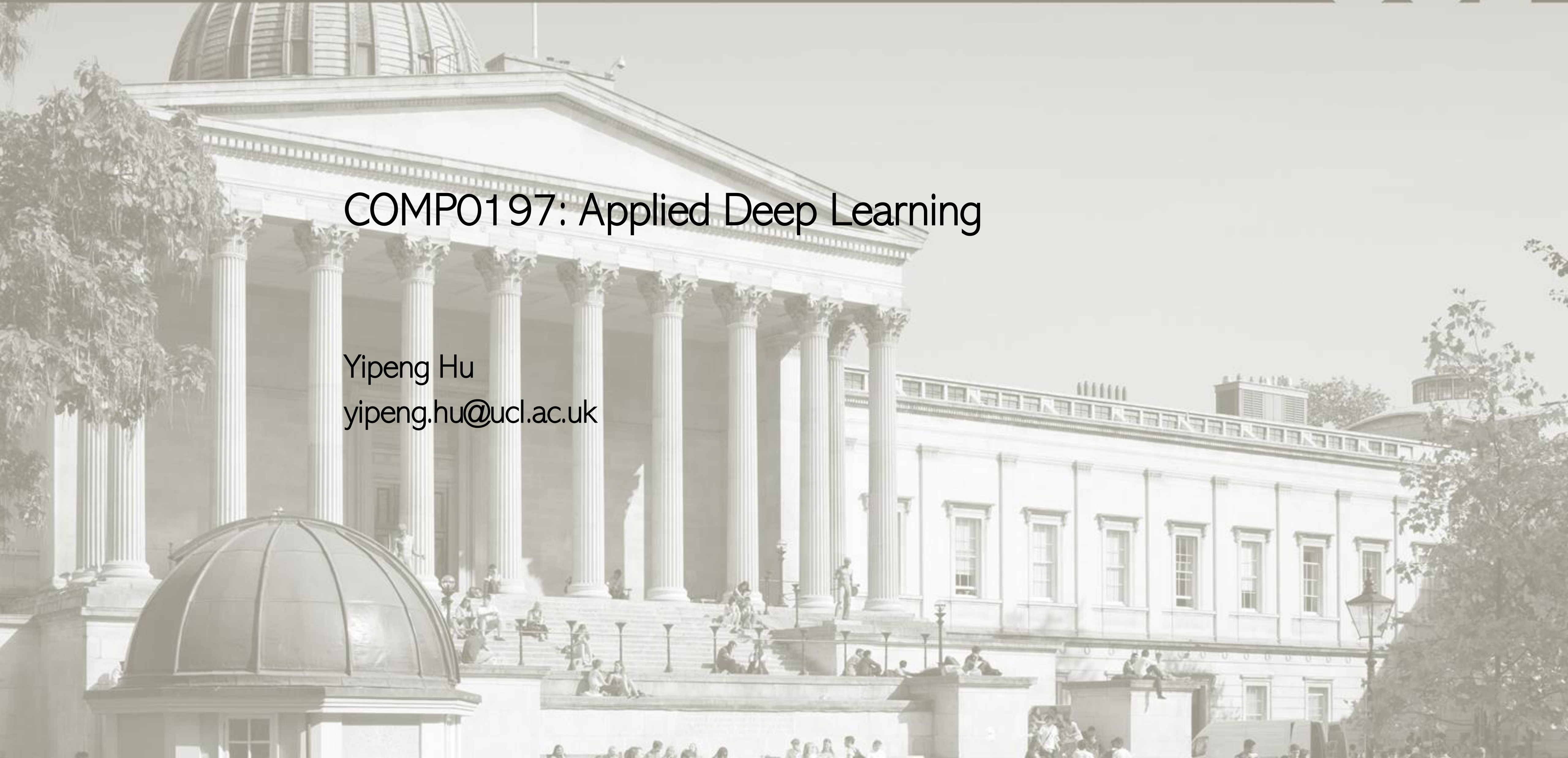


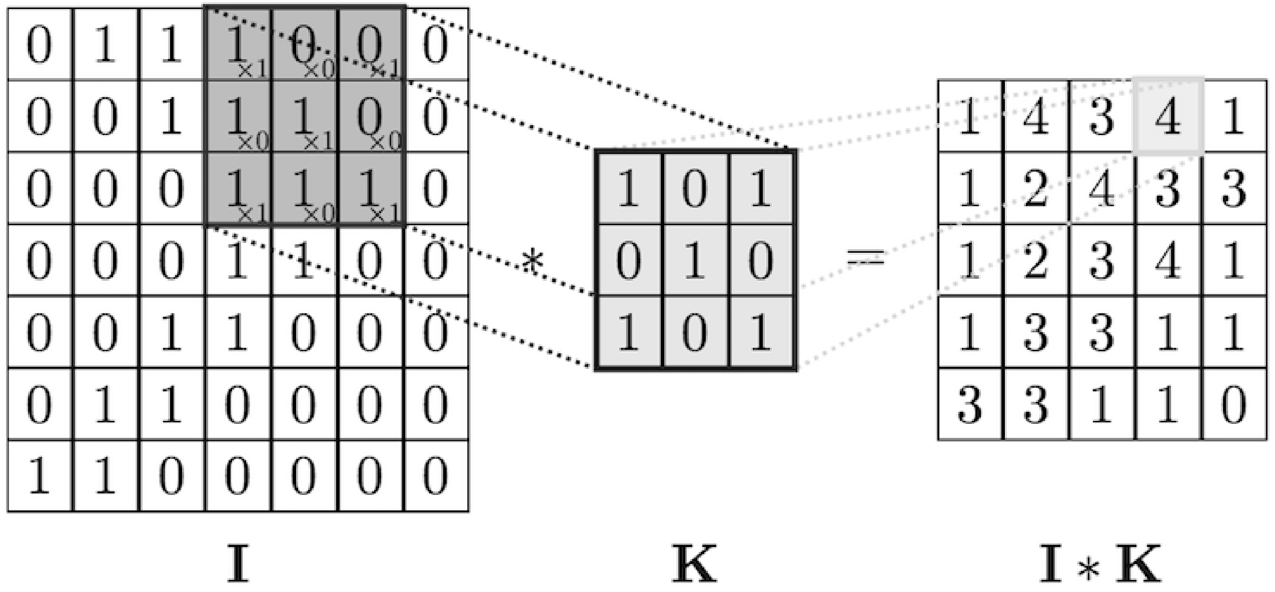
COMP0197: Applied Deep Learning

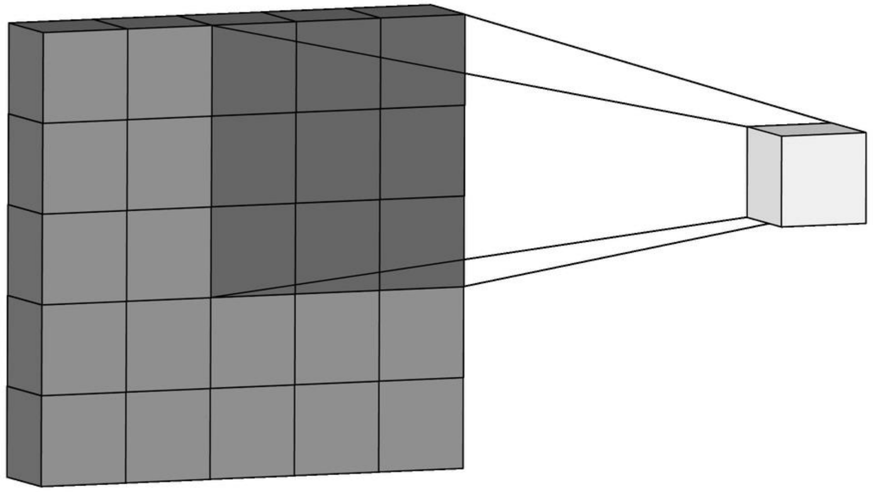
Yipeng Hu
yipeng.hu@ucl.ac.uk

Convolutional Neural Networks

Discrete convolution

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$





Convolutional Neural Networks

$$\begin{bmatrix} x_1 & x_4 & x_7 \\ x_2 & x_5 & x_8 \\ x_3 & x_6 & x_9 \end{bmatrix} * \begin{bmatrix} k_1 & k_3 \\ k_2 & k_4 \end{bmatrix} = \begin{bmatrix} \sum\sum\begin{pmatrix} x_1k_1 & x_4k_3 \\ x_2k_2 & x_5k_4 \end{pmatrix} & \sum\sum\begin{pmatrix} x_4k_1 & x_7k_3 \\ x_5k_2 & x_8k_4 \end{pmatrix} \\ \sum\sum\begin{pmatrix} x_2k_1 & x_5k_3 \\ x_3k_2 & x_6k_4 \end{pmatrix} & \sum\sum\begin{pmatrix} x_5k_1 & x_8k_3 \\ x_6k_2 & x_9k_4 \end{pmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} x_1k_1 + x_2k_2 + x_4k_3 + x_5k_4 & x_4k_1 + x_5k_2 + x_7k_3 + x_8k_4 \\ x_2k_1 + x_3k_2 + x_5k_3 + x_6k_4 & x_5k_1 + x_6k_2 + x_8k_3 + x_9k_4 \end{bmatrix}$$

$$\begin{bmatrix} \qquad\qquad\qquad\qquad\qquad\qquad \\ \\ \\ \\ \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} x_1k_1 + x_2k_2 + x_4k_3 + x_5k_4 \\ x_2k_1 + x_3k_2 + x_5k_3 + x_6k_4 \\ x_4k_1 + x_5k_2 + x_7k_3 + x_8k_4 \\ x_5k_1 + x_6k_2 + x_8k_3 + x_9k_4 \end{bmatrix}$$

Convolutional Neural Networks

$$\begin{bmatrix} x_1 & x_4 & x_7 \\ x_2 & x_5 & x_8 \\ x_3 & x_6 & x_9 \end{bmatrix} * \begin{bmatrix} k_1 & k_3 \\ k_2 & k_4 \end{bmatrix} = \begin{bmatrix} \sum\sum\begin{pmatrix} x_1k_1 & x_4k_3 \\ x_2k_2 & x_5k_4 \end{pmatrix} & \sum\sum\begin{pmatrix} x_4k_1 & x_7k_3 \\ x_5k_2 & x_8k_4 \end{pmatrix} \\ \sum\sum\begin{pmatrix} x_2k_1 & x_5k_3 \\ x_3k_2 & x_6k_4 \end{pmatrix} & \sum\sum\begin{pmatrix} x_5k_1 & x_8k_3 \\ x_6k_2 & x_9k_4 \end{pmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} x_1k_1 + x_2k_2 + x_4k_3 + x_5k_4 & x_4k_1 + x_5k_2 + x_7k_3 + x_8k_4 \\ x_2k_1 + x_3k_2 + x_5k_3 + x_6k_4 & x_5k_1 + x_6k_2 + x_8k_3 + x_9k_4 \end{bmatrix}$$

$$\begin{bmatrix} k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 & 0 \\ 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 & 0 \\ 0 & 0 & 0 & 0 & k_1 & k_2 & 0 & k_3 & k_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} x_1k_1 + x_2k_2 + x_4k_3 + x_5k_4 \\ x_2k_1 + x_3k_2 + x_5k_3 + x_6k_4 \\ x_4k_1 + x_5k_2 + x_7k_3 + x_8k_4 \\ x_5k_1 + x_6k_2 + x_8k_3 + x_9k_4 \end{bmatrix}$$

Convolutional Neural Networks | Convolution and Cross-Correlation

Convolution

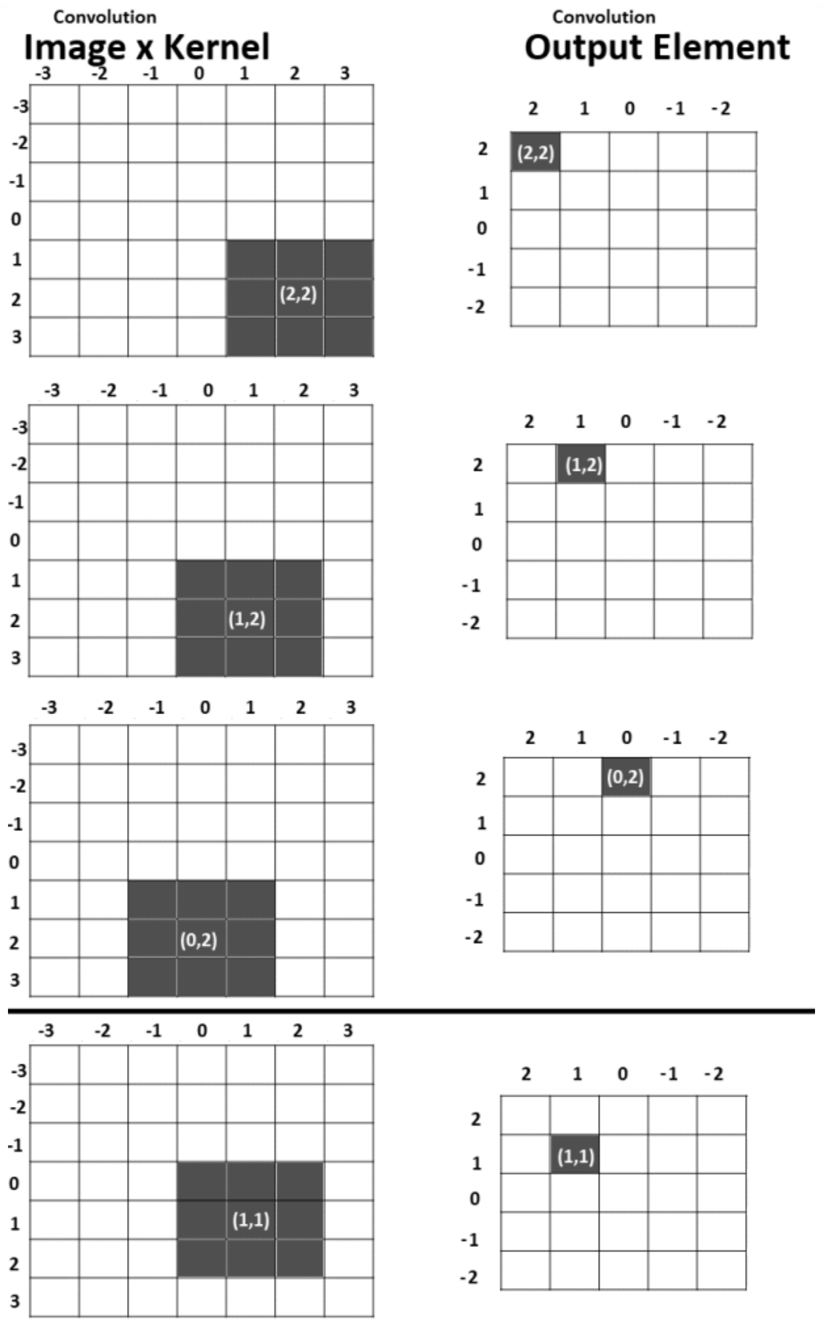
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n).$$

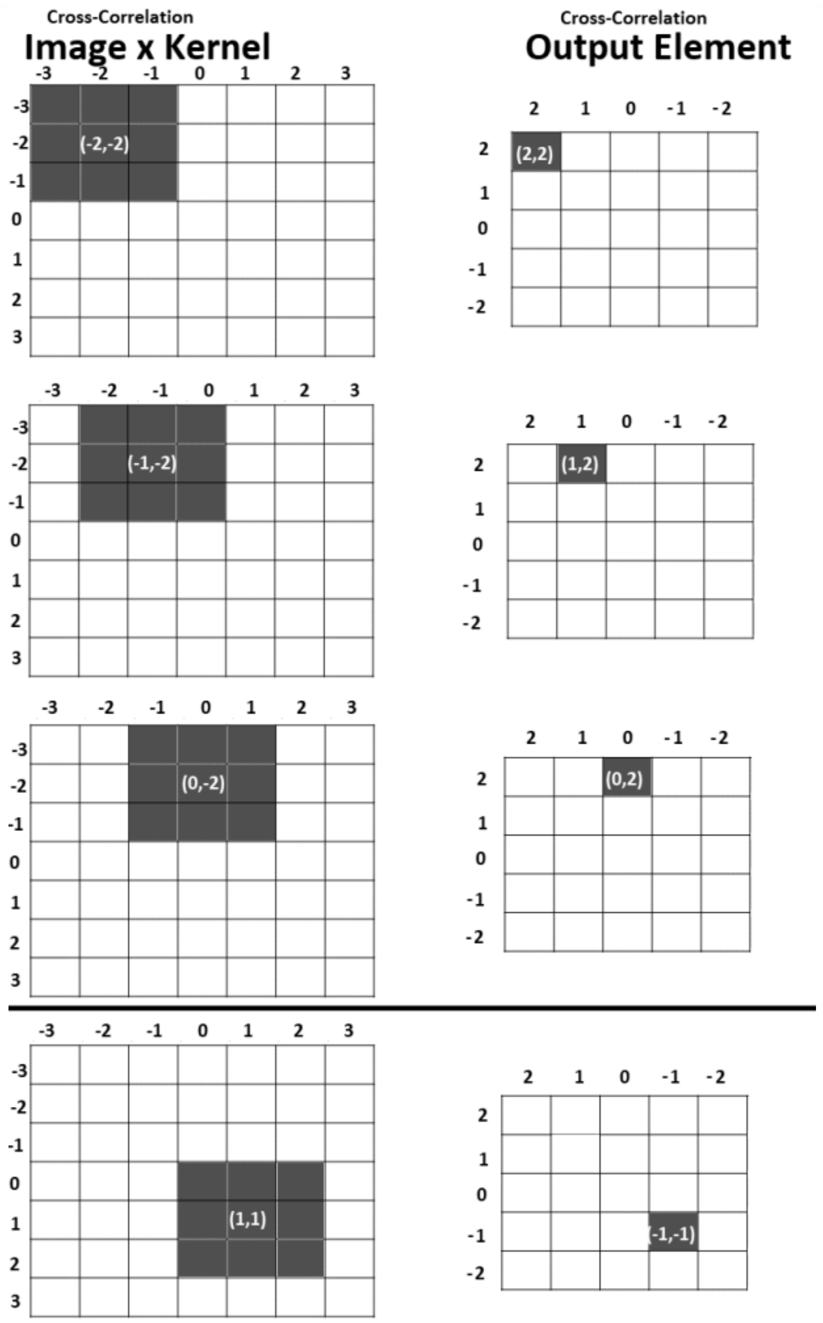
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n).$$

Cross-correlation

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

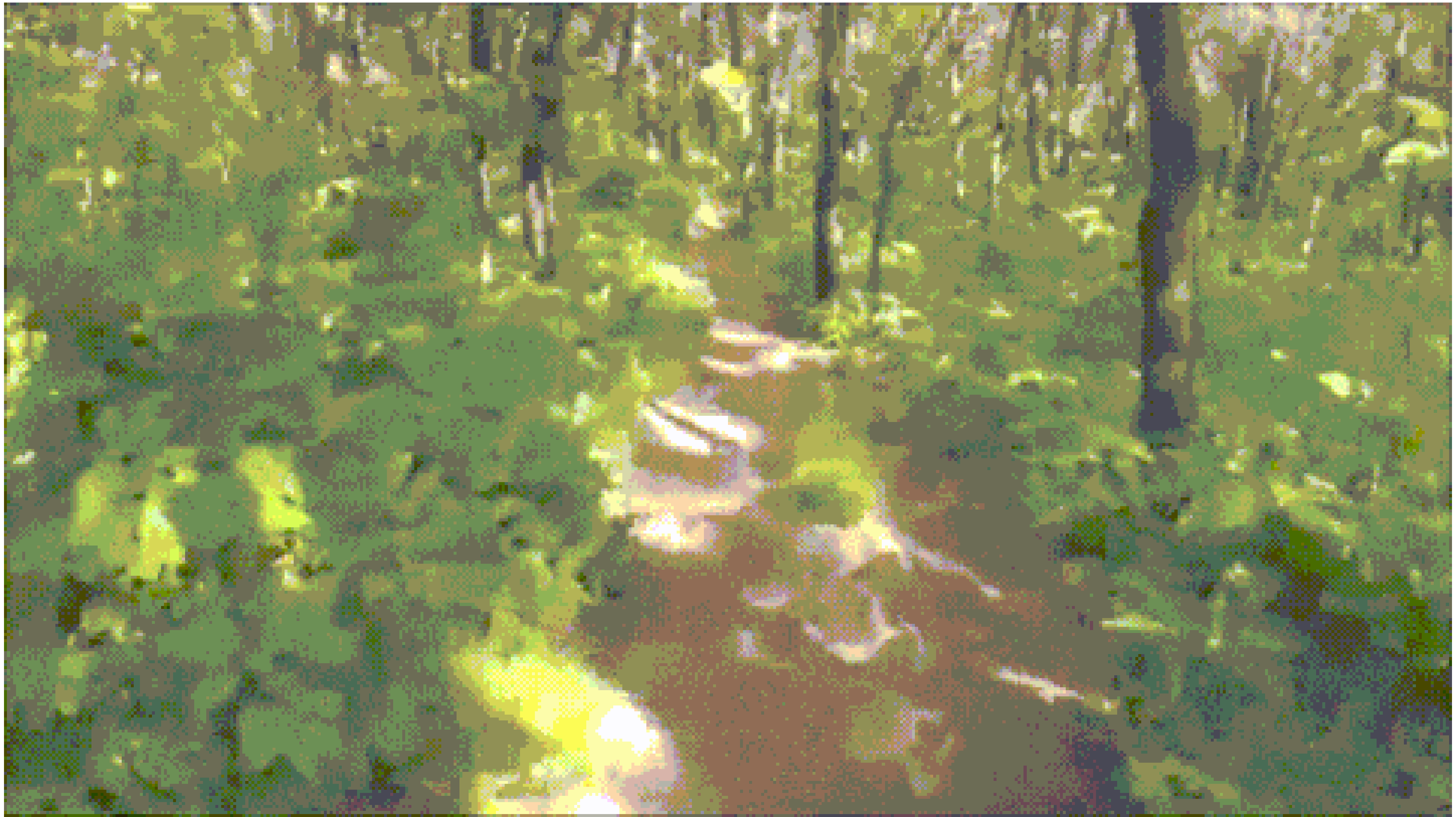
Convolutional Neural Networks | Convolution and Cross-Correlation



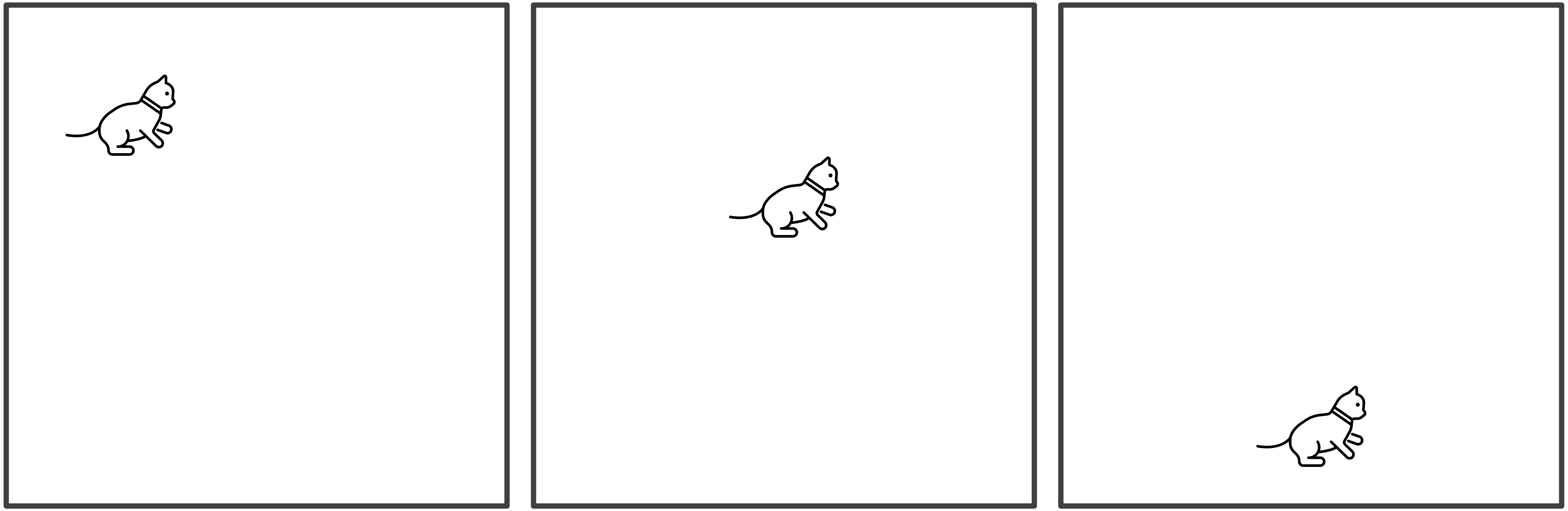


Convolutional Neural Networks | Intuition

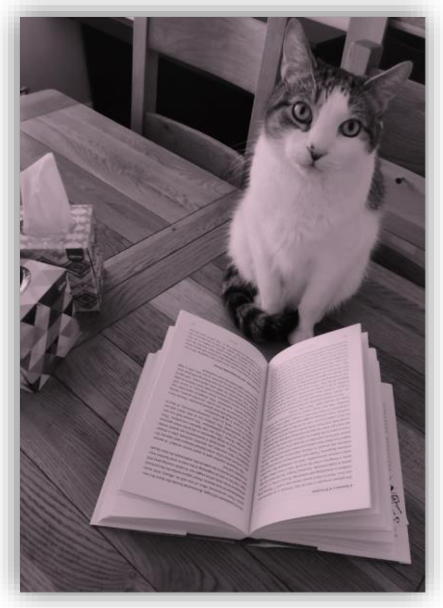
Translation invariance



Translation invariance



Translation invariance

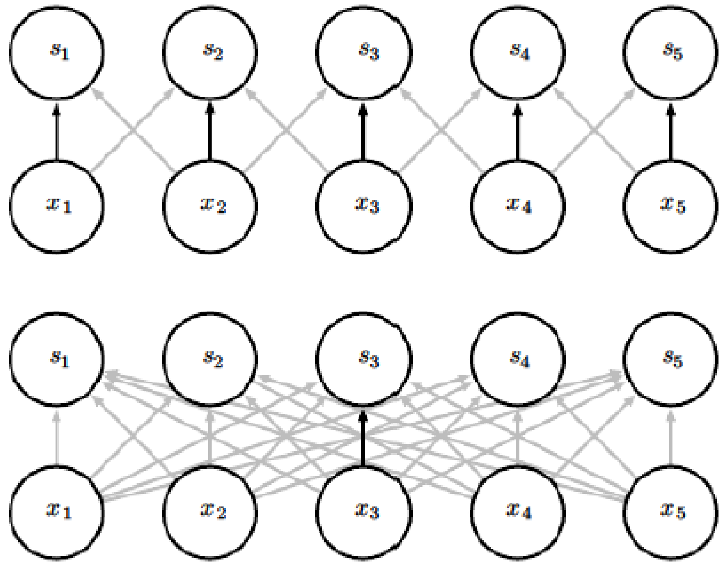


Parameter sharing

Regularisation effect*, efficiency

$$\begin{bmatrix} w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} x_1 w_1 + x_2 w_2 + x_4 w_3 + x_5 w_4 \\ x_2 w_1 + x_3 w_2 + x_5 w_3 + x_6 w_4 \\ x_4 w_1 + x_5 w_2 + x_7 w_3 + x_8 w_4 \\ x_5 w_1 + x_6 w_2 + x_8 w_3 + x_9 w_4 \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} & w_{1,7} & w_{1,8} & w_{1,9} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} & w_{2,7} & w_{2,8} & w_{2,9} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} & w_{3,7} & w_{3,8} & w_{3,9} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & w_{4,5} & w_{4,6} & w_{4,7} & w_{4,8} & w_{4,9} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}$$

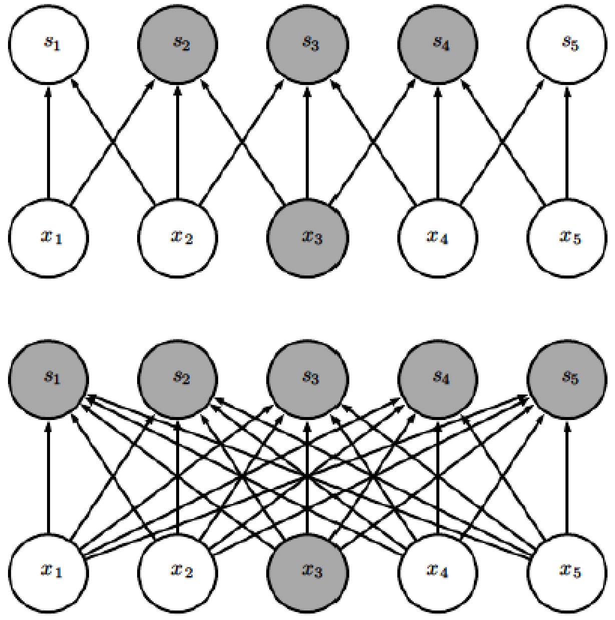


Sparse weights

Memory, computation, statistical efficiency

$$\begin{bmatrix} w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} x_1 w_1 + x_2 w_2 + x_4 w_3 + x_5 w_4 \\ x_2 w_1 + x_3 w_2 + x_5 w_3 + x_6 w_4 \\ x_4 w_1 + x_5 w_2 + x_7 w_3 + x_8 w_4 \\ x_5 w_1 + x_6 w_2 + x_8 w_3 + x_9 w_4 \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} & w_{1,7} & w_{1,8} & w_{1,9} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} & w_{2,7} & w_{2,8} & w_{2,9} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} & w_{3,7} & w_{3,8} & w_{3,9} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & w_{4,5} & w_{4,6} & w_{4,7} & w_{4,8} & w_{4,9} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}$$



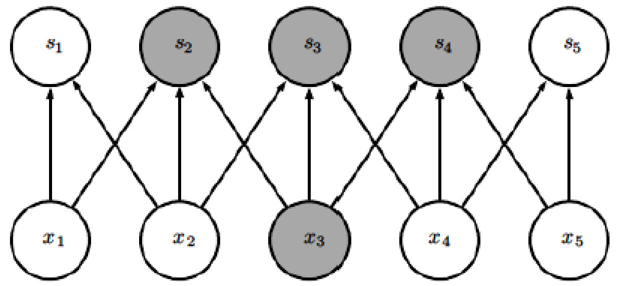
Convolutional Neural Networks | Intuition

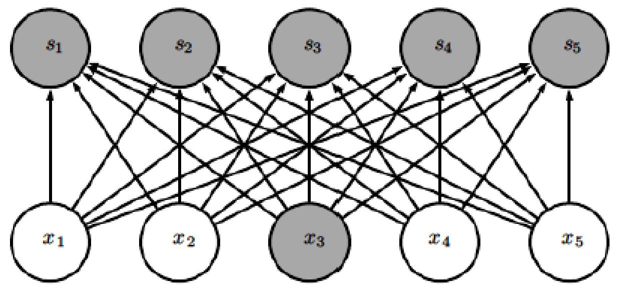
Infinitely strong prior

Underfitting

$$\begin{bmatrix} w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 & 0 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} x_1 w_1 + x_2 w_2 + x_4 w_3 + x_5 w_4 \\ x_2 w_1 + x_3 w_2 + x_5 w_3 + x_6 w_4 \\ x_4 w_1 + x_5 w_2 + x_7 w_3 + x_8 w_4 \\ x_5 w_1 + x_6 w_2 + x_8 w_3 + x_9 w_4 \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} & w_{1,7} & w_{1,8} & w_{1,9} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} & w_{2,7} & w_{2,8} & w_{2,9} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} & w_{3,7} & w_{3,8} & w_{3,9} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} & w_{4,5} & w_{4,6} & w_{4,7} & w_{4,8} & w_{4,9} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}$$





Convolutional Neural Networks | Sampling

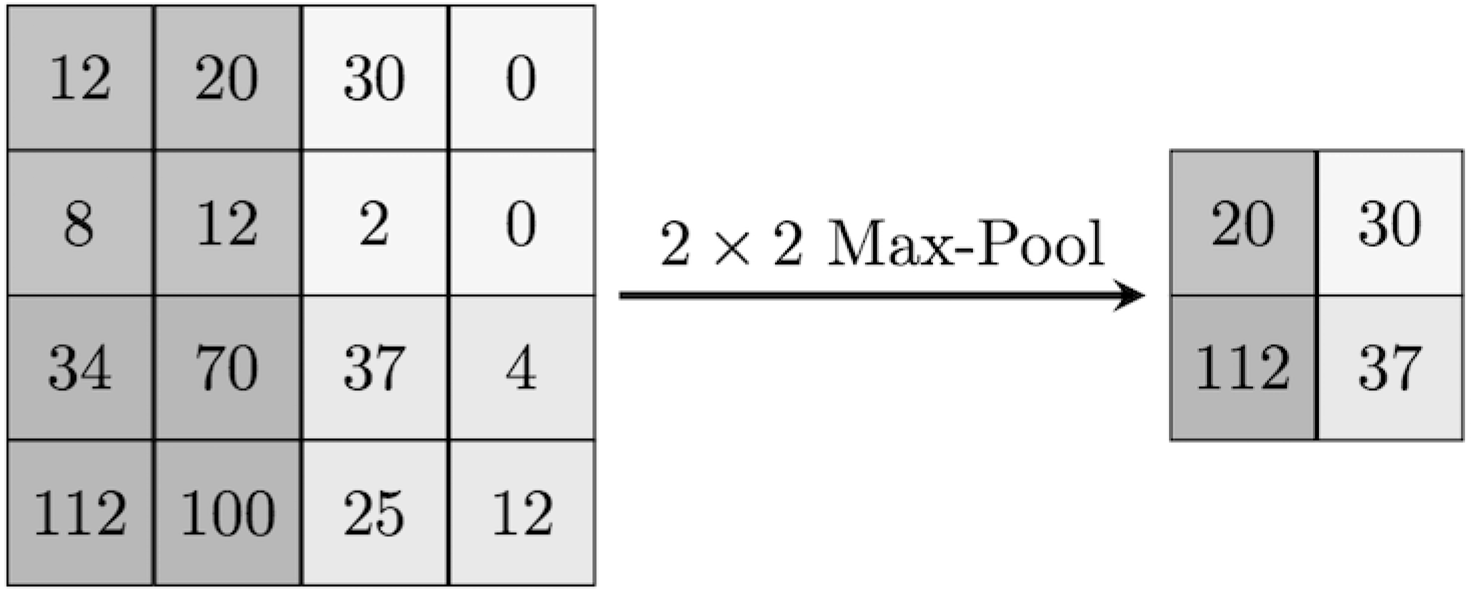
Pooling

- Max pooling
- Average pooling

Invariance
No learnable parameters

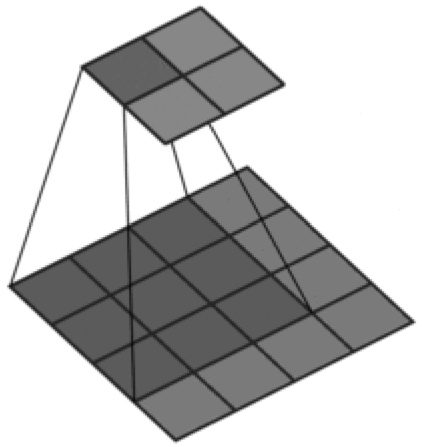
Required for down-sampling?
Conv weight -> activation -> pooling?

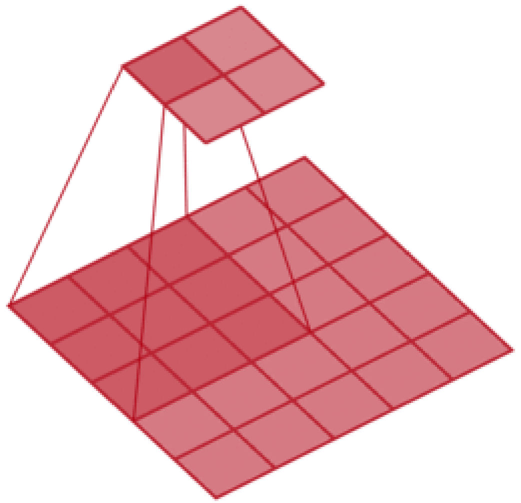
Moving windows with strides



Strides for down-sampling

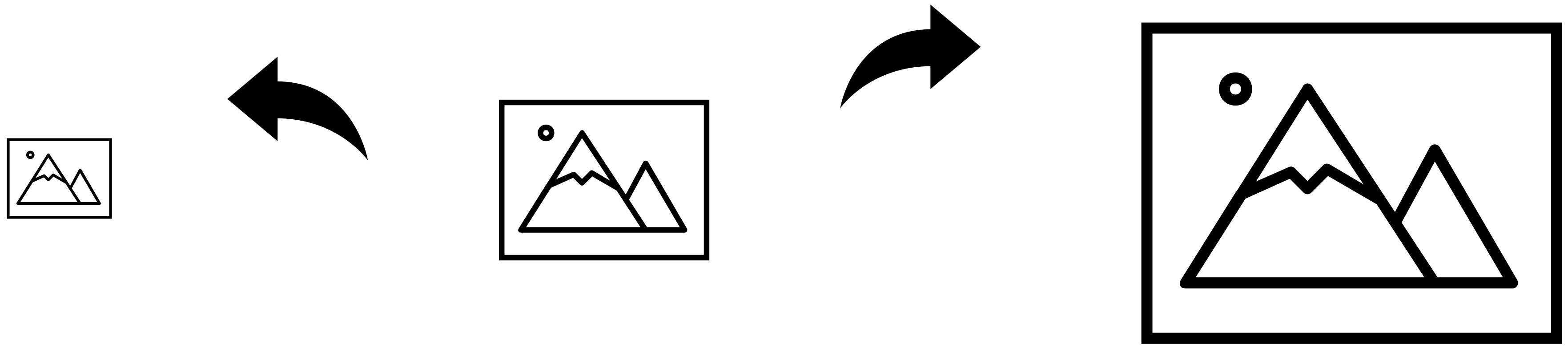
- Strides > 1
- Pooling
- Convolution with strides?





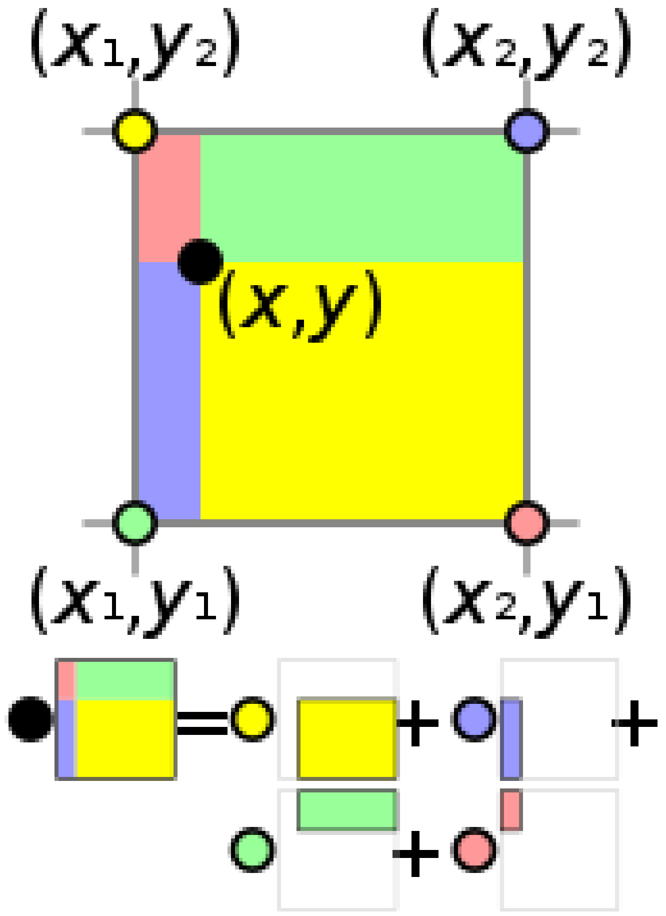
Interpolation for down-sampling and up-sampling

Bilinear, bicubic, spline.



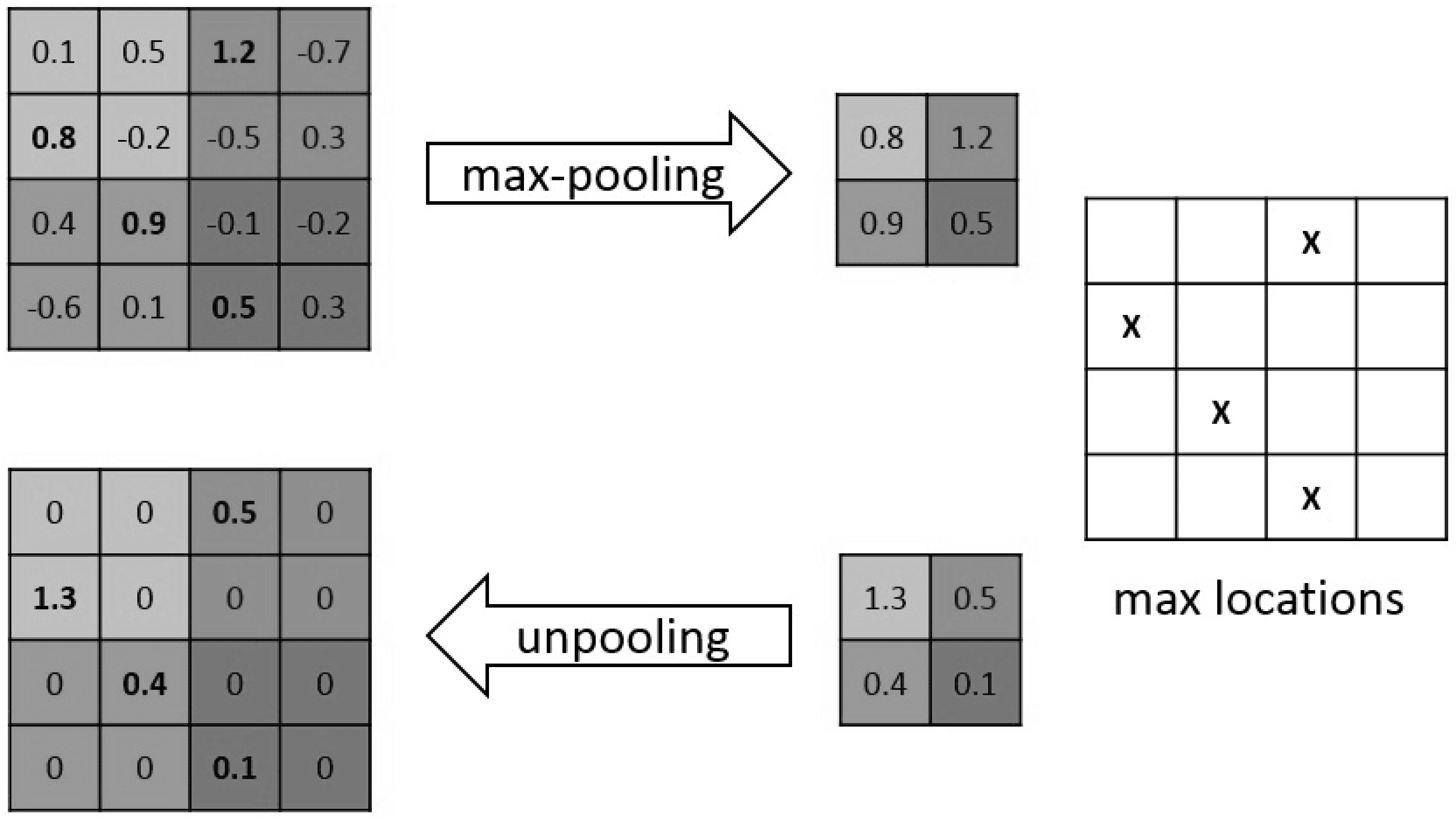
Interpolation for down-sampling and up-sampling

Is bilinear interpolation a linear operation?



“Un-pooling” for up-sampling

e.g. a fixed position for the max
and zeros everywhere else



Transpose convolution

- Fractionally strided convolution
- Deconvolution
- Up-sampling with parameters

$$\mathbf{y}^{(N\times 1)} = \mathbf{W}^{(N\times M)}\mathbf{x}^{(M\times 1)}$$

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

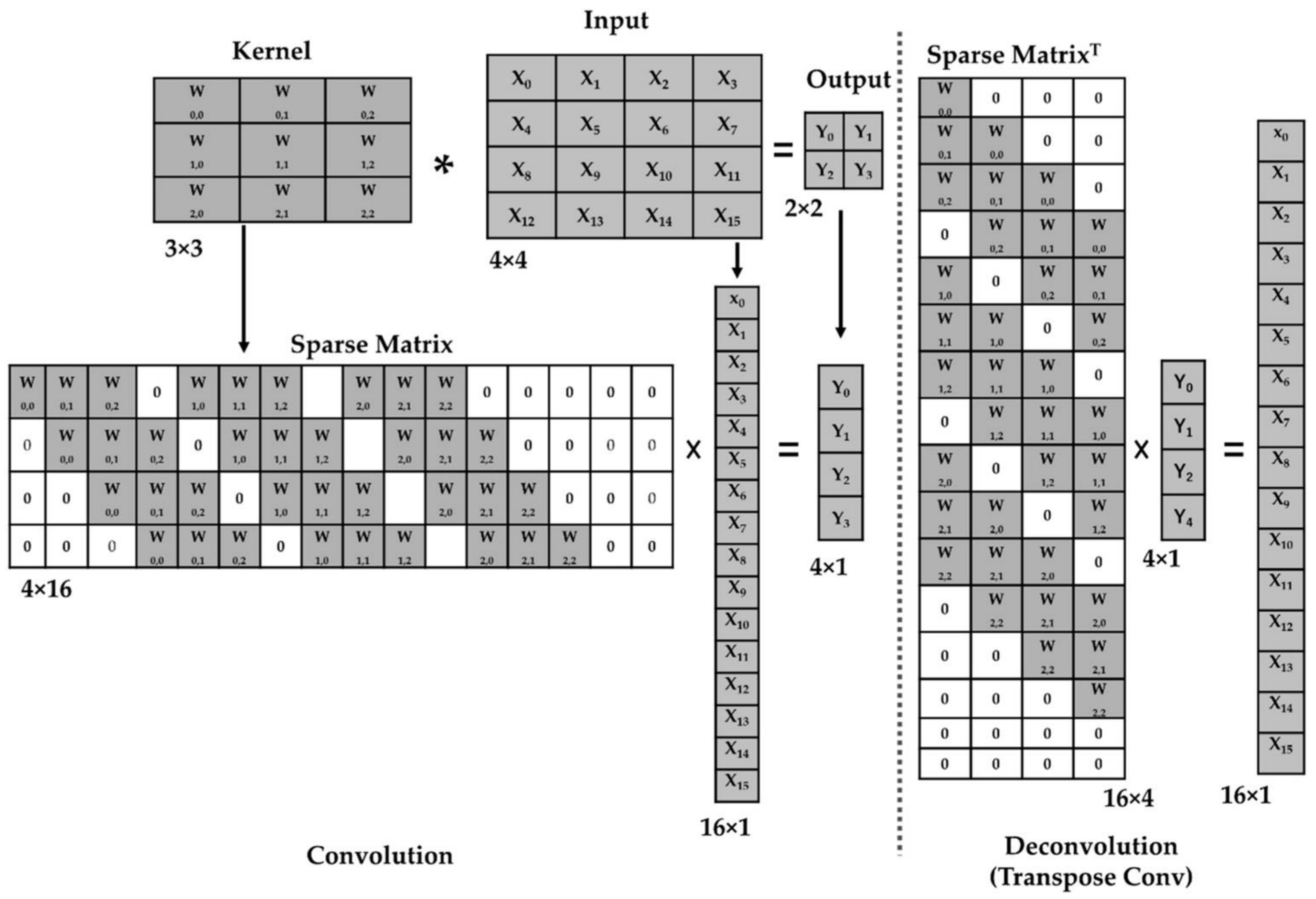
Diagonal-constant Toeplitz matrix

$$W_{i,j} = Const_{i,j}$$

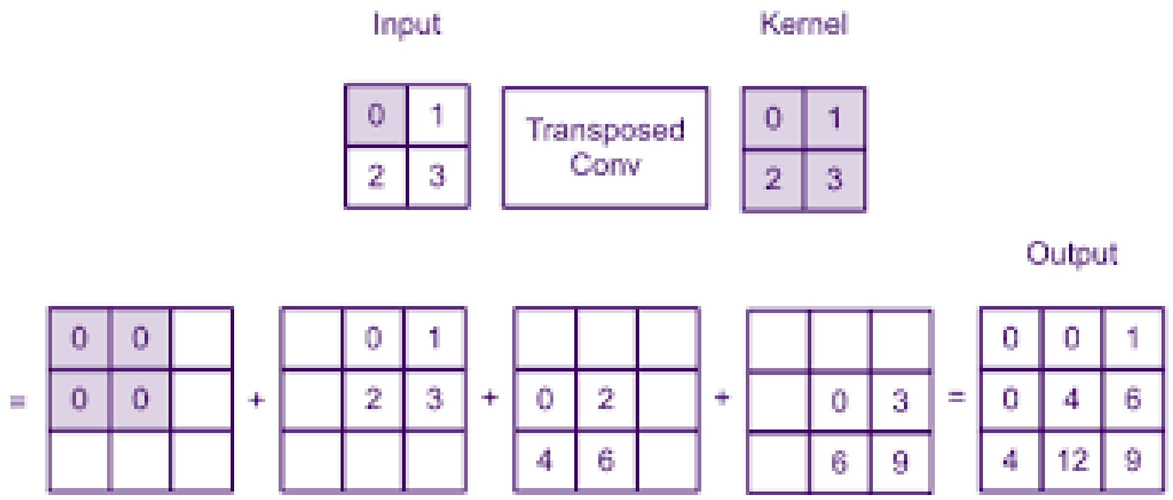
$$\widehat{W}_{j,i} = Const_{i,j}$$

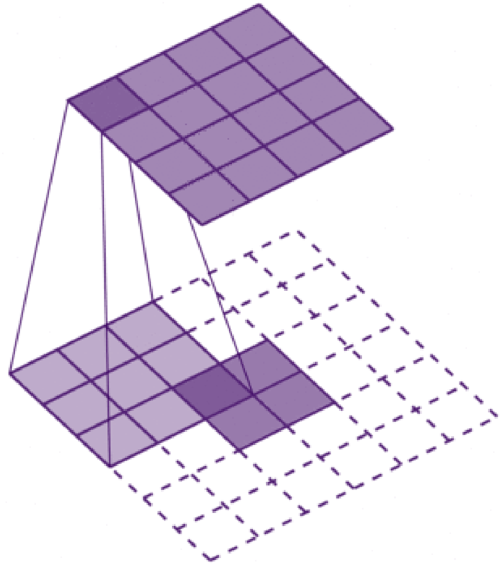
The reverse of convolution

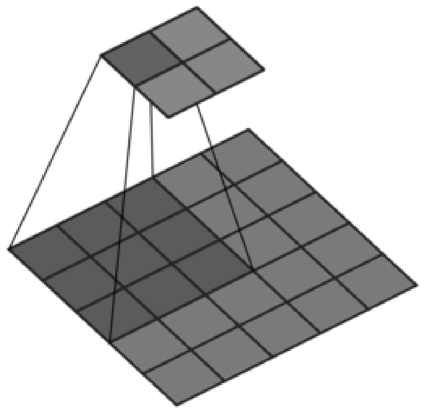
$$\widehat{\mathbf{W}}^{\mathrm{T}}\mathbf{y} = \mathbf{x}$$



Transpose convolution



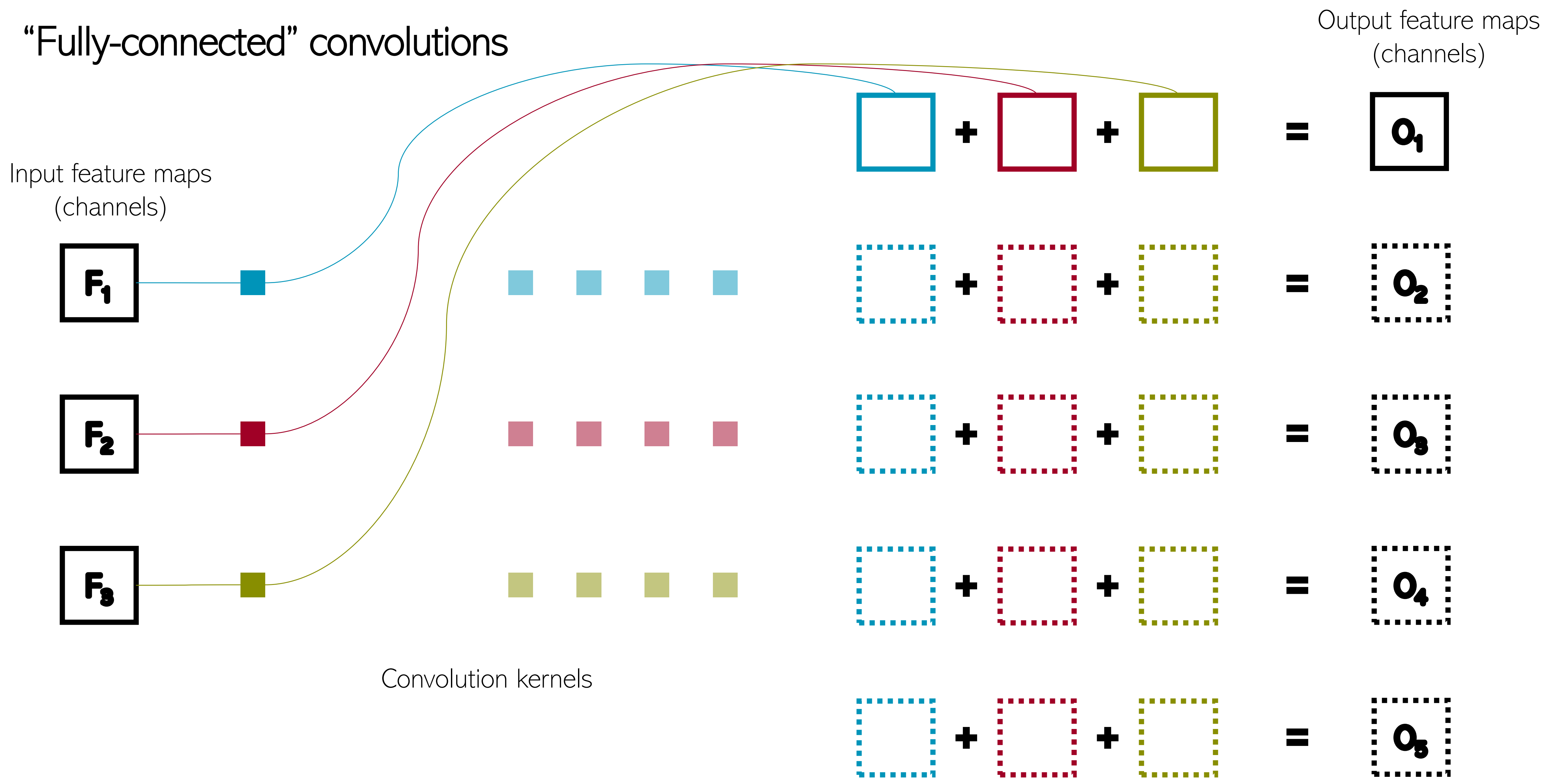




Transpose convolution vs. conv + interpolation, for up-sampling?

Convolutional Neural Networks | Anatomy of a Convolutional Layer

"Fully-connected" convolutions



Tensors

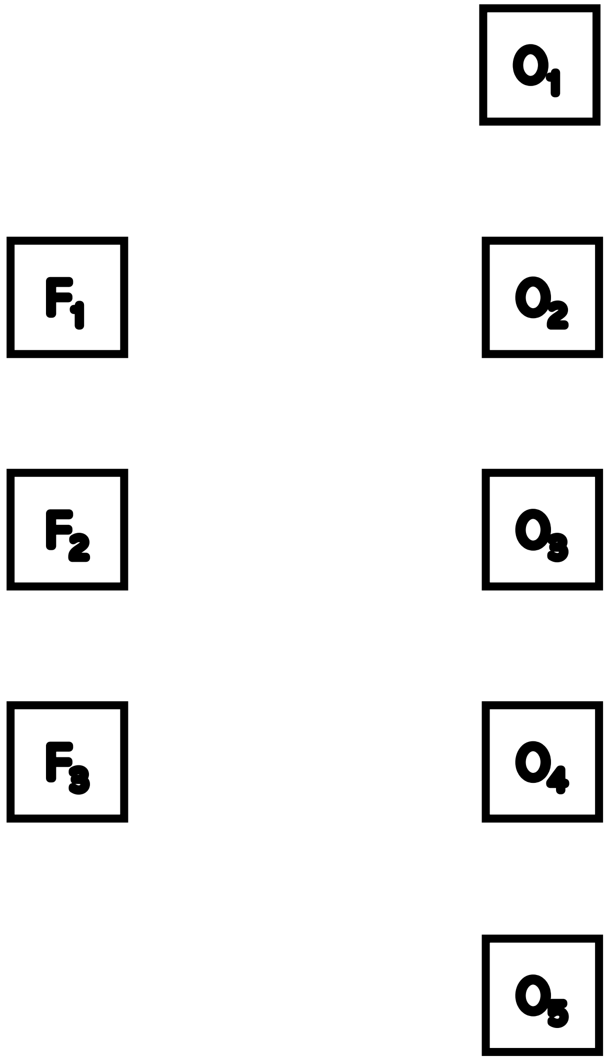
Input/output feature maps

Tensor for 2D feature maps
N x H x W x C (TensorFlow)
N x C x H x W (PyTorch)

N: no. of data in a minibatch (batch size)
H: height of feature map
W: width of feature map

Tensor for 1D feature vector
N x L x C (TensorFlow)
N x C x L (PyTorch)
L: length of feature vector

Tensor for 3D feature tensor
N x D x H x W x C (TensorFlow)
N x C x D x H x W (PyTorch)
D: depth of feature tensor



Convolutional Neural Networks | Kernels

Convolutional Neural Networks | Kernels

Dimensionality

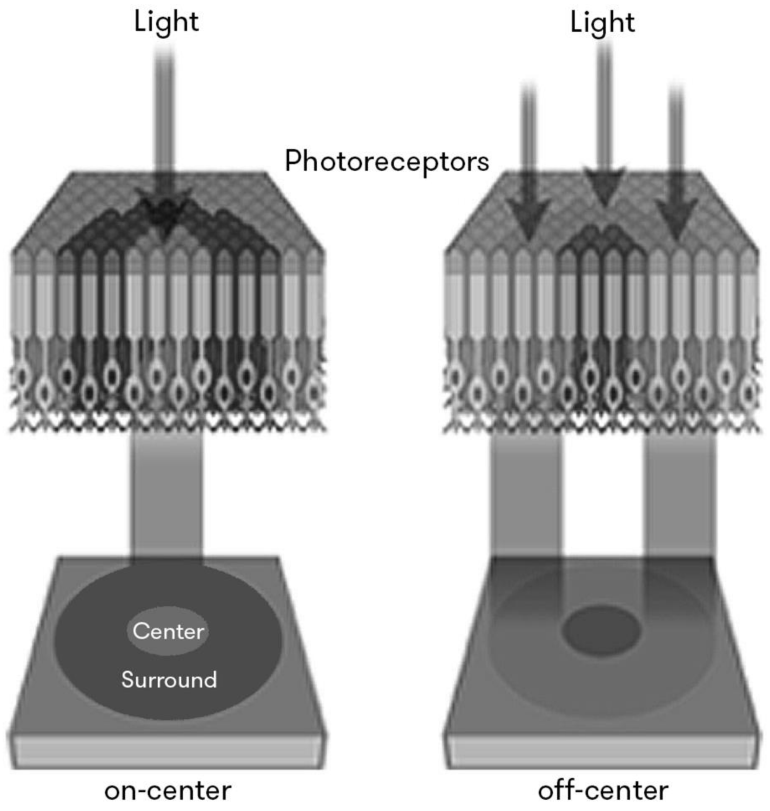
1D

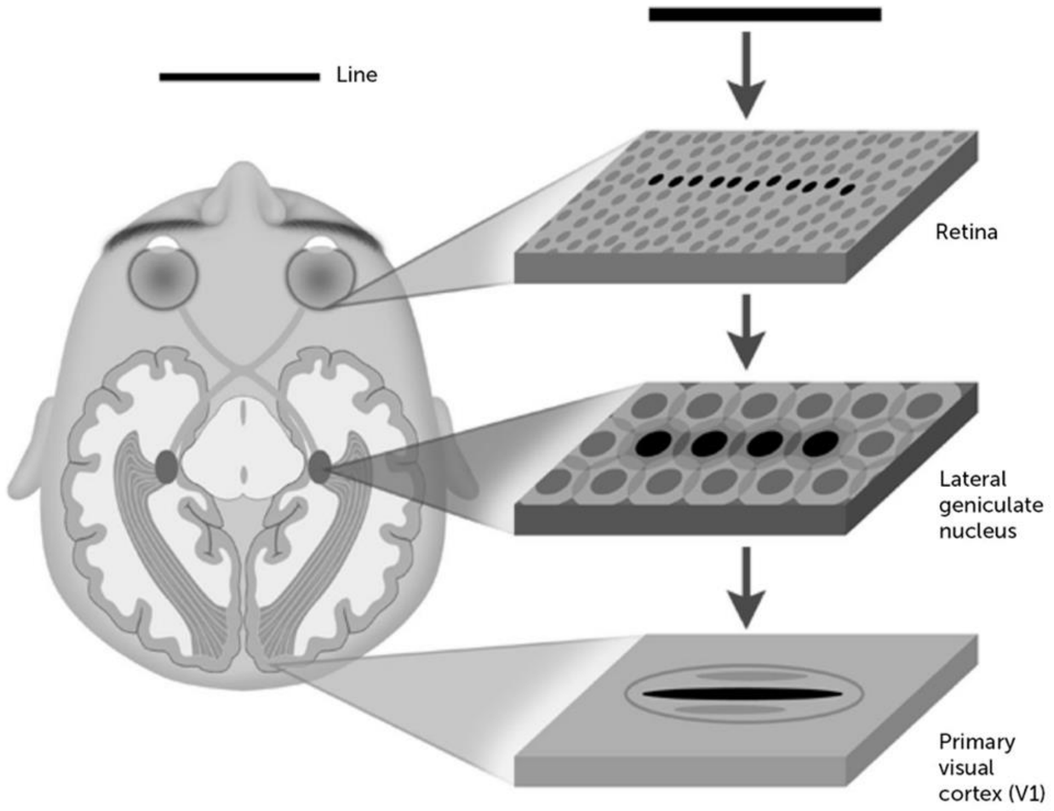
2D

3D

Receptive field

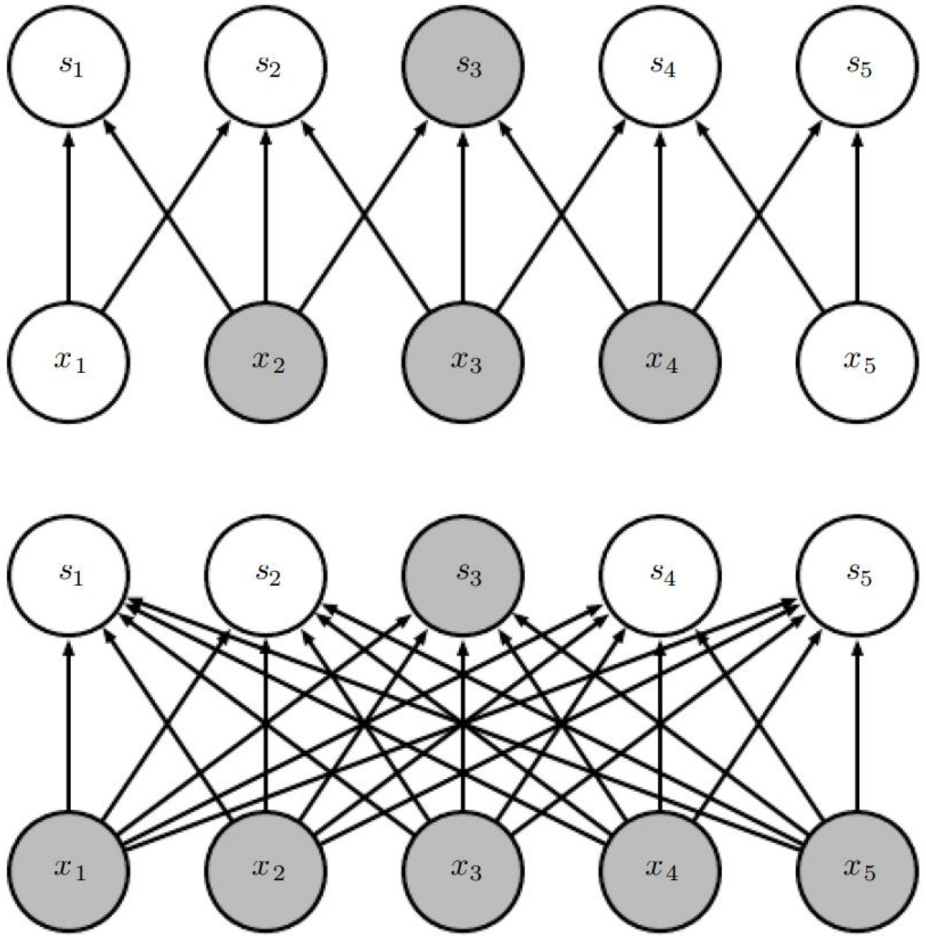
neuroscience / physics

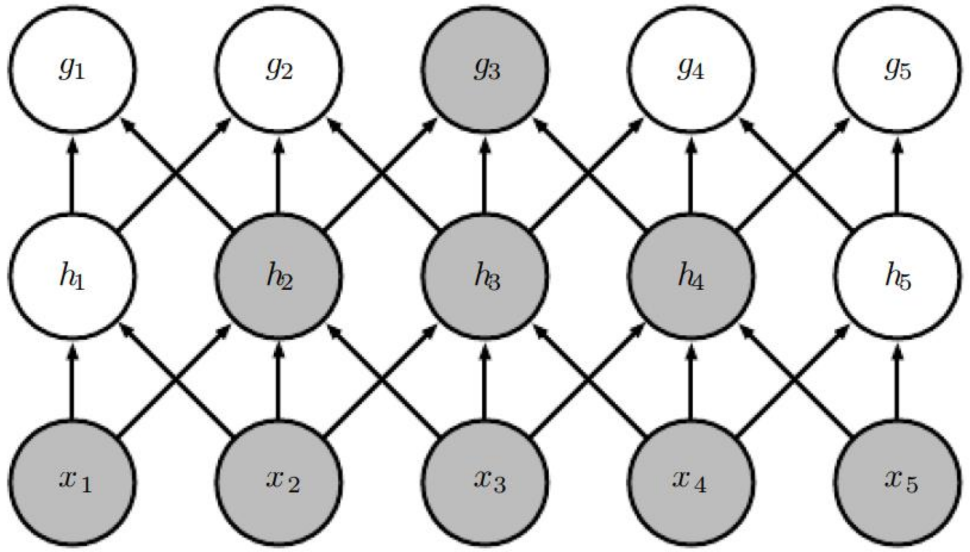




Receptive field

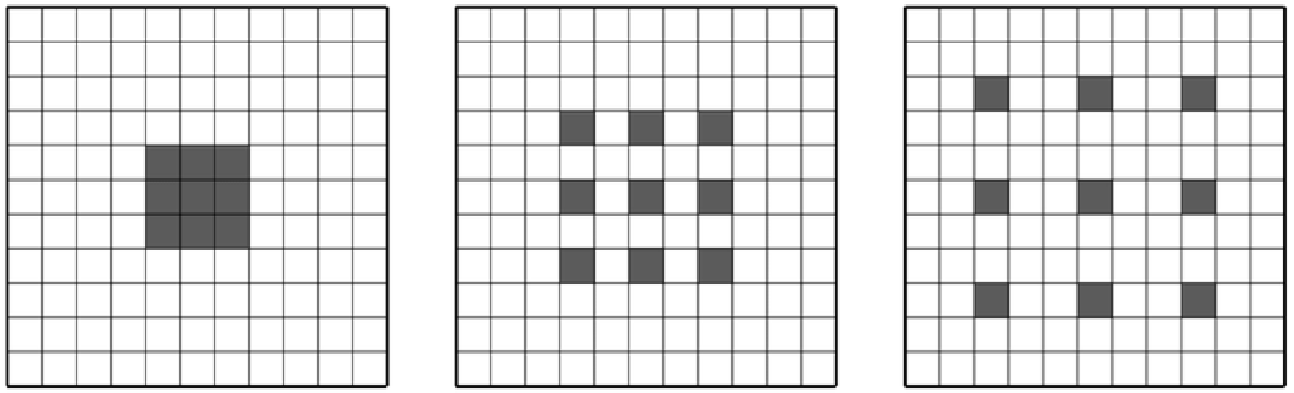
Fully-connected layers vs. convolutional layers vs. convolution with strides*

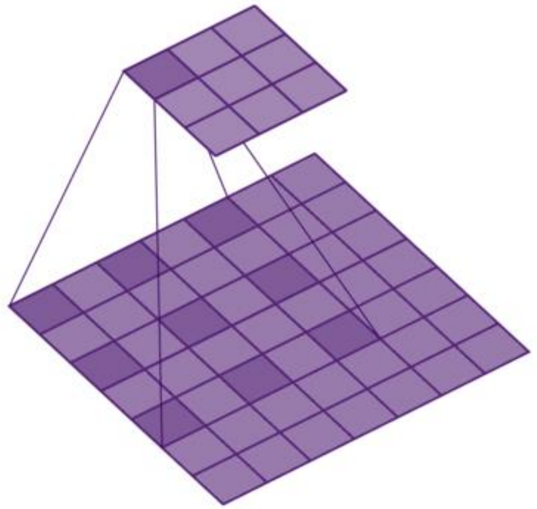




Receptive field

Dilated kernels (Atrous convolution)



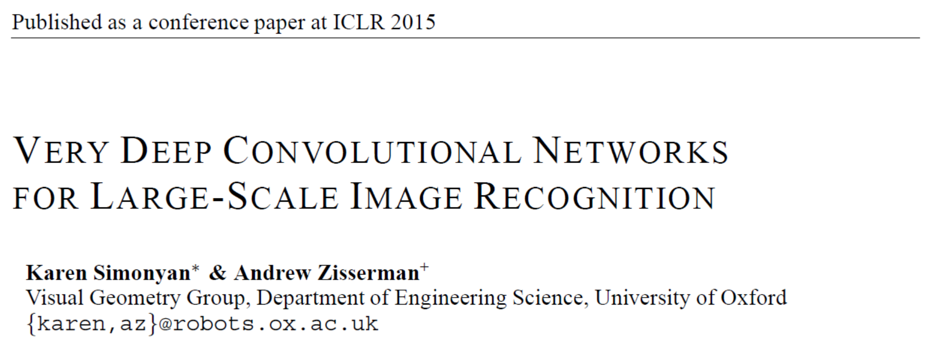


Convolutional Neural Networks | Kernels

Shape and size

- Dilated
- Adaptive*
- Lower-dimensional 2.5D kernels*

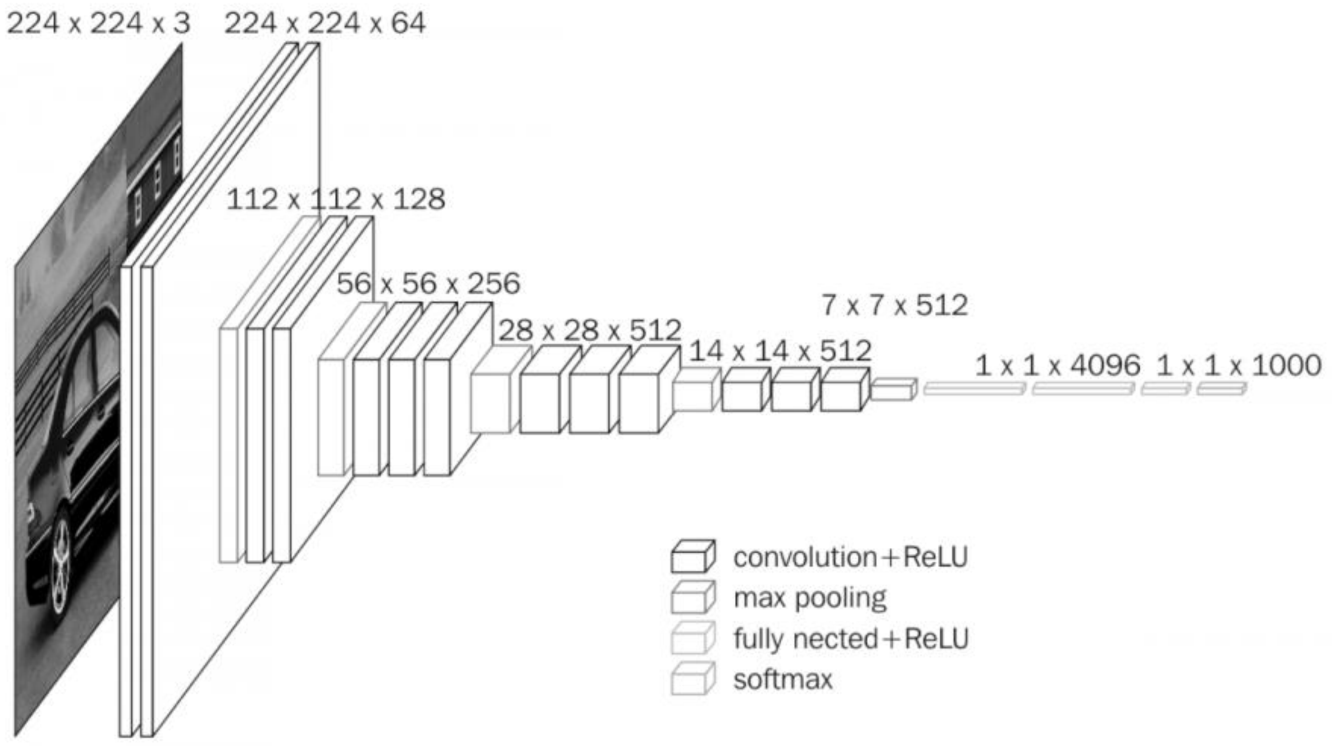
- VGG: 3x3 kernels
- Bottleneck 1x1 kernels vs. fully-connected layers



Published as a conference paper at ICLR 2015

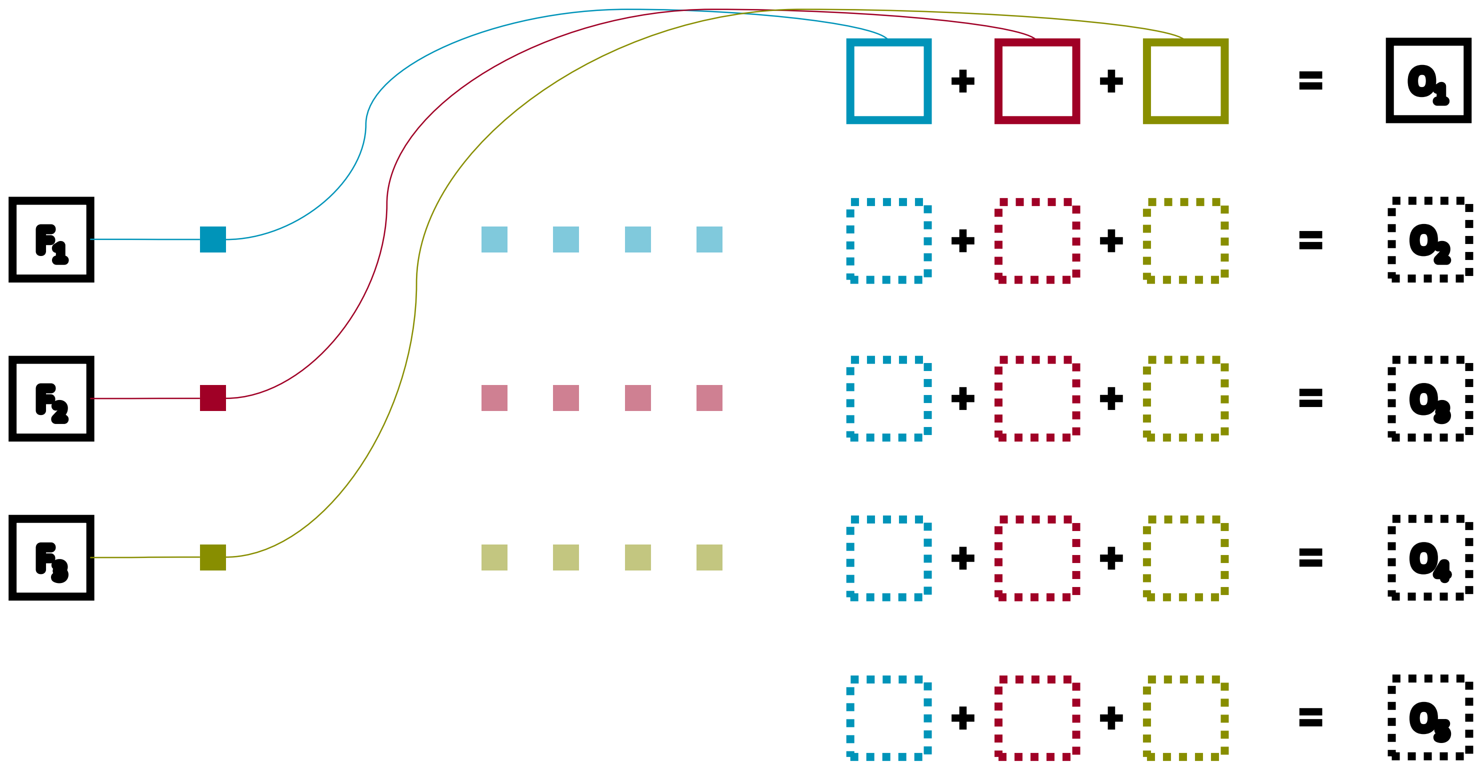
VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* & **Andrew Zisserman**+
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk



Shape and size

- 1x1 kernels
- “Channel-wise” linear projection (fully-connected layer)
- Dimensionality reduction for information “bottleneck” layer
- Upsampling



Kernel constraints

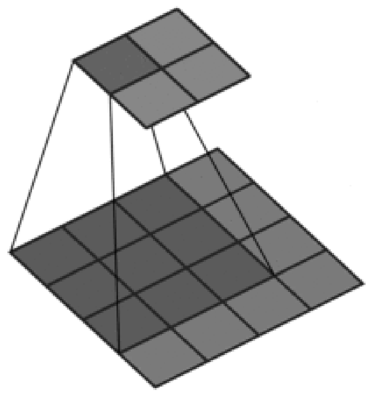
- Soft constraints: Regularise, e.g. L^2-norm*

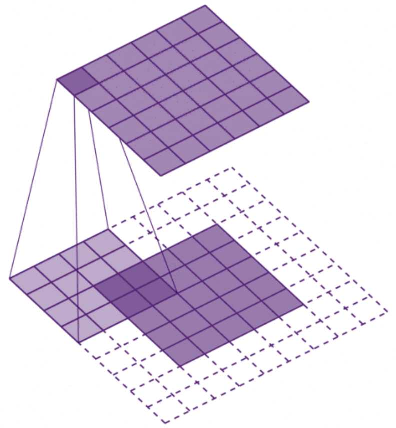
- Hard constraint: MaxNorm*

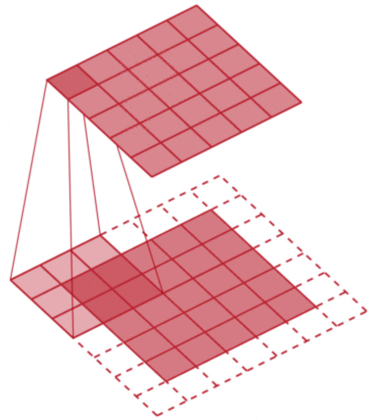
Padding

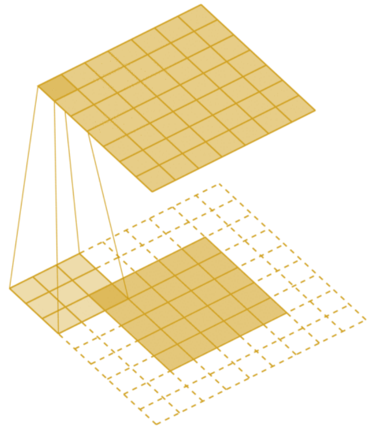
No padding, arbitrary padding, half padding, full padding

- No strides
- With strides*
- Transpose*
- Dilated*









https://github.com/vdumoulin/conv_arithmetic

Depthwise separable kernels

- Separable convolution

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$\mathbf{K} = \mathbf{u} * \mathbf{v}$

$\mathbf{I} * \mathbf{K} = \mathbf{I} * \mathbf{u} * \mathbf{v}$ (associativity)

Computational complexity

$O(M \times N \times 3 \times 3) \rightarrow O\big(M \times N \times (3 + 3)\big)$

$O(M \times N \times m \times n) \rightarrow O(M \times N \times (m + n))$

Singular value decomposition

$$\mathbf{K} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & S_3 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}^{\mathrm{T}}$$

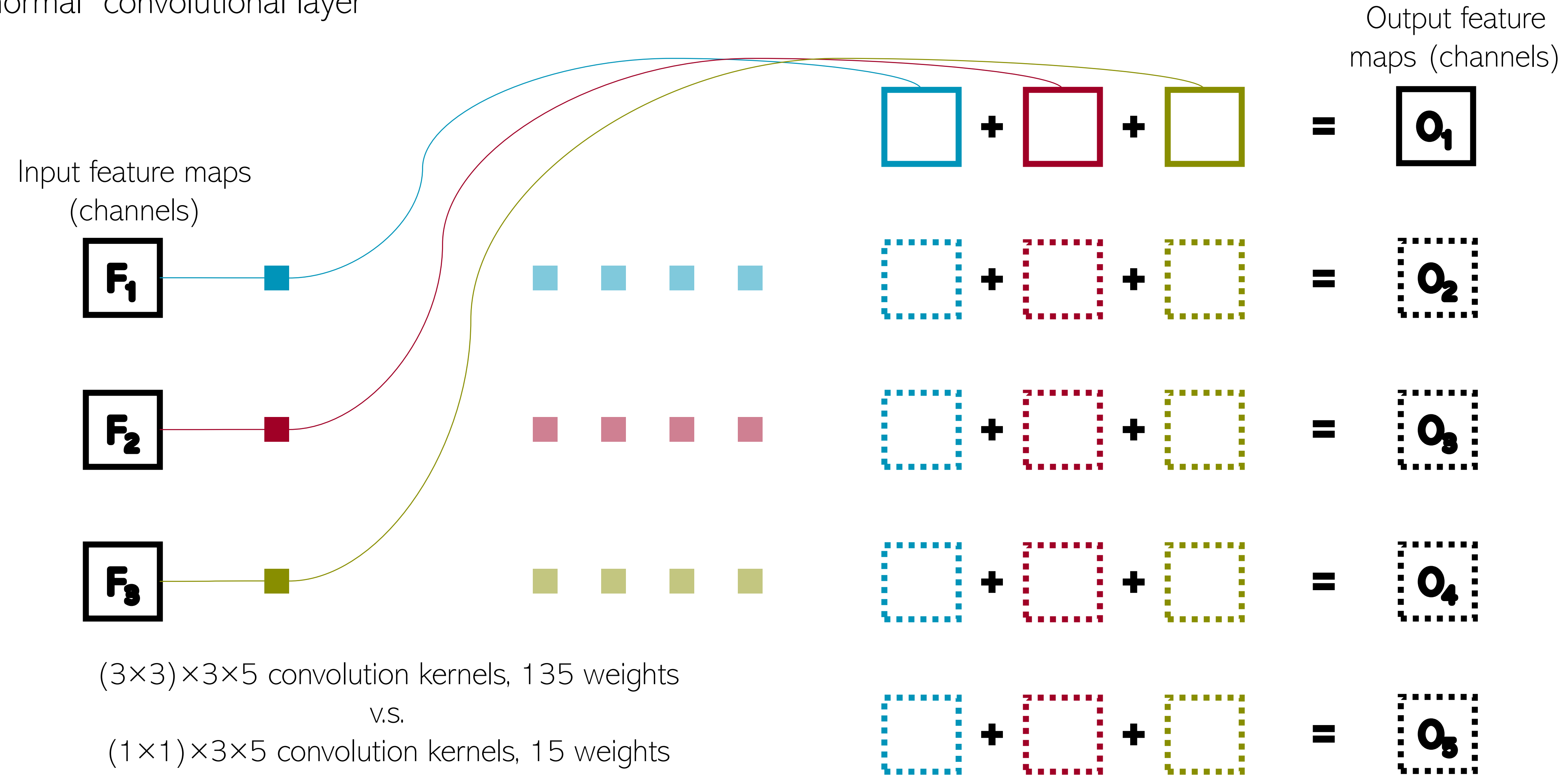
If $\mathbf{rank}(\mathbf{K}) = \mathbf{1}$ (test the number of non-singular values / linearly-independent vectors)

$\mathbf{K} = S_1 \mathbf{u}_1 \mathbf{v}_1^{\mathrm{T}} = S_1 \mathbf{u}_1 * \mathbf{v}_1^{\mathrm{T}}$ (definition of convolution)

Reduced degrees of freedom from 9 to 6

Depthwise separable kernels

- Revisit the "normal" convolutional layer



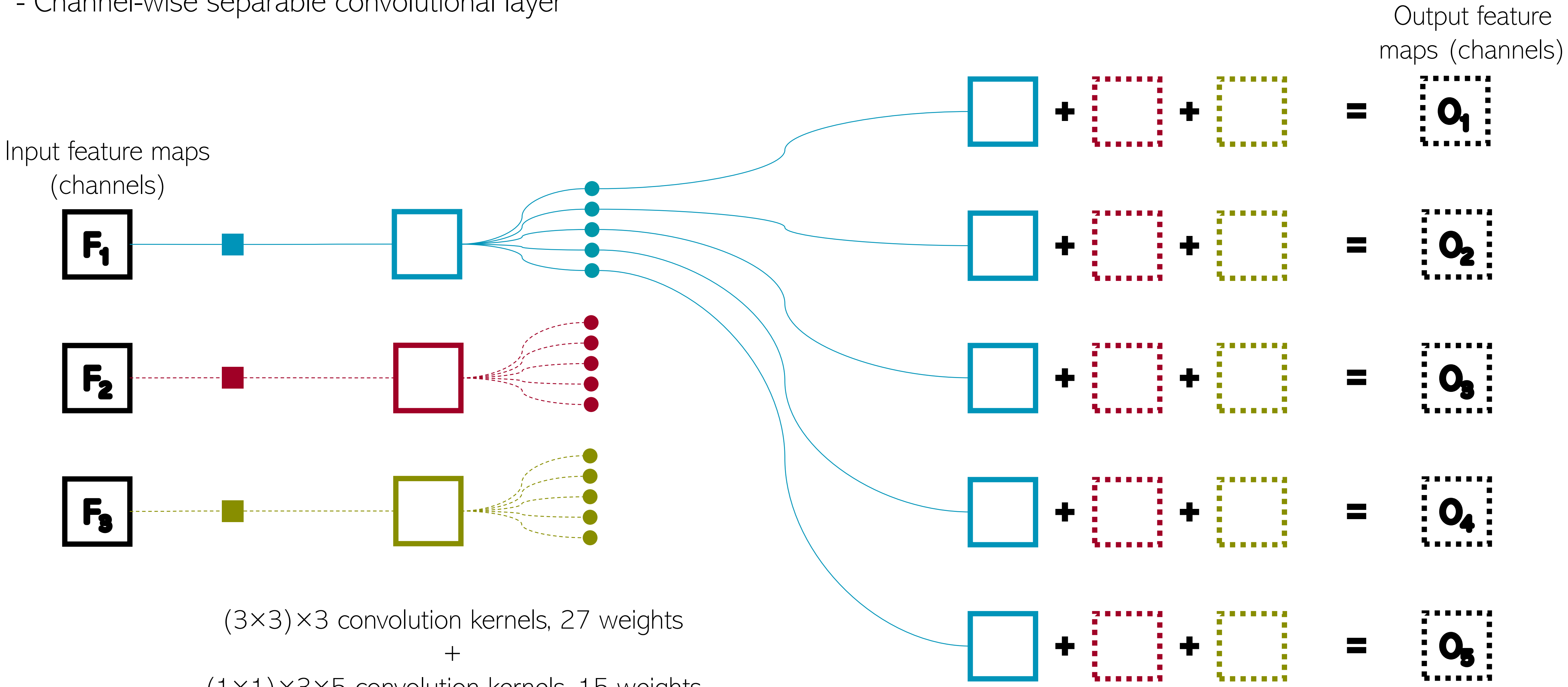
(3×3)×3×5 convolution kernels, 135 weights

v.s.

(1×1)×3×5 convolution kernels, 15 weights

Depthwise separable kernels

- Channel-wise separable convolutional layer



(3×3)×3 convolution kernels, 27 weights
+
(1×1)×3×5 convolution kernels, 15 weights

Depthwise separable kernels

- $(3\times3)\times3\times5=135$ vs. $(3\times3)\times3+(1\times1)\times3\times5=42$
- $(5\times5)\times64\times128=204{,}800$ vs. $(5\times5)\times64+(1\times1)\times64\times128=9{,}792$

- Model scaling
    * Width – number of (input and output) channels
    * Depth – number of layers
    * Resolution – feature map size

- "MobileNet", scaling vs. performance
- "EfficientNet", neural architecture search

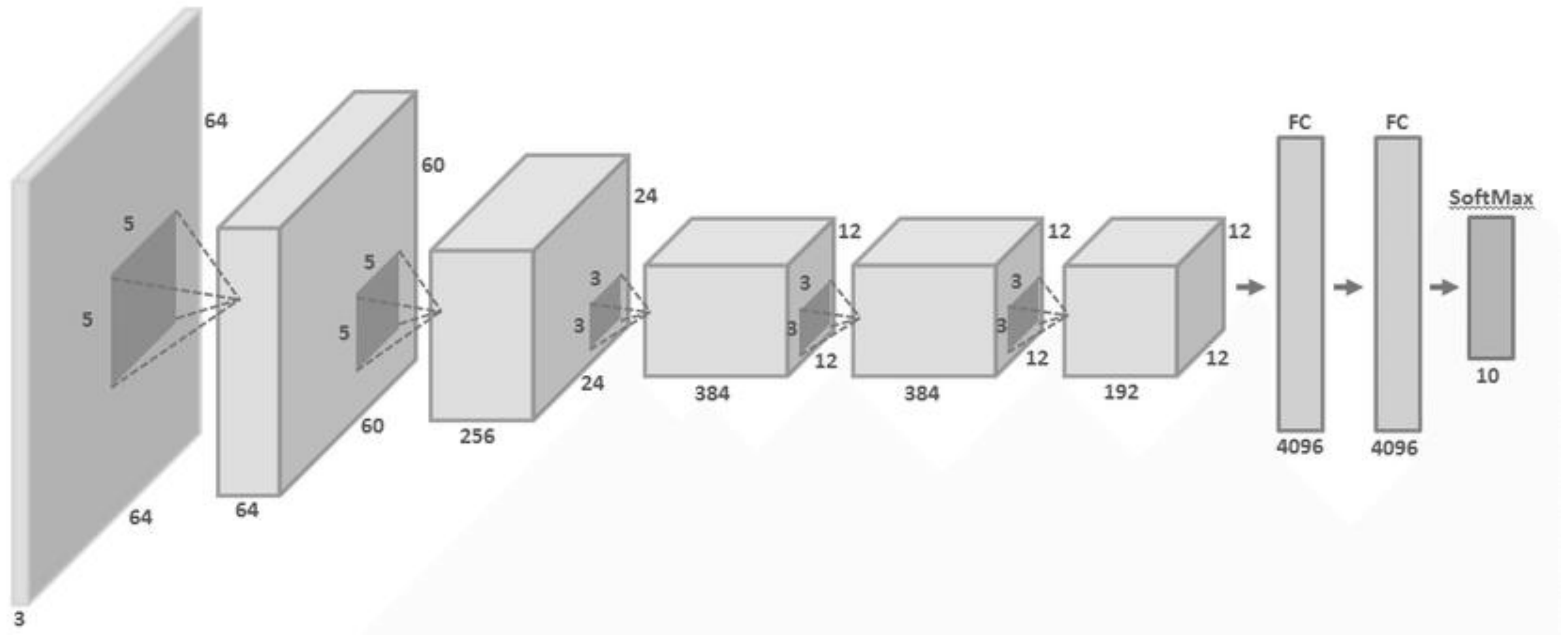
Convolutional Neural Networks | Convolution Arithmetic

Dumoulin, V. and Visin, F., 2016.
A guide to convolution arithmetic for deep learning.
arXiv preprint arXiv:1603.07285.
https://github.com/vdumoulin/conv_arithmetic

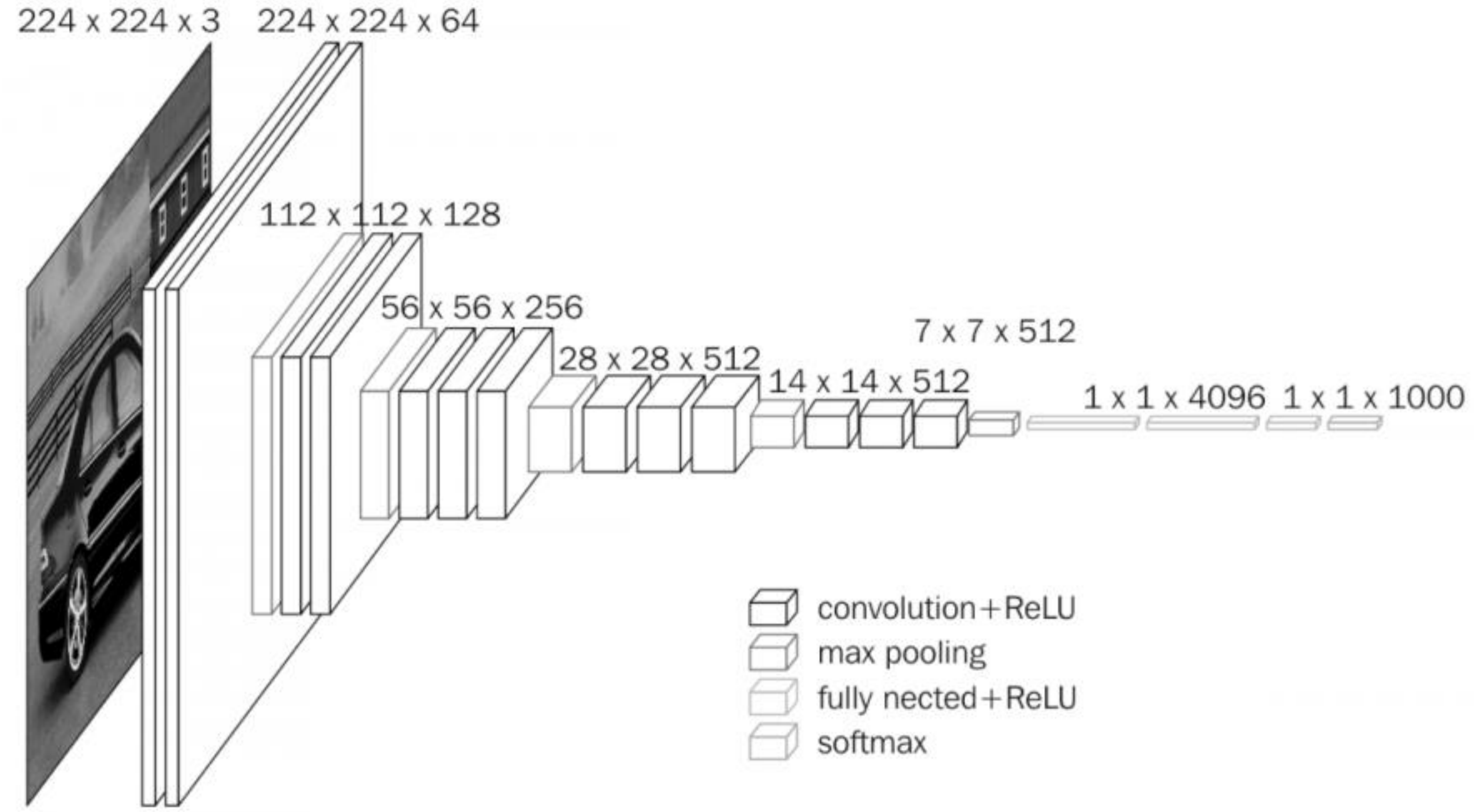
Things need to consider
- Input/output feature map size and even/odd
- Kernel size and even/odd
- Strides
- Padding
- Kernel dilation rate
- Transpose convolution
- Pooling

Convolutional Neural Networks | Applied Architectures

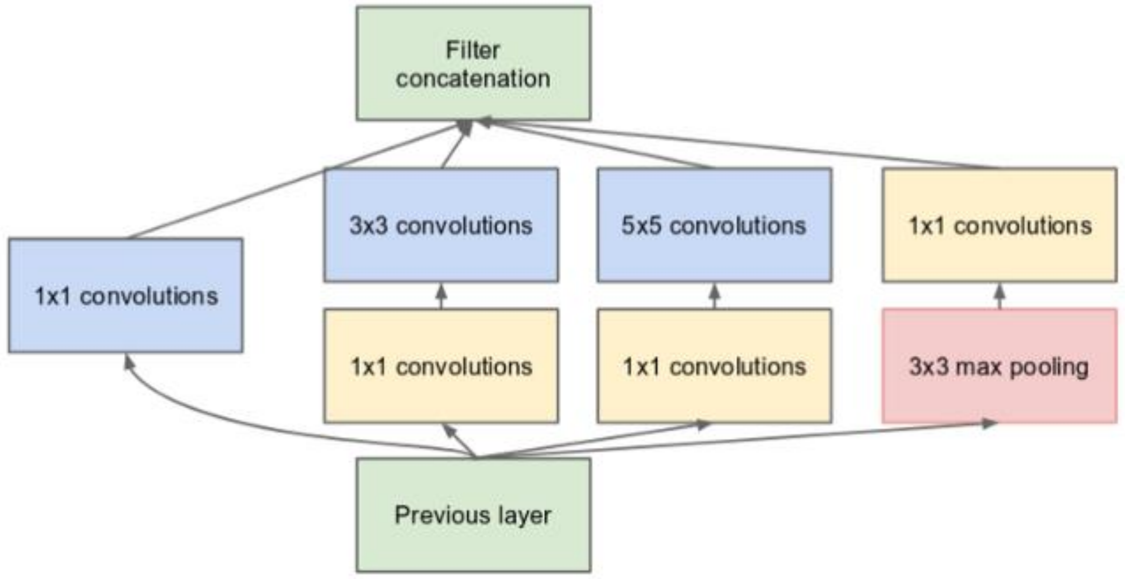
AlexNet

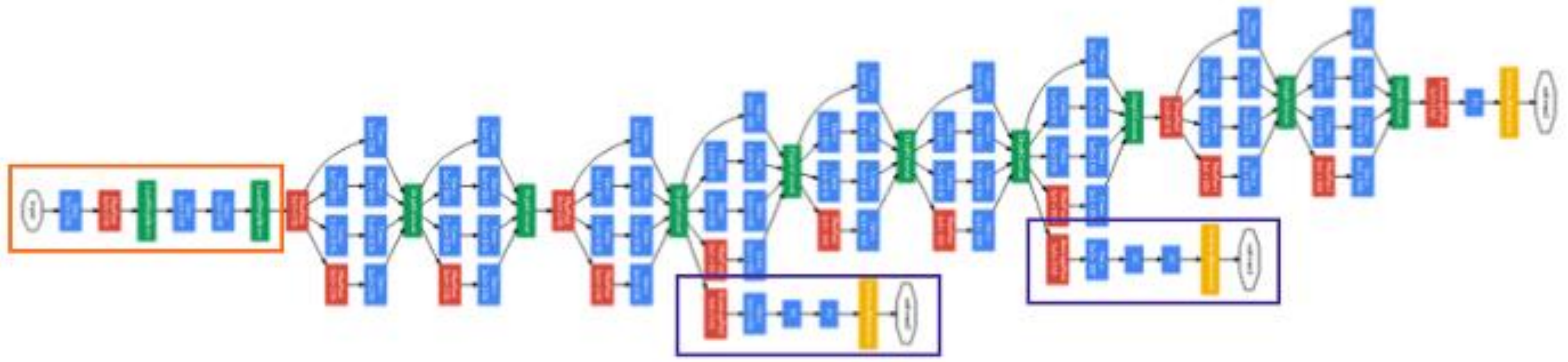

64
64
3
5
5
60
60
64
5
5
24
24
256
3
3
12
12
384
3
3
12
12
384
3
3
12
12
192
FC
4096
FC
4096
SoftMax
10

VGG

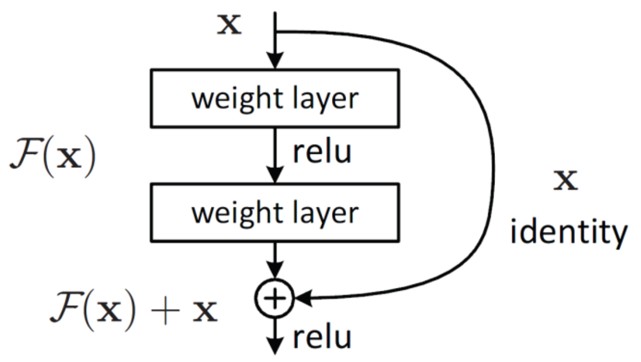

224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

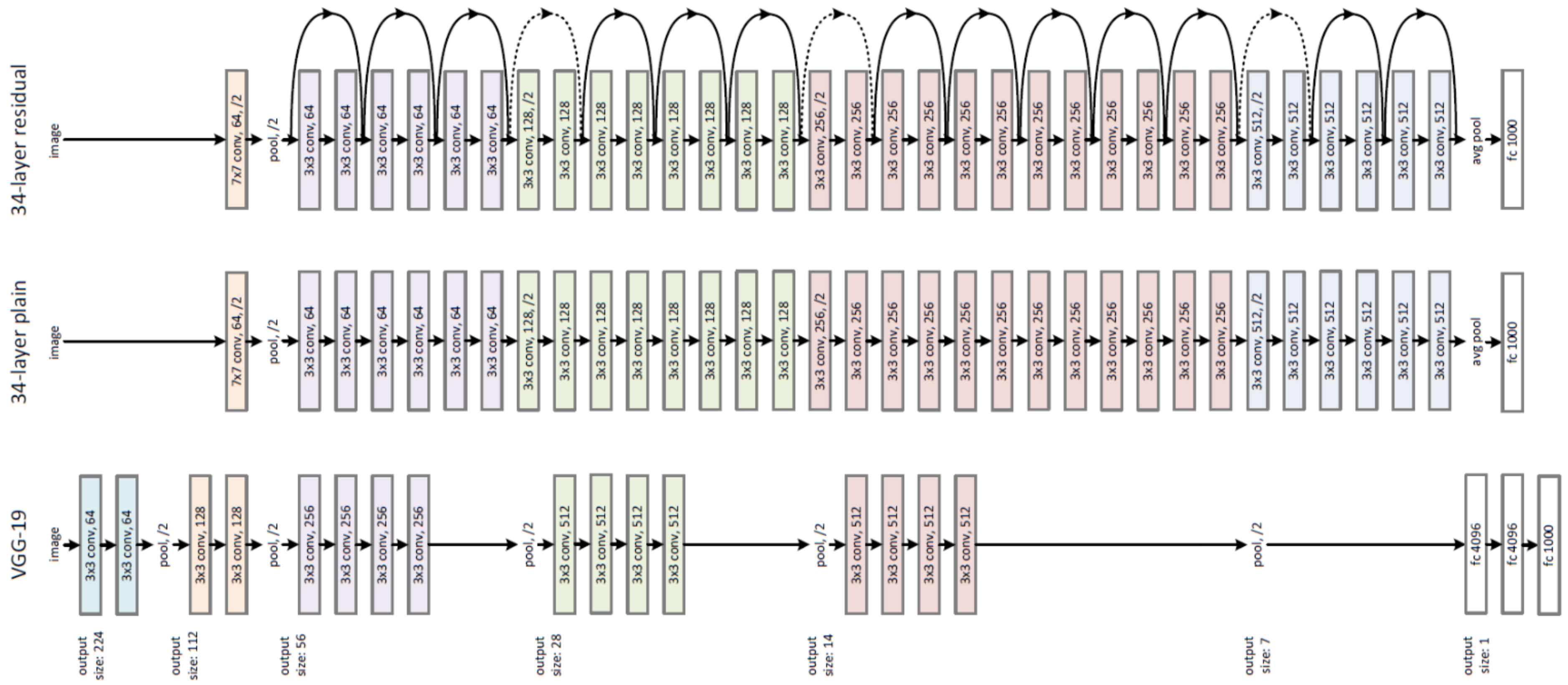
Inception


Filter concatenation
1x1 convolutions
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer

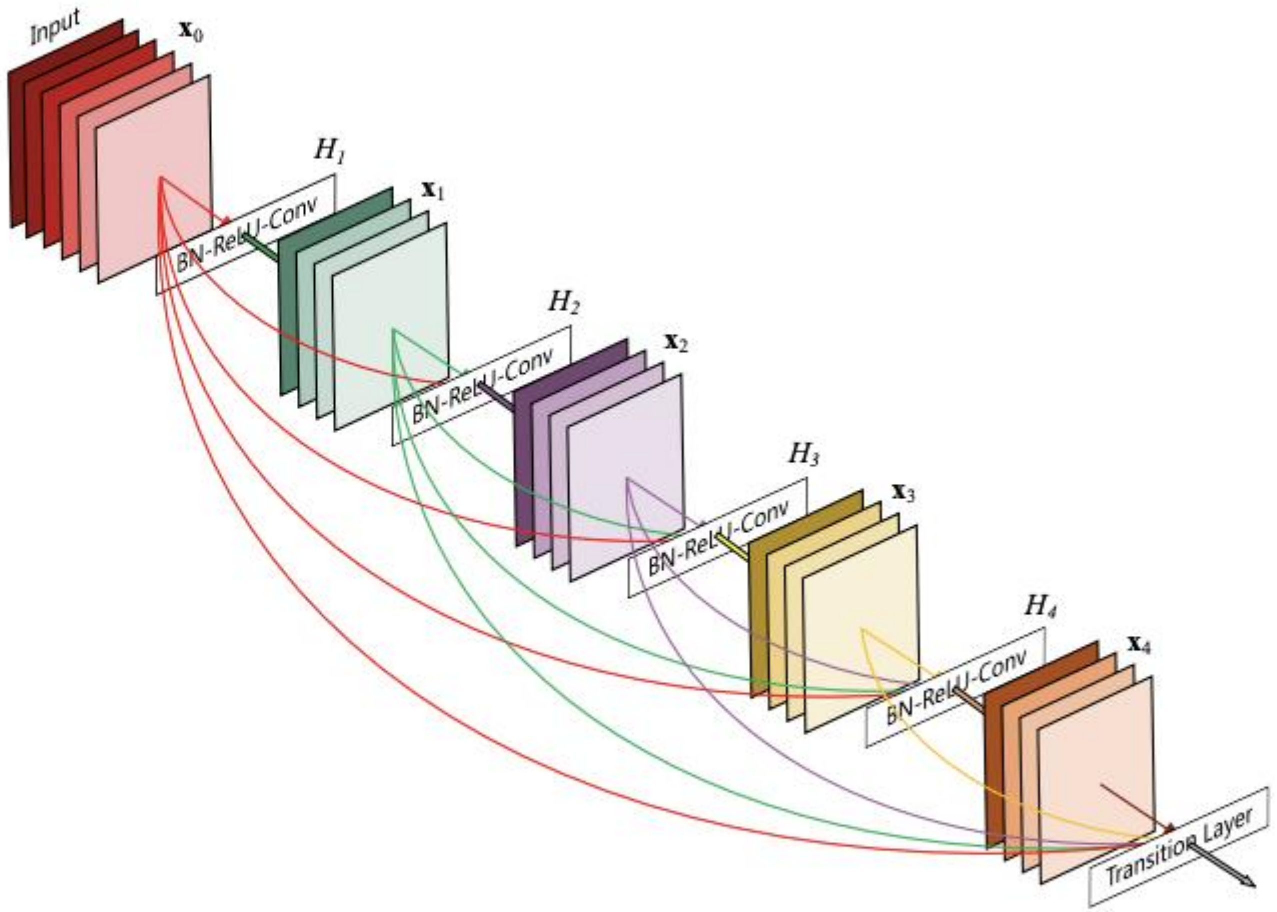


ResNet

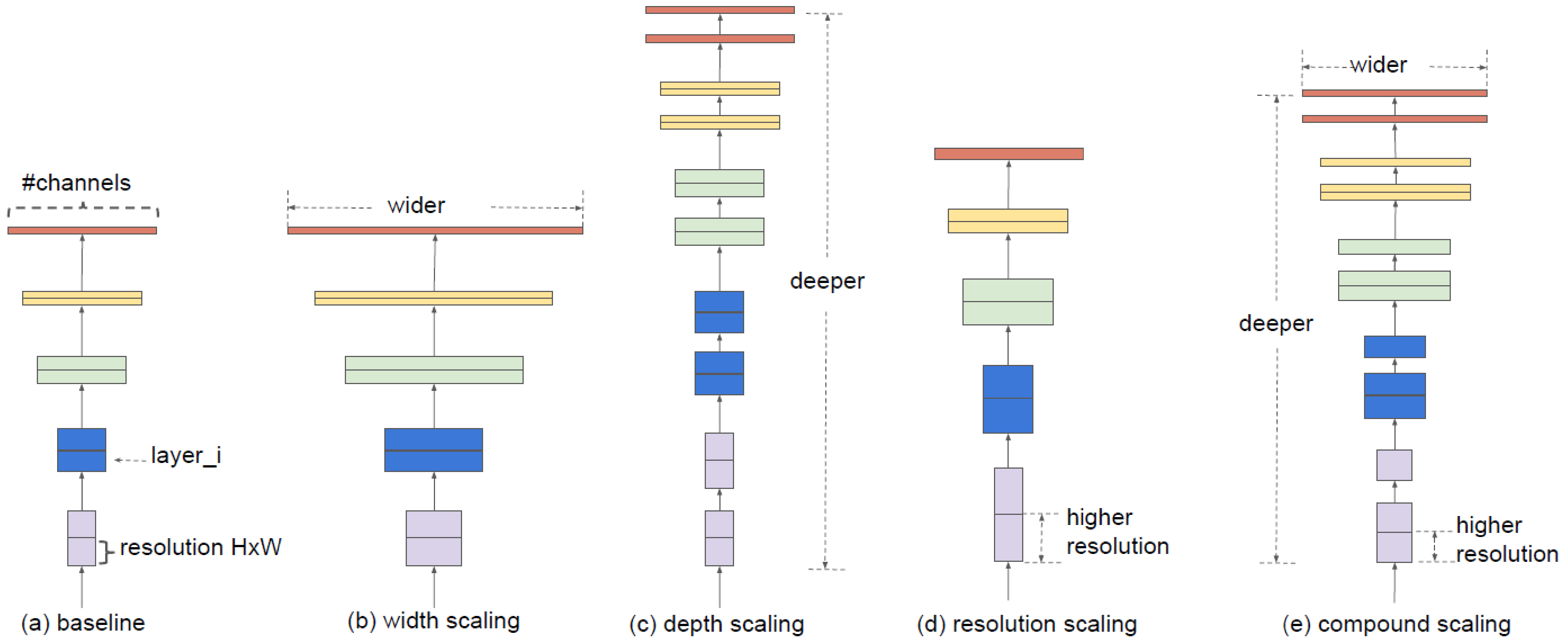

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
x
identity


34-layer residual
34-layer plain
VGG-19
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000
fc 4096
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1

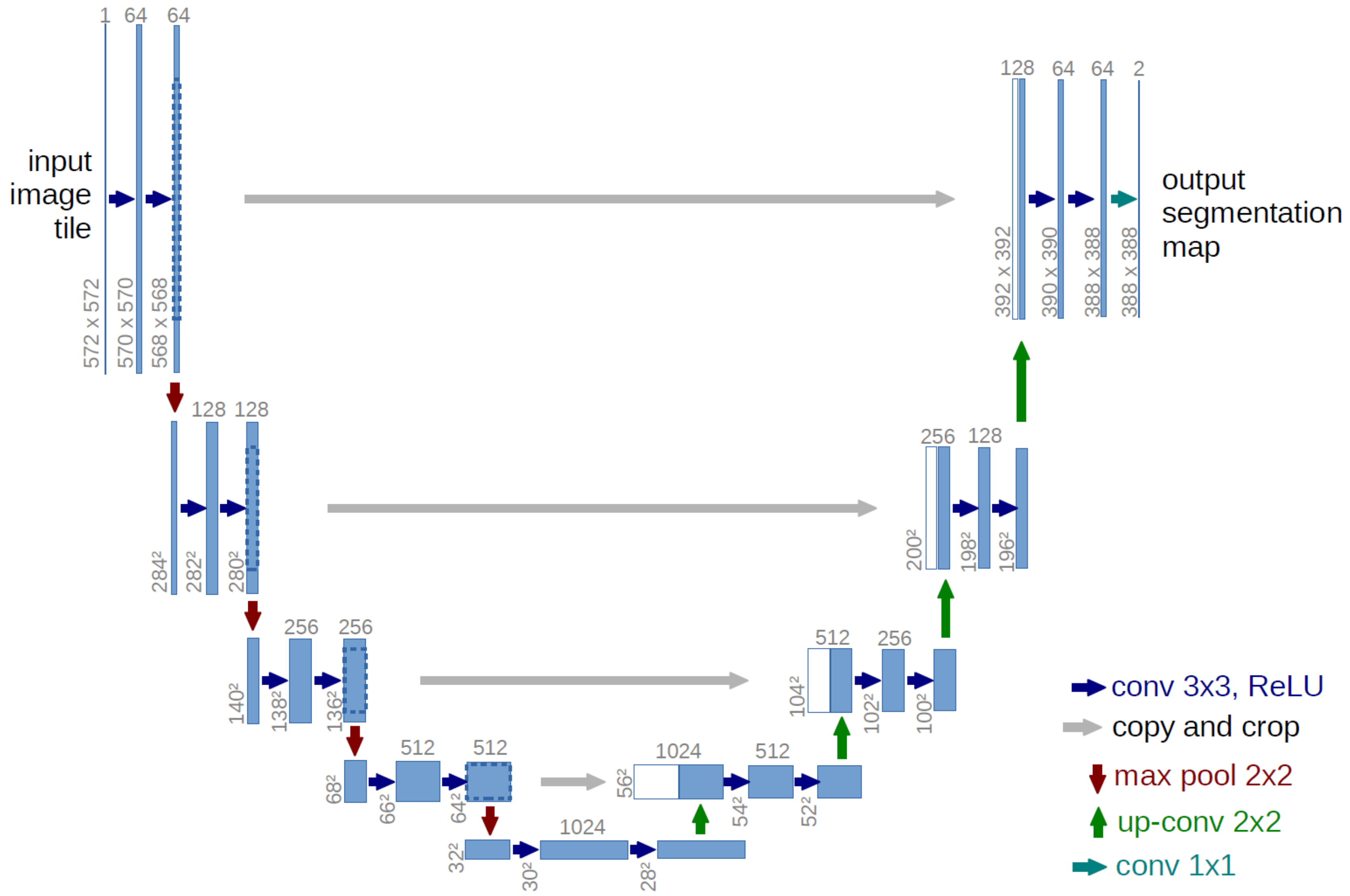
DenseNet


Input
$\mathbf{x}_0$
H_1
BN-ReLU-Conv
$\mathbf{x}_1$
H_2
BN-ReLU-Conv
$\mathbf{x}_2$
H_3
BN-ReLU-Conv
$\mathbf{x}_3$
H_4
BN-ReLU-Conv
$\mathbf{x}_4$
Transition Layer

EfficientNet



UNet


1
64
64
input image tile
572 x 572
570 x 570
568 x 568
128
128
284²
282²
280²
256
256
140²
138²
136²
512
512
68²
66²
64²
1024
32²
30²
28²
1024
512
56²
54²
52²
512
256
104²
102²
100²
256
128
200²
198²
196²
128
64
64
2
392 x 392
390 x 390
388 x 388
388 x 388
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Convolution and cross-correlation

Intuition

Translation-invariance, Parameter sharing, Sparse weights, Infinitely strong prior

Sampling

Pooling, Strides, Interpolation, Un-pooling, Transpose convolution

Anatomy of a convolutional layer

Kernels

Dimensionality, receptive field, shape and size, kernel constraints, padding, depthwise separable convolution

Convolution arithmetic

Applied architectures

Using a different CNN for the “image classification” tutorial
Adding/reducing a resolution level in the “image segmentation” tutorial