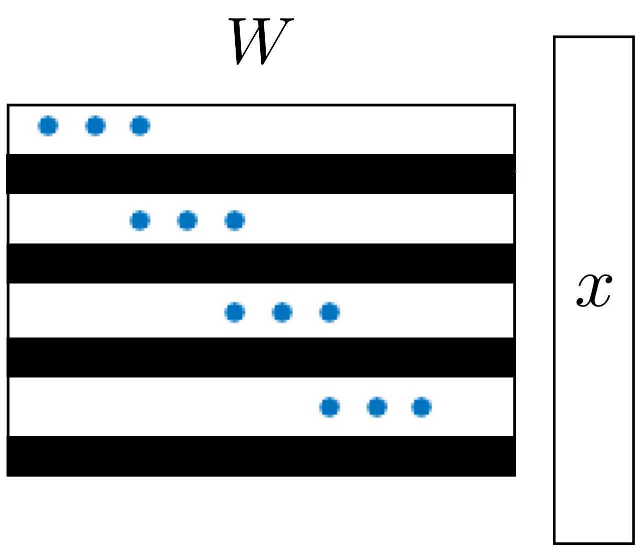
✦ max and average pooling are invariant to permutations of responses within a cell

✦ Once spacial resolution has been decreased, we can afford to increase the number of channels

Pooling: Convolution with Stride

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

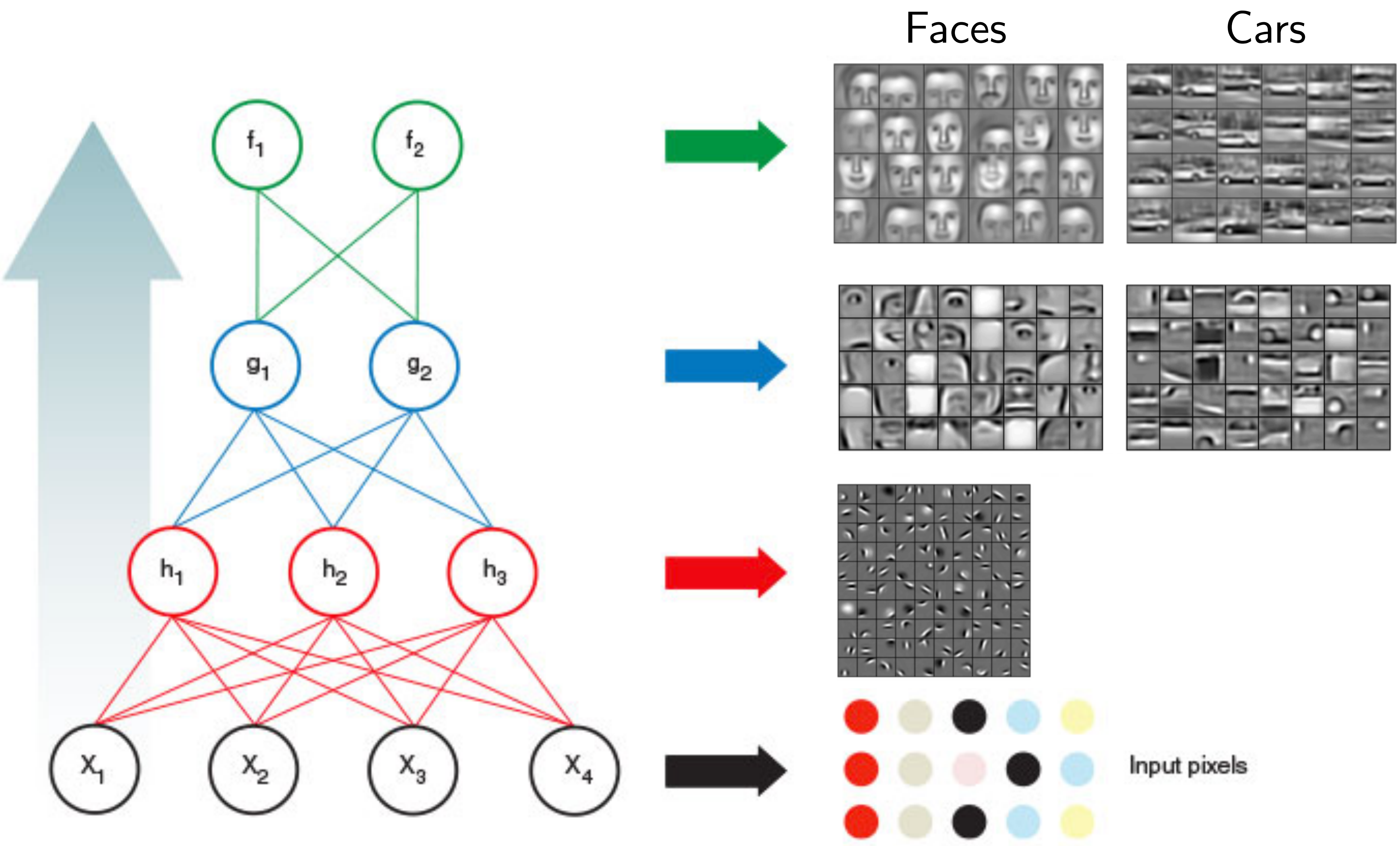




All variants and detail: [Dumoulin, Visin (2018): A guide to convolution arithmetic for deep learning]

Hierarchy of Parts Phenomenon

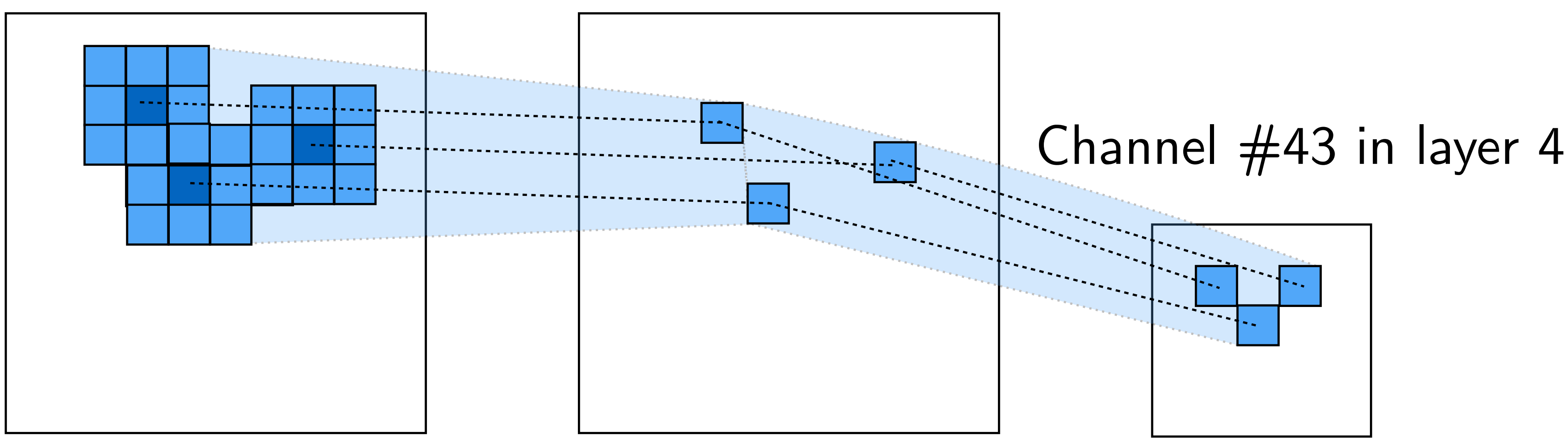
- In networks trained for different complex problems many intermediate layers activations correspond object parts



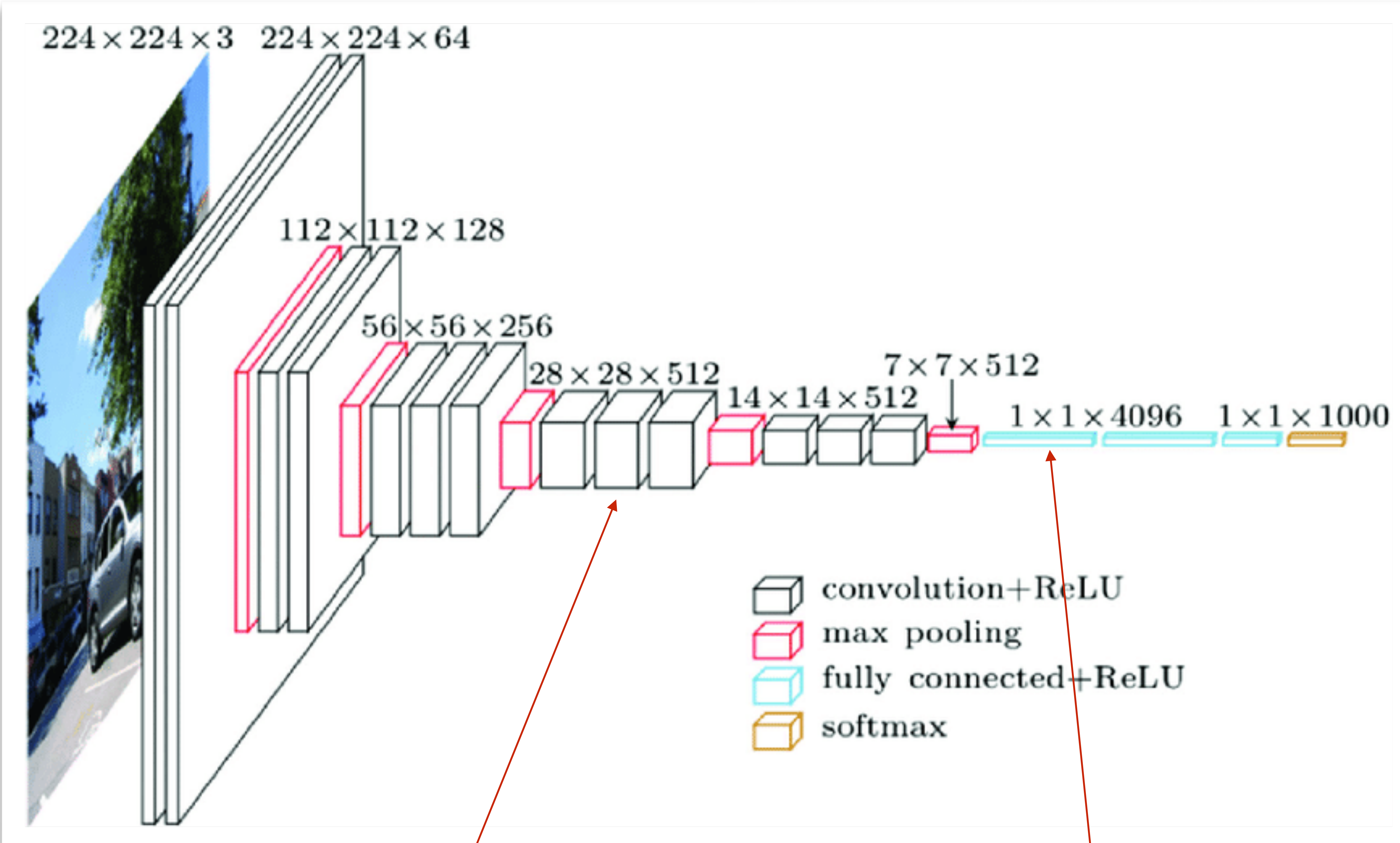
Hierarchy of Parts Phenomenon

- In networks trained for different complex problems many intermediate layers activations correspond object parts

lamps in places net | wheels in object net | people in video net



Classification CNN: 1x1 and Structured Conv



too many weights here --
could use structured convolution

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

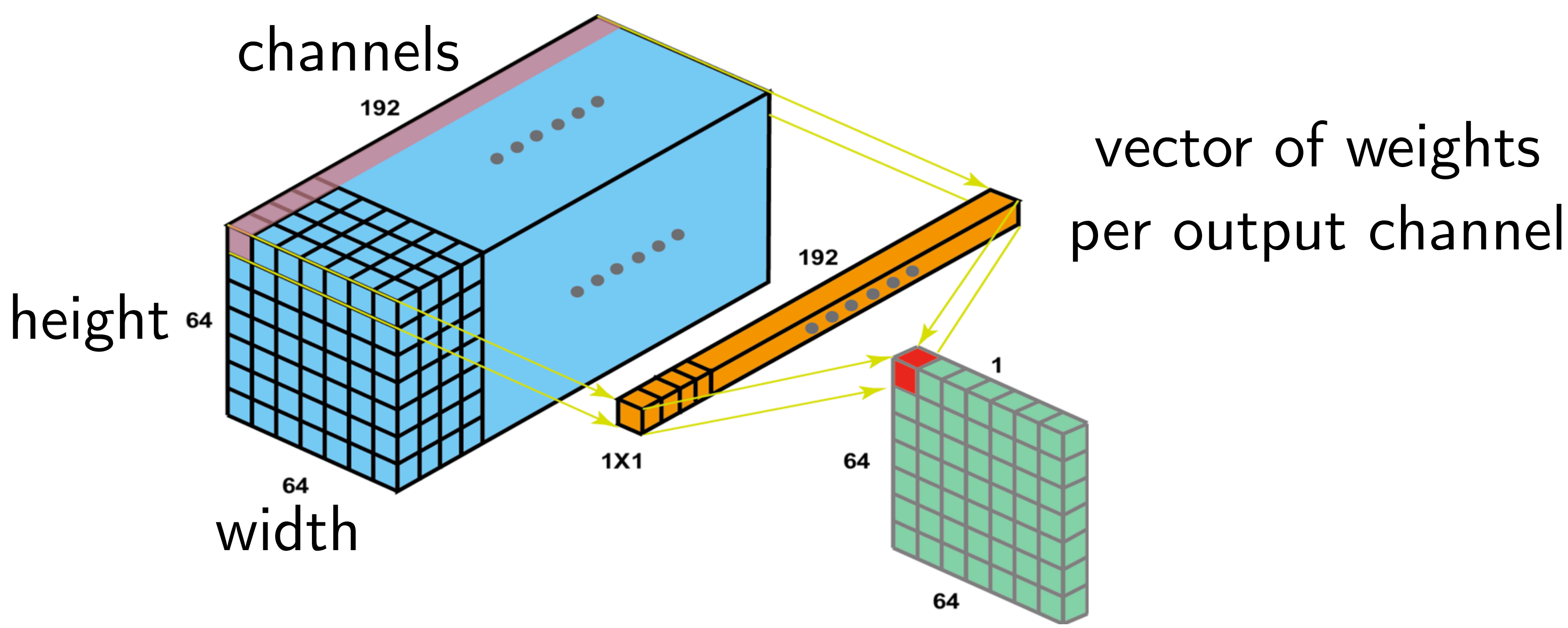
1x1 and Structured Convolutions

- Kernel size 1×1:

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

$$= \sum_c w_{o,c,0,0}\ x_{c,i,j}$$

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



Example 3×3, $256\to256$, is too expensive, simplify:

Input: WxHx256

$\mathrm{conv}(1\times1, 256 \to 64)$

$\mathrm{conv}(3\times3, 64 \to 64)$

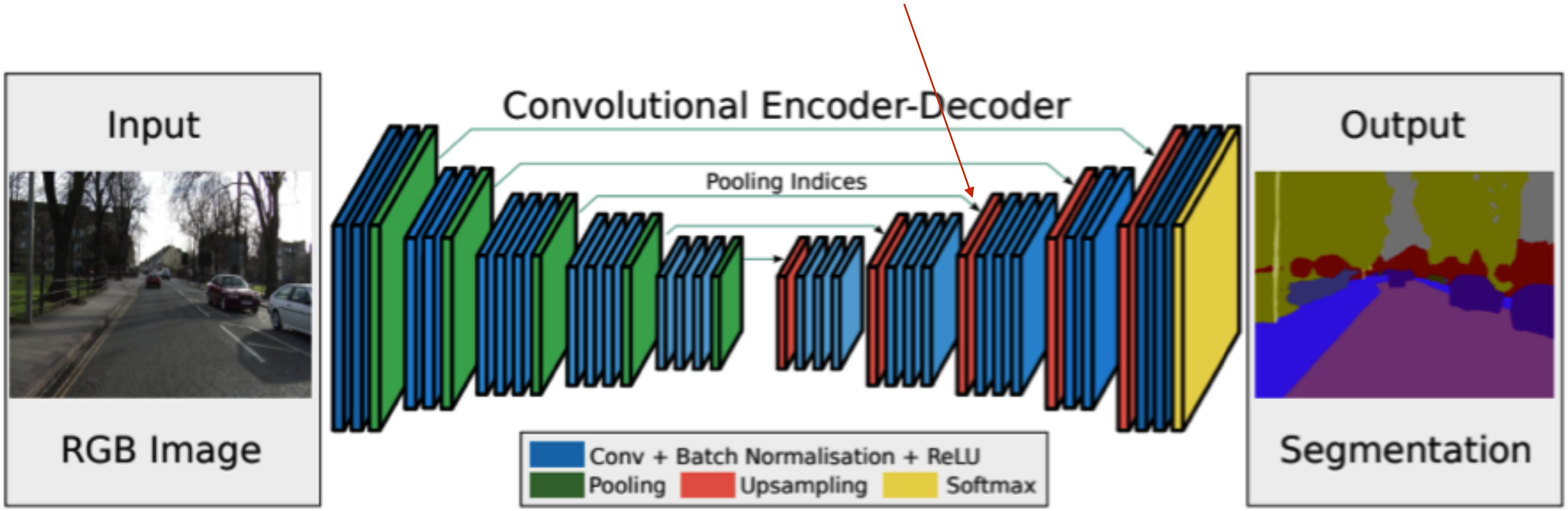
$\mathrm{conv}(1\times1, 64 \to 256)$

Output: W'xH'x256

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

Deconvolution for Segmentation

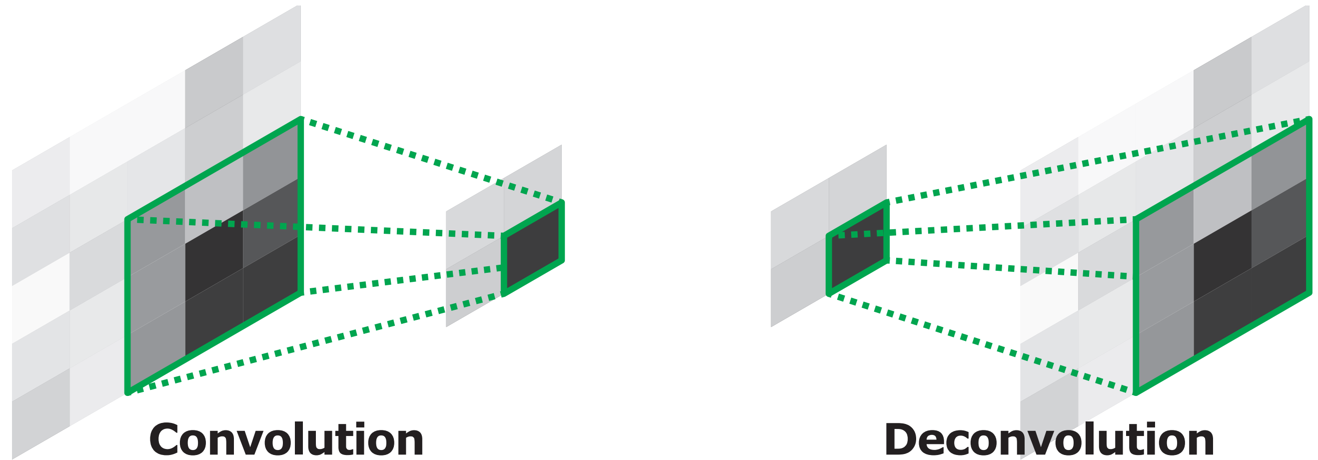
Semantic segmentation architectures need unpooling / upsampling



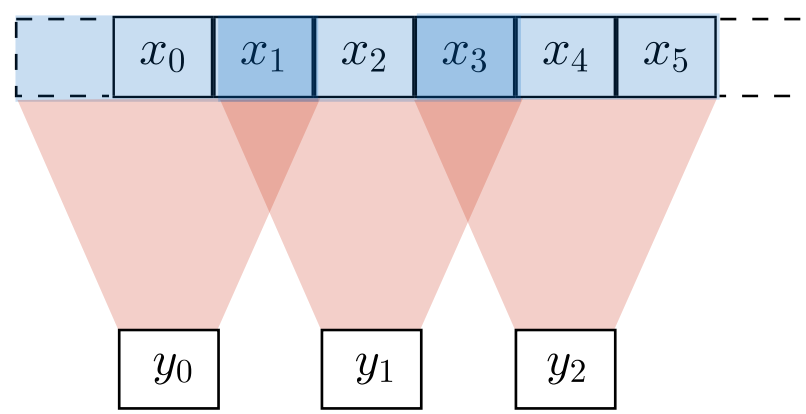
We will look at up-sampling with “transposed” convolution (“deconvolution”)

Transposed Convolution

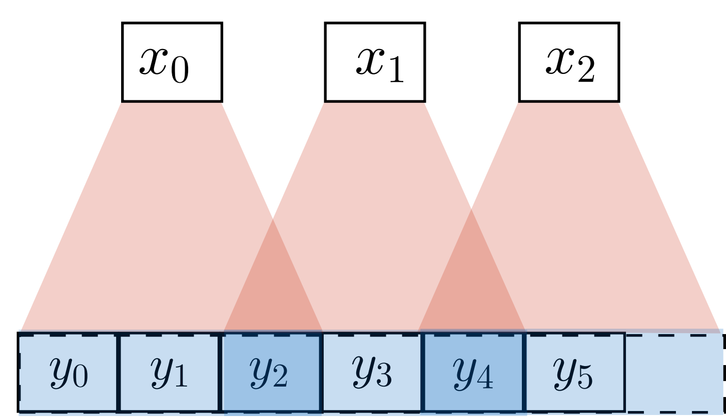
- Deconvolution = Transposed strided convolution = backprop of strided convolution



Stride 2 Convolution



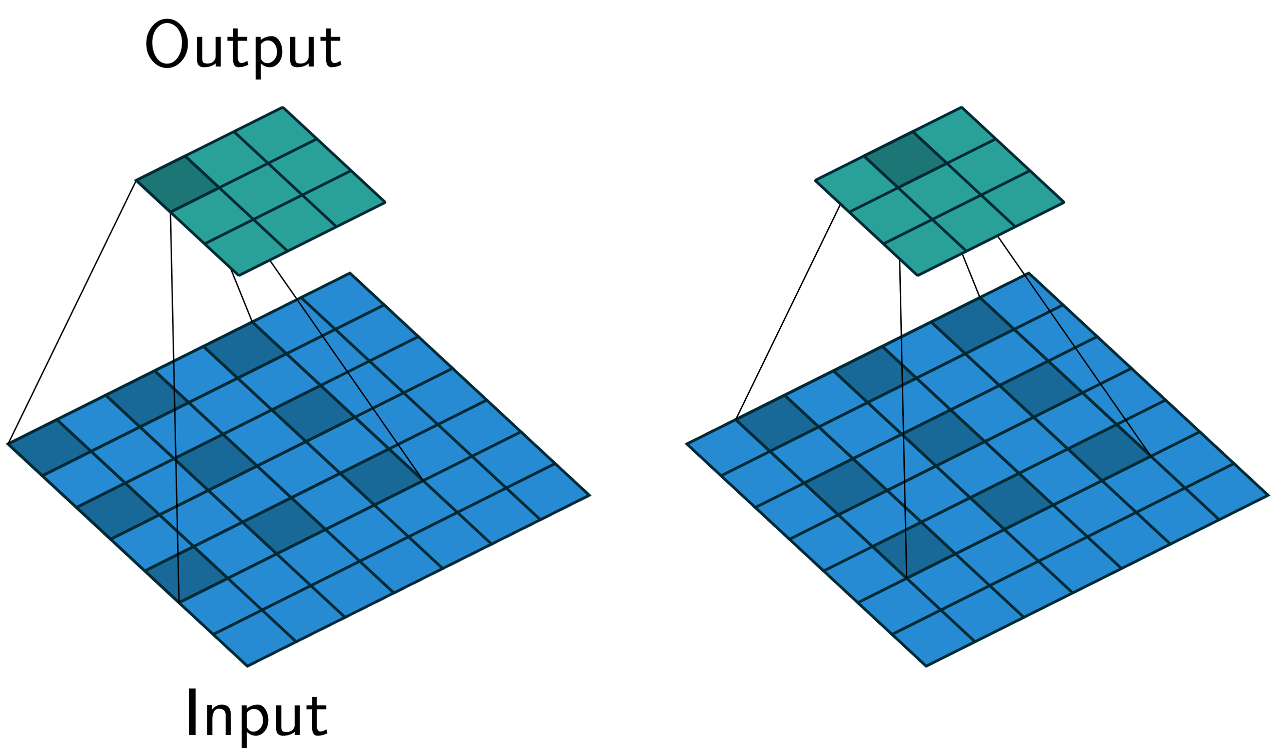
Stride 2 Deconvolution



Sparse & Deformable 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