BIOS 7747: Machine Learning for Biomedical Applications

Convolutional neural networks

Antonio R. Porras (antonio.porras@cuanschutz.edu)

Department of Biostatistics and Informatics
Colorado School of Public Health
University of Colorado Anschutz Medical Campus

Outline

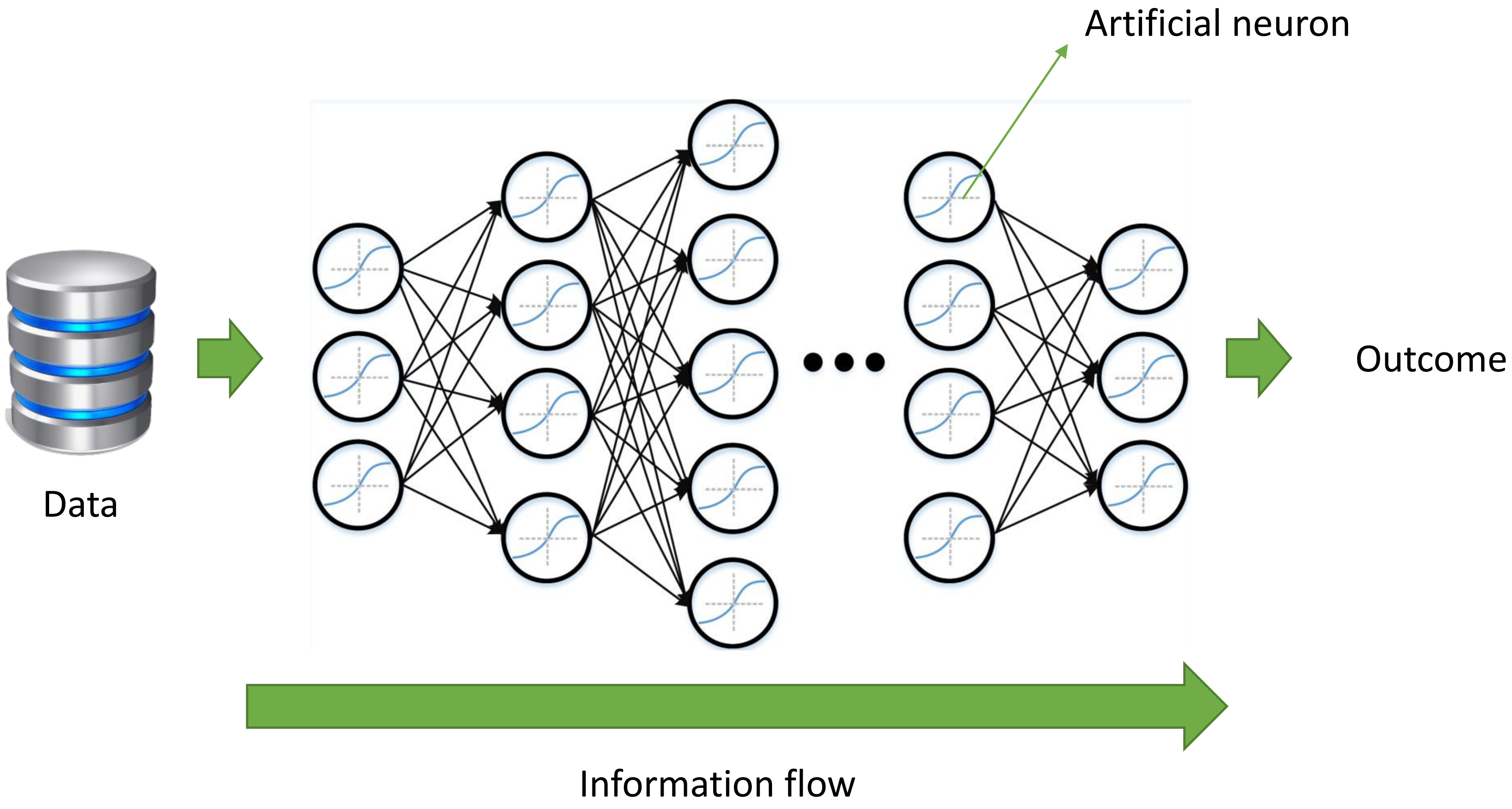
- Introduction to convolutional neural networks
  - Convolutions
  - Downsampling
  - Activation
  - Full architecture
  - Training

- Architectures and basic design concepts

- Fully convolutional networks

Introduction to convolutional neural networks

- In previous class: artificial neural networks



Introduction to convolutional neural networks

- What happens if we have too much data?

  - Thousands or millions of observations?
    - It does not affect the network architecture
    - It will likely decrease overfitting and build more robust models

  - Thousands or millions of features?
    - It will require a much wider network
    - It will likely increase overfitting (many more weights)

Introduction to convolutional neural networks

- Images (and temporal signals) usually have a high number of pixels (temporal samples)
  - 2D images with size 200x200: 40,000 features per image
  - 3D images with size 200x200x200: 8,000,000 features per image
  - Signal sampled at 120Hz for 5 minutes: 36,000 features per signal

- Number of parameters needed only in the first hidden layer:

$$\underbrace{\#Neurons * \#Features}_{w} + \underbrace{\#Neurons}_{b}$$

- To evaluate one image using a network with only 1,000 neurons in one hidden layer we would need :

$$4 * (\underbrace{40{,}000}_{\text{image}} + \underbrace{1{,}000 * 40{,}000 + 1{,}000}_{\text{hidden layer}} + \underbrace{1{,}000 * 1 + 1}_{\text{output neuron}}) \approx 160MB$$

- More realistic scenario of a <u>narrow</u> 8-layer network :
  - $4 * (40{,}000 + 40{,}000 * 40{,}000 + 40{,}000 + 40{,}000 * 30{,}000 + 30{,}000 + 30{,}000 * 20{,}000 + 20{,}000 + 20{,}000 * 10{,}000 + 10{,}000 + 10{,}000 * 2{,}000 + 2{,}000 + 2{,}000 * 512 + 512 + 512 * 128 + 128 + 128 * 1 + 1)$ ≈14GB

Introduction to convolutional neural networks

- Memory needs:

Memory for model parameters

Memory for outputs

Memory for parameter gradients

Memory for error and losses

Memory for optimizer's momentum

Memory for operations

Library overhead, other resources, OS management, etc.

Convolutions

- Spatial operation that aggregates the information in a specific neighborhood

- Defined using a base kernel and a convolution operation

weights

Spatial kernel:

$w_{-1,-1}$	$w_{-1,0}$	$w_{-1,1}$
$w_{0,-1}$	$w_{0,0}$	$w_{0,1}$
$w_{1,-1}$	$w_{1,0}$	$w_{1,1}$

Neighborhood: 3x3

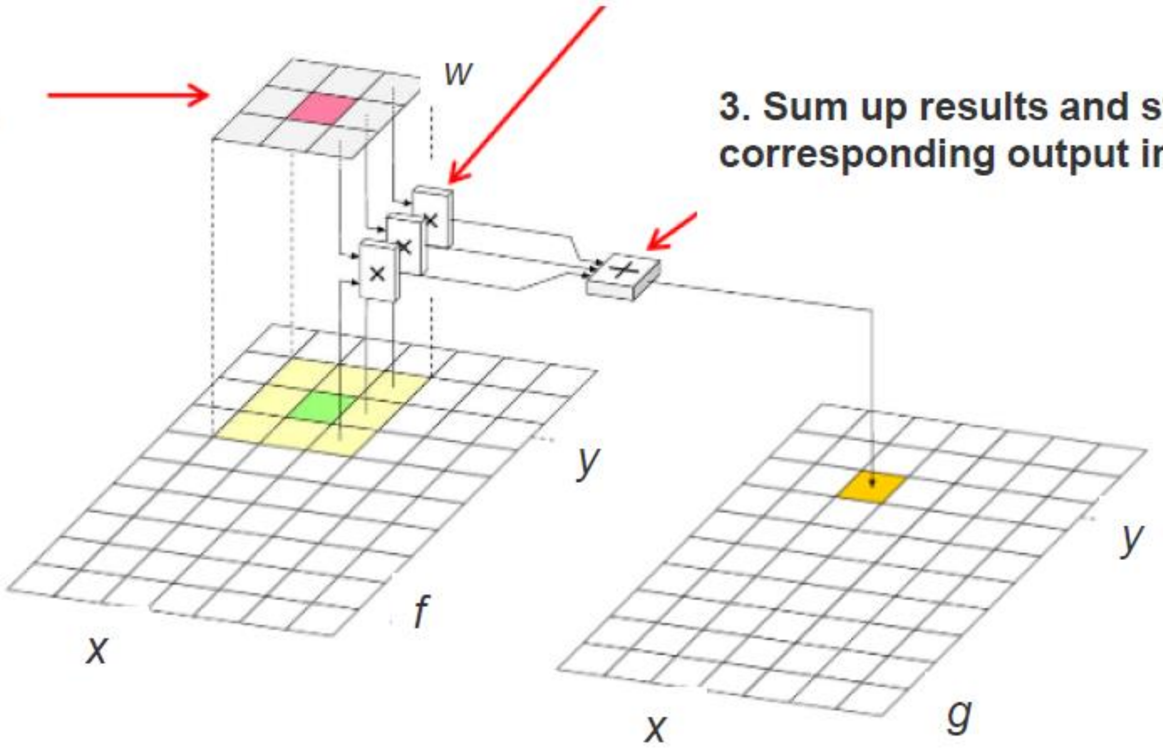
Convolutions

For each image position (x,y):

1. Move filter matrix *w* over image such that *w*(0,0) coincides with current image position (x,y)

2. Multiply all filter coefficients *w(s,t)* with corresponding pixel *f(x+s,y+t)*

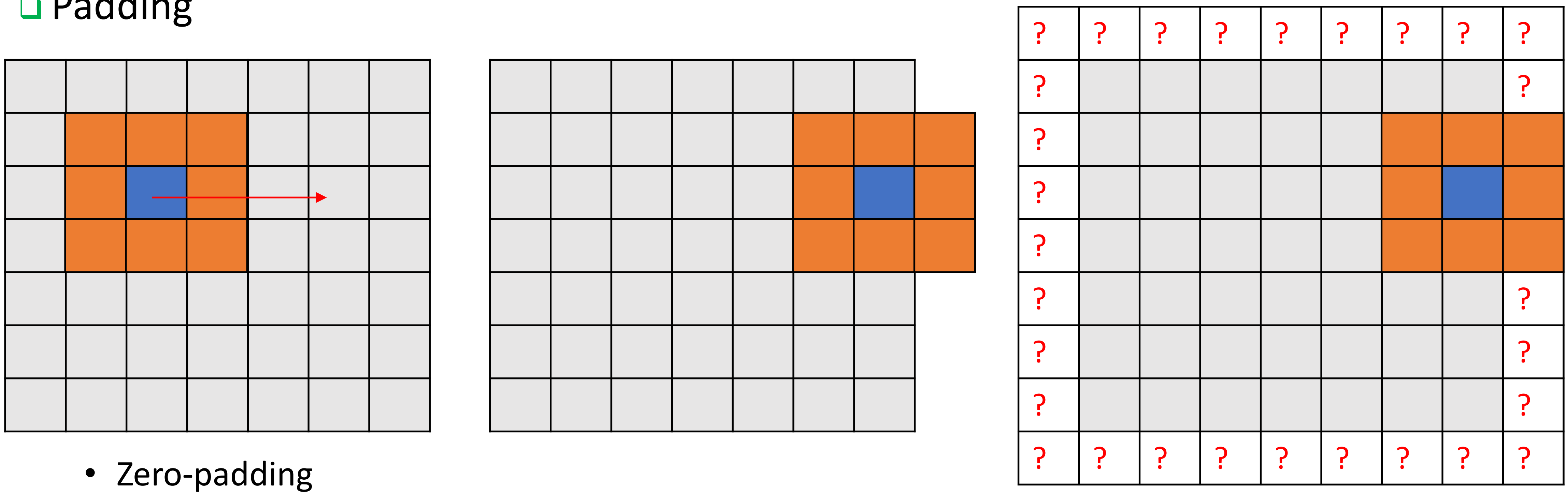
3. Sum up results and store sum in corresponding output image g(x,y)



Cross-correlation of image *f* and filter *w*: $g(x,y) = \sum_{i,j=-1}^{1} f(x+i, y+j) * w(i,j)$

Convolutions

- Padding



- Zero-padding
- Mirroring
- Replication
- Circle-padding
- Border removal

Convolutions

- Visual (or temporal) patterns consists in:
  - Intensity value in a continuous region (brighter or darker)
  - Intensity change between regions (edge strength)

- Convolutions can provide information about:
  - Regional intensity
    [aka smoothing or (weighted) average filter]

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  - Edge information
    [aka sharpening or differential filter]

$$\begin{bmatrix} \frac{-1}{8} & \frac{-1}{8} & \frac{-1}{8} \\ \frac{-1}{8} & 1 & \frac{-1}{8} \\ \frac{-1}{8} & \frac{-1}{8} & \frac{-1}{8} \end{bmatrix}$$

Convolutions

- Convolution vs. cross-correlation

Cross-correlation: $g(x, y) = \sum_{i,j=-1}^{1} f(x+i, y+j) * w(i, j)$

Convolution: $g(x, y) = \sum_{i,j=-1}^{1} f(x+i, y+j) * w(-i, -j)$

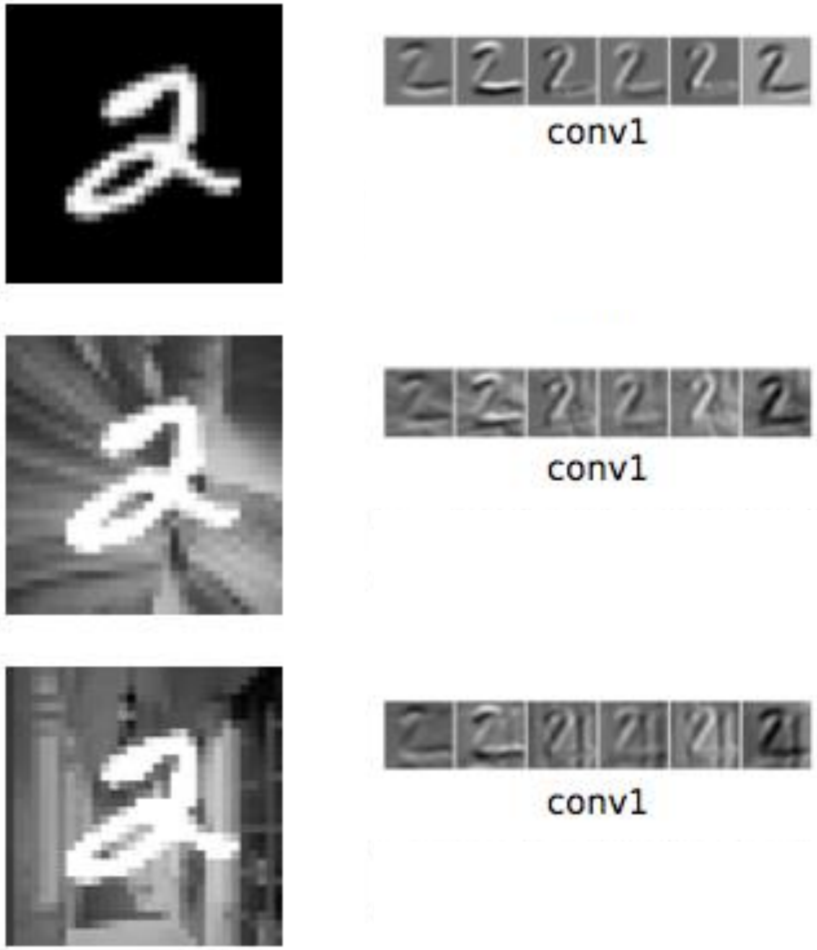
- To implement a convolution, a 180° rotation must be applied to the kernel

- They are only equivalent in symmetric kernels

- Most libraries implement cross-correlations, not convolutions

Convolutions

- Example of convolutions



Convolutions

- Convolutional neural networks use convolutional filters to calculate spatial or temporal features
  - Convolution operations replace the linear decision function of the perceptron

$$z = wx + b \quad \Rightarrow \quad z = w * I + b$$

- Comparison:
  - Parameters required for linear function: $\#Neurons \text{ x } \#Features + \#Neurons$
  - Parameters required for convolution: $\#Filters \text{ x } FilterSize + \#Filters$

- The number of convolution parameters does not depend on the spatial size (or signal length)

Convolutions

- Example: Extraction of 10 features of a 200x200 image

  - Fully connected layer with 10 neurons:
    - Number of parameters: 10 x $(200 \mathrm{~x}\ 200) + 10$ = 400,010 parameters
    - Every perceptron combines all image information (most information will not be relevant so there are very high chances of overfitting)
    - Output: 10 features

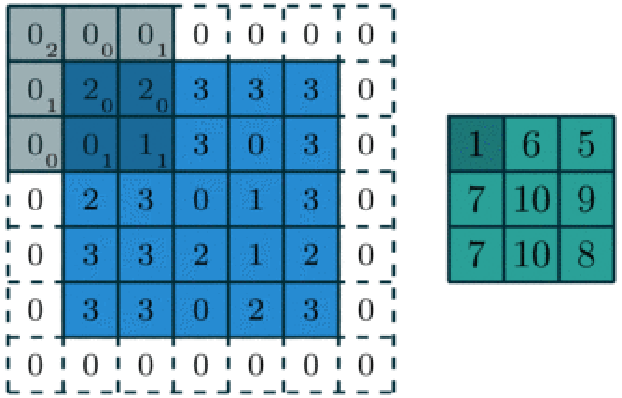
  - Convolutional network with 10 filters:
    - Region size: 5x5 neighborhood
    - Number of parameters 10 x $25 + 10$ = 260 parameters
    - Every feature combines only regional information
    - Output: 10 features at each pixel (10x200x200)

Downsampling

- Downsampling: reduces the amount of data (dimensionality reduction)
  - Increases robustness to slight changes in rotation and translation
  - Convolutions of lower resolution images with same kernels aggregate information from larger regions

- Two main approaches to downsampling

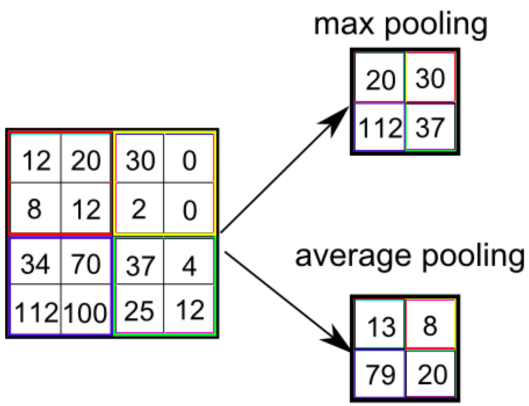
<u>Convolution with strides</u>



2x2 strides

- Faster

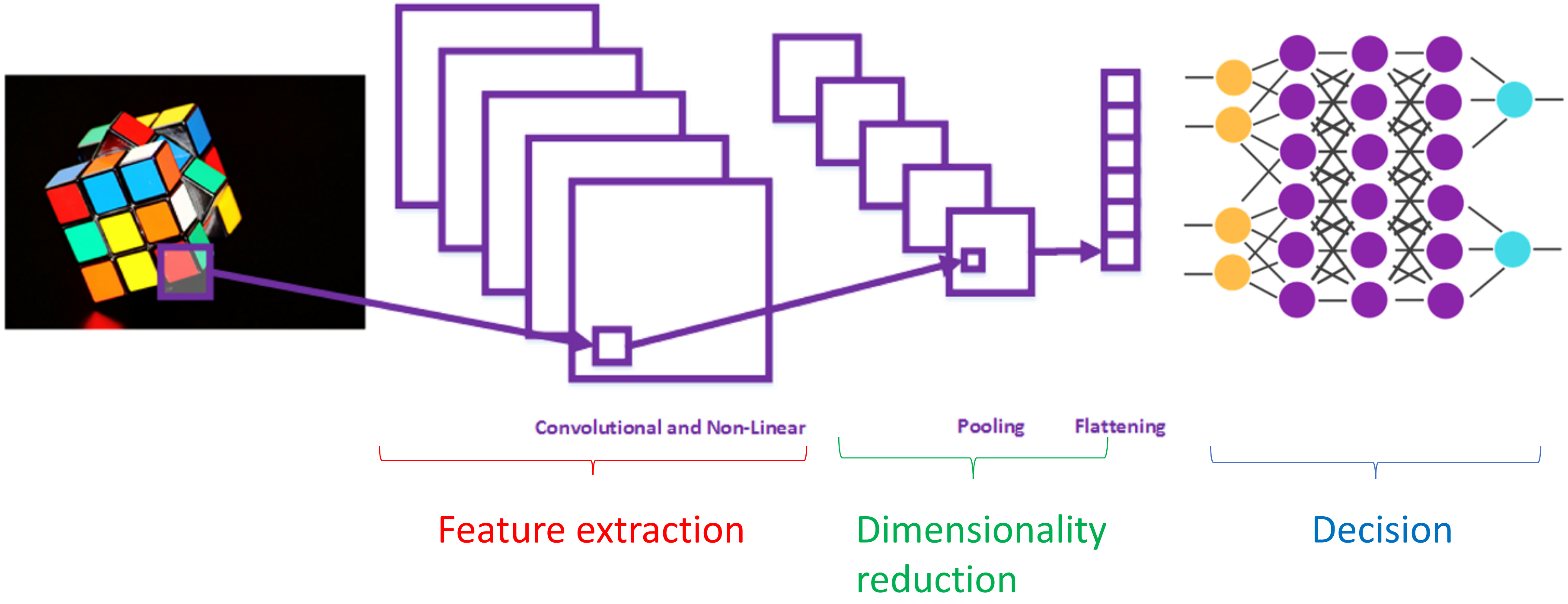
<u>Pooling (after convolution)</u>



2x2 pooling with 2x2 strides

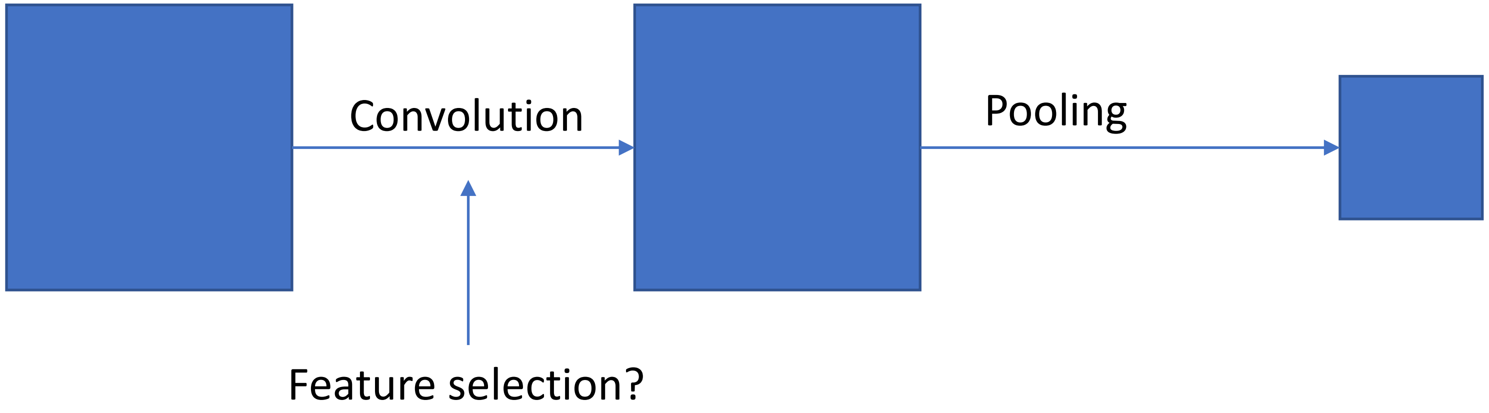
Downsampling

❑ Typical convolutional neural network

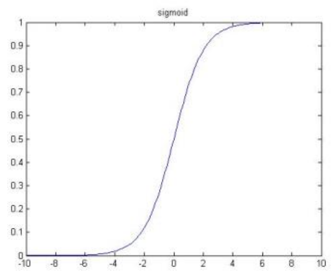


Activation

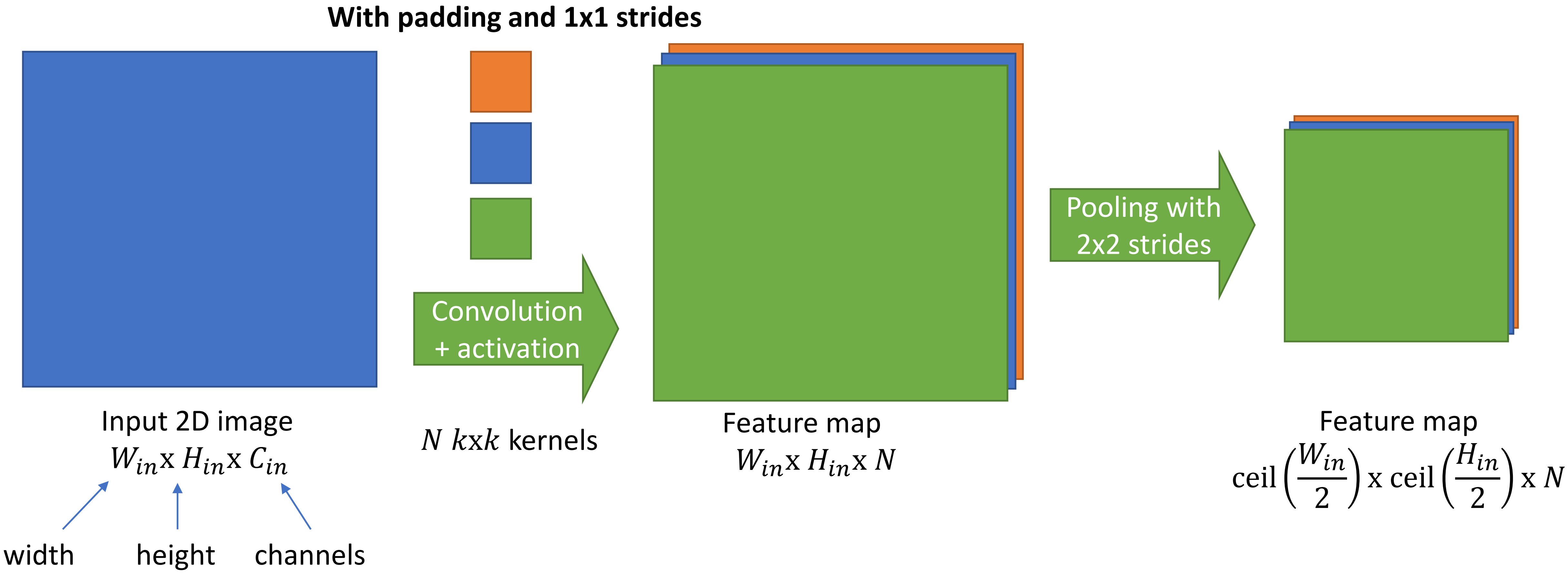
- Although pooling can eliminate meaningless features, it needs to either choose between different potentially meaningful features (max pool) or aggregate potentially meaningless features (average pooling)



- The activation function
  - An activation enables kernels to learn how to zero-out irrelevant spatial patterns

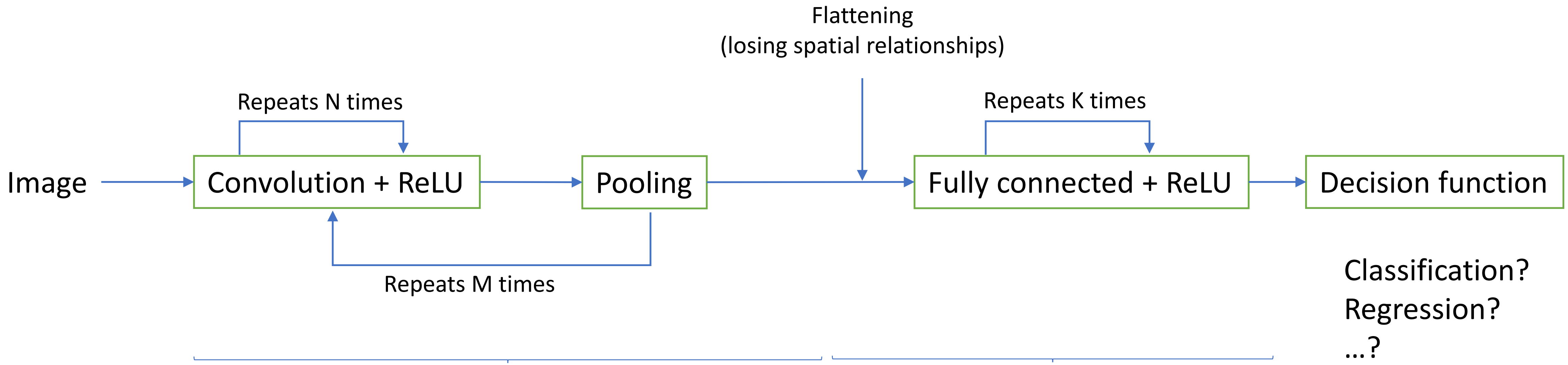


Full architecture



Number of parameters: $\boldsymbol{N\text{x}(k \text{ x } k \text{ x } C_{in} + 1)}$

Full architecture



Feature expansive : number of features increase as spatial dimensions decrease

Reduction: feature selection/reduction and decision making

Similar to feature quantification and aggregation

Similar to supervised/unsupervised model training

Training

- How can we backpropagate a ~~convolution~~ cross-correlation operation?

$$z = w * I + b$$

Single channel

3x3 kernel

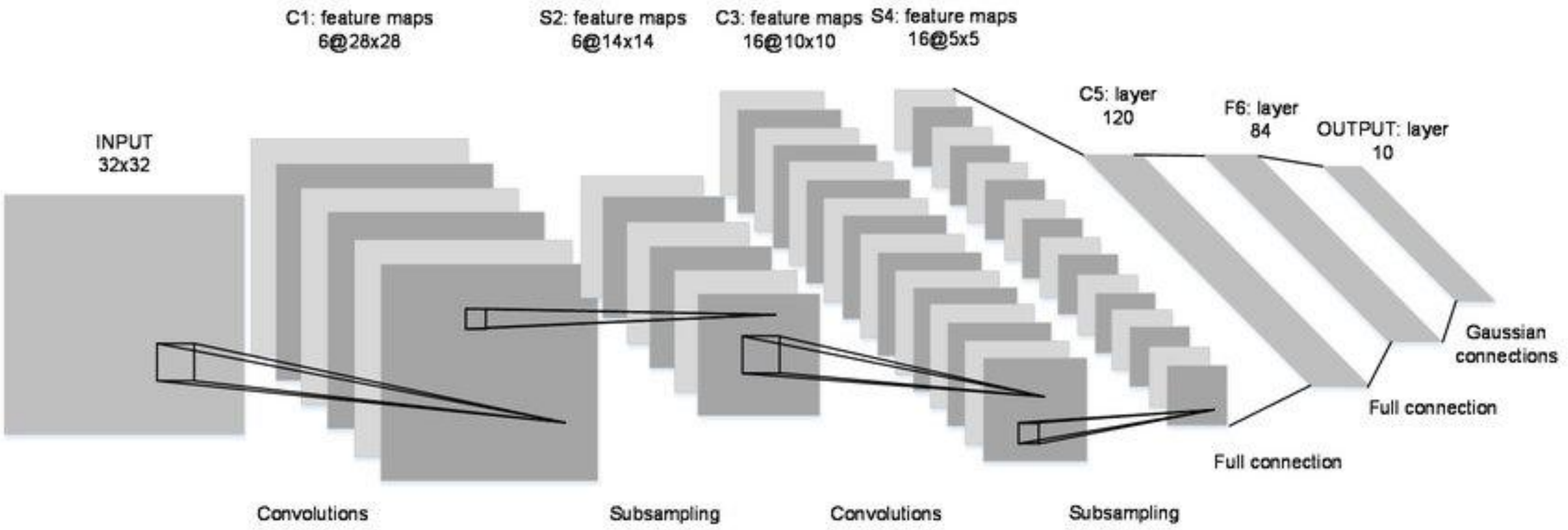
$$z(i,j) = \sum_{u=1}^{-1} \sum_{v=1}^{-1} I(i-u, j-v) w(u,v) + b$$

Derivative at location (i,j): $$\frac{\partial L}{\partial w}(i,j) = \frac{\partial L}{\partial z(i,j)} \frac{\partial z(i,j)}{\partial w} = \frac{\partial L}{\partial z(i,j)} I(i-1{:}i+1, j-1{:}j+1)$$

$$\frac{\partial L}{\partial b}(i,j) = \frac{\partial L}{\partial z(i,j)}$$

Architectures and basic design concepts

- LeNet-5

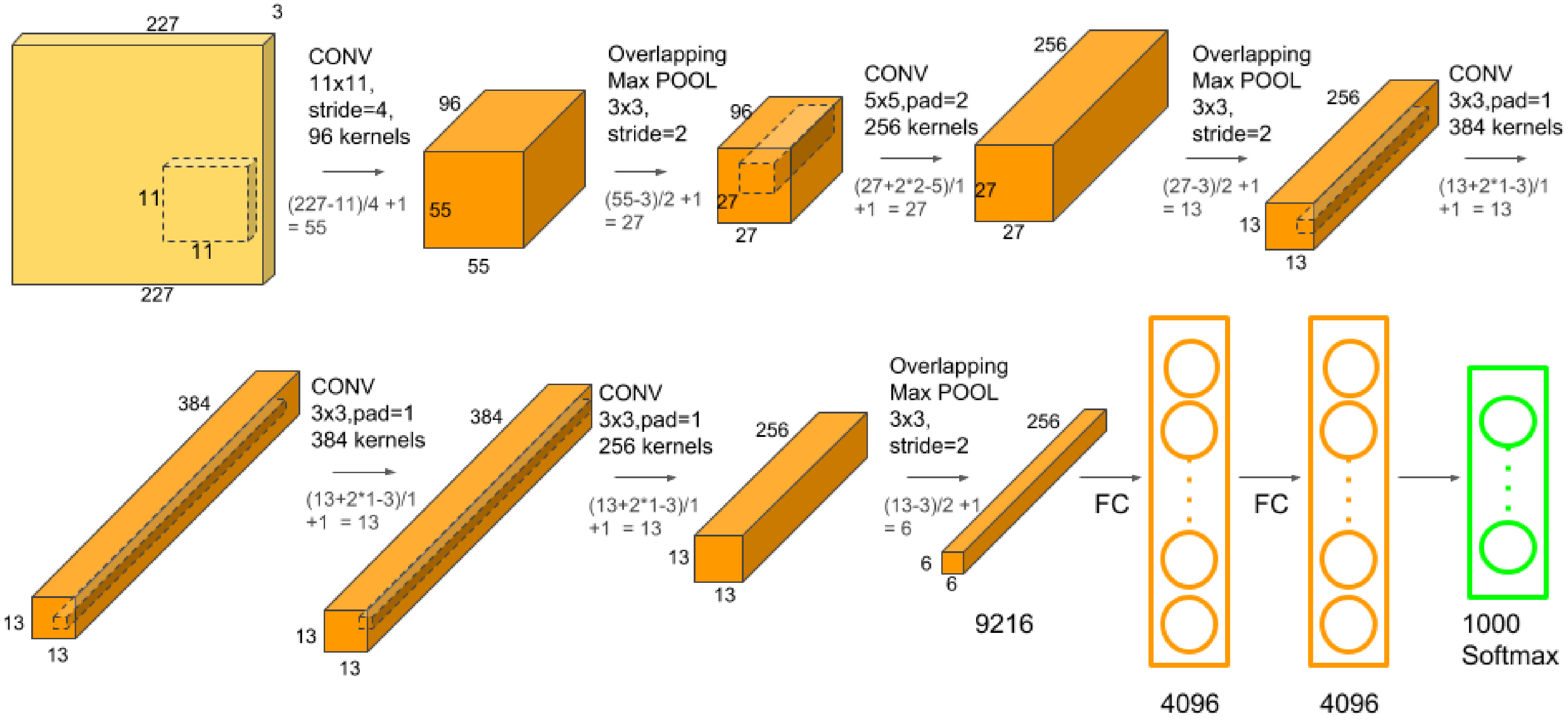


5x5 convolution (1 stride)
2x2 pooling (2 strides)

Y. LeCun, *et al.*, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, 1998

Architectures and basic design concepts

- AlexNet



$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

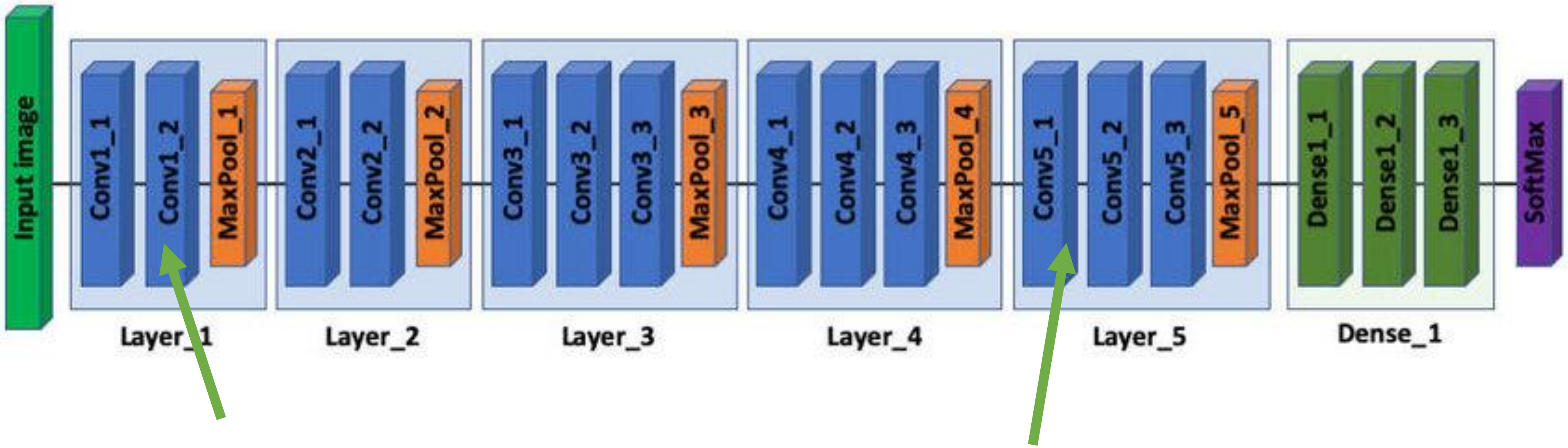
Softmax activation for disjoint classification

A. Krizhevsky et al.,. "ImageNet Classification with Deep Convolutional Neural Networks", *NIPS*, 2012

Architectures and basic design concepts

- VGG-16

All convolutions are 3x3, 1x1 stride, 1 padding
All max pooling layers are 2x2 with 2x2 strides



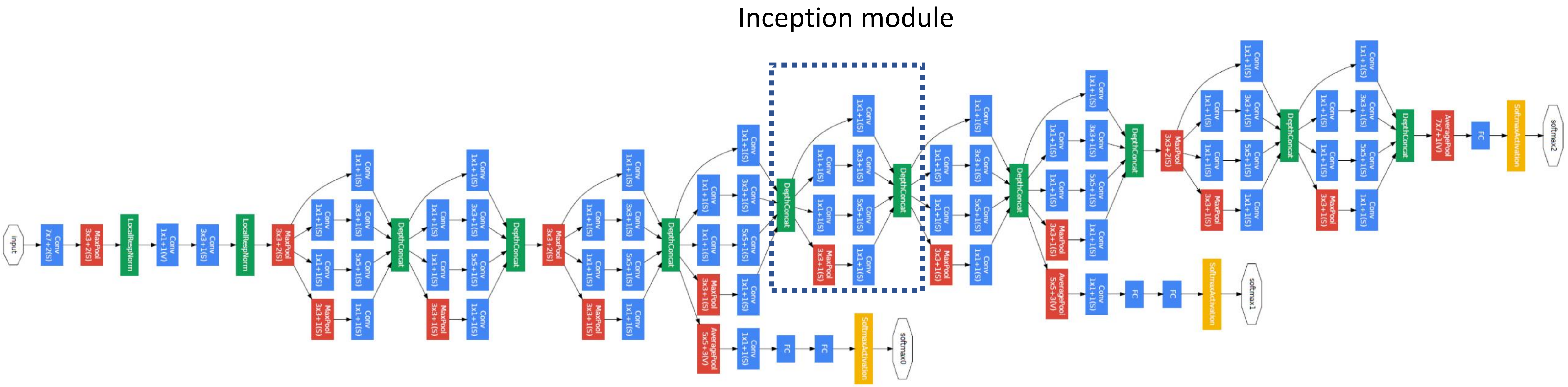
Why two 3x3 kernels? It covers the same space than a 5x5 kernel with less parameters (3x3x2=18 vs. 5x5=25).

Three 3x3 kernels (27 parameters) covers the same space than one 7x7 kernel (49 parameters).

K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *ICLR*, 2015

Architectures and basic design concepts

- GoogLeNet

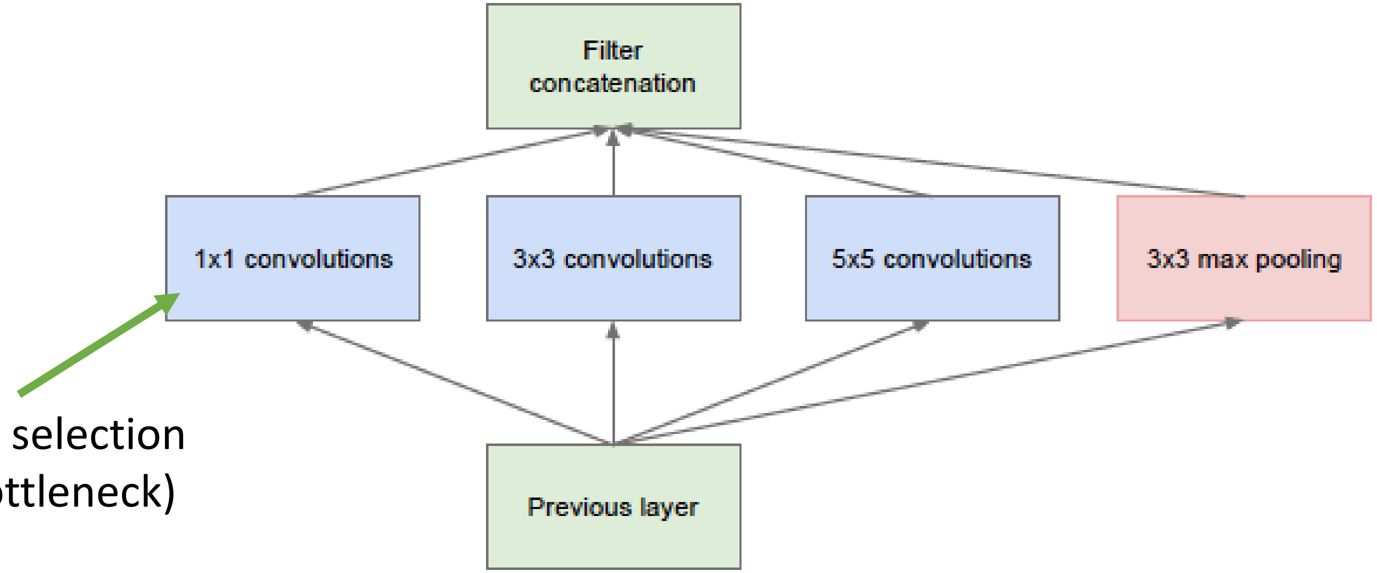


C. Szegedy *et al.*, “Going Deeper with Convolutions”, *CVPR*, 2015

Architectures and basic design concepts

- GoogLeNet

Theoretical inception module: multi-scale filter bank (remember multi-scale Gabor filter banks?)



1x1 strides

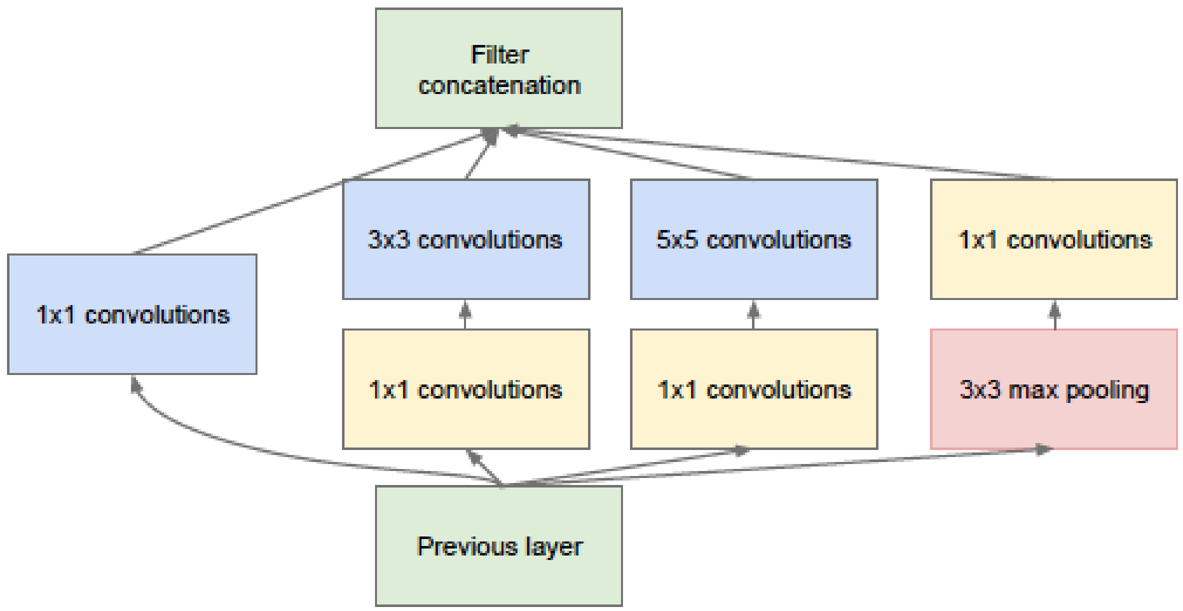
Feature selection
(aka bottleneck)

C. Szegedy *et al.*, "Going Deeper with Convolutions", *CVPR*, 2015

Architectures and basic design concepts

- GoogLeNet

Inception and dimensionality reduction in GoogLeNet

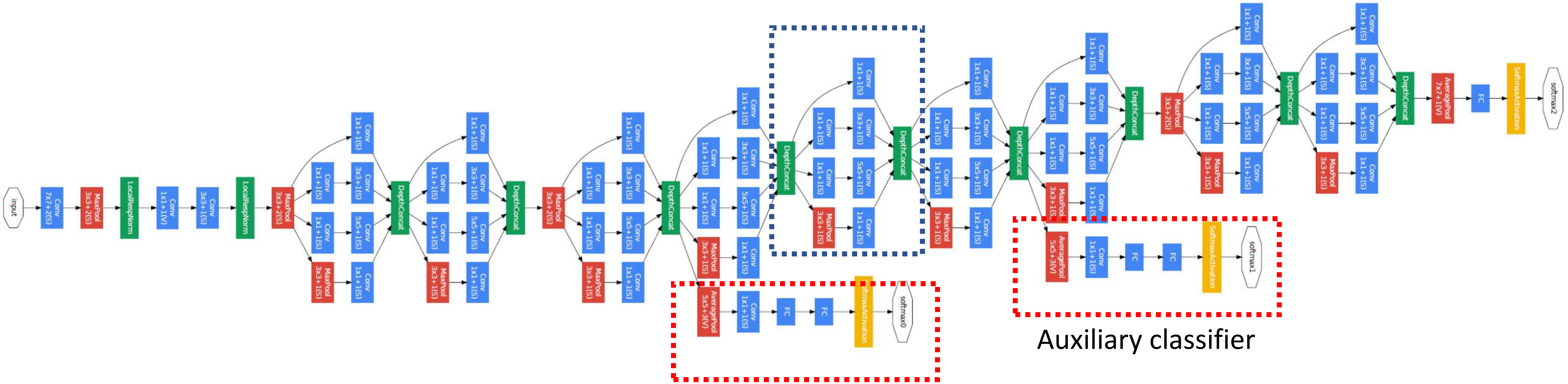


C. Szegedy *et al.*, "Going Deeper with Convolutions", *CVPR*, 2015

Architectures and basic design concepts

- GoogLeNet

Inception module



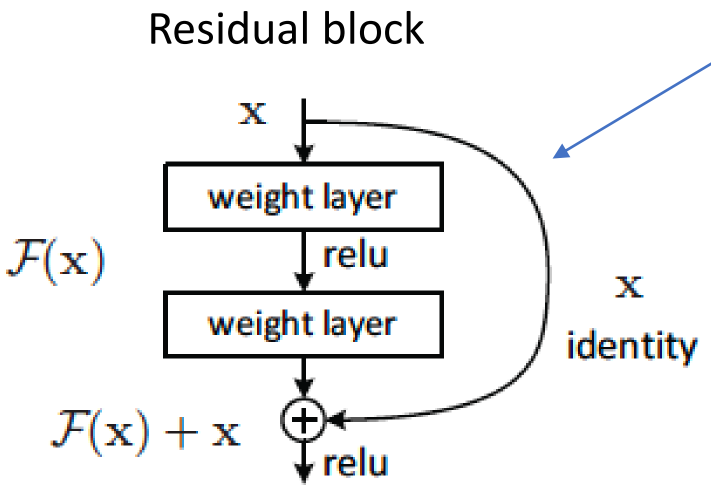
Auxiliary classifier

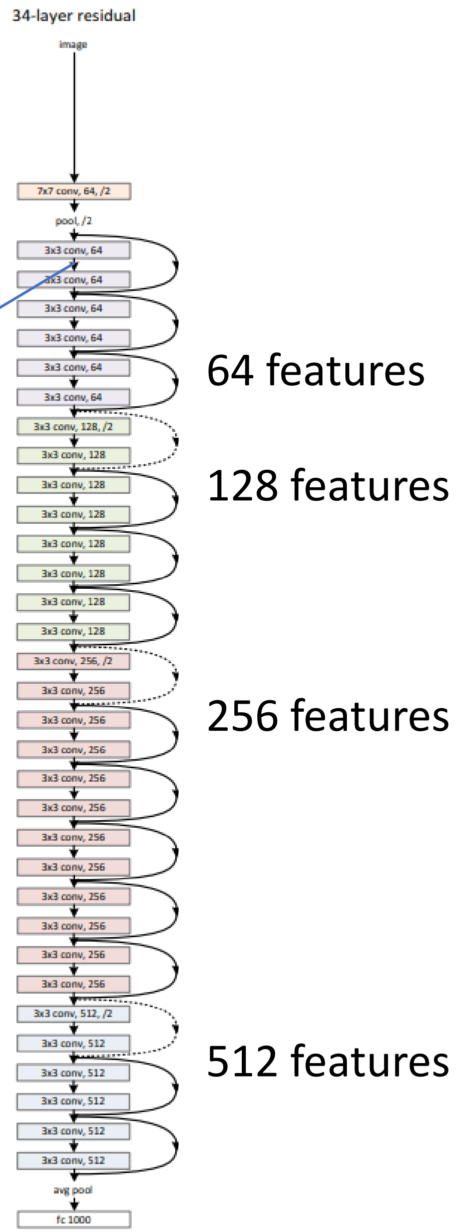
Auxiliary classifier

C. Szegedy *et al.*, “Going Deeper with Convolutions”, *CVPR*, 2015

Architectures and basic design concepts

- ResNet
  - Deeper networks highly suffer from vanishing gradient problem
  - Residual blocks allows backpropagation of gradients without vanishing further
  - Resnet also showed that it may be optimal to double the number of features as the dimensions halve in deeper layers

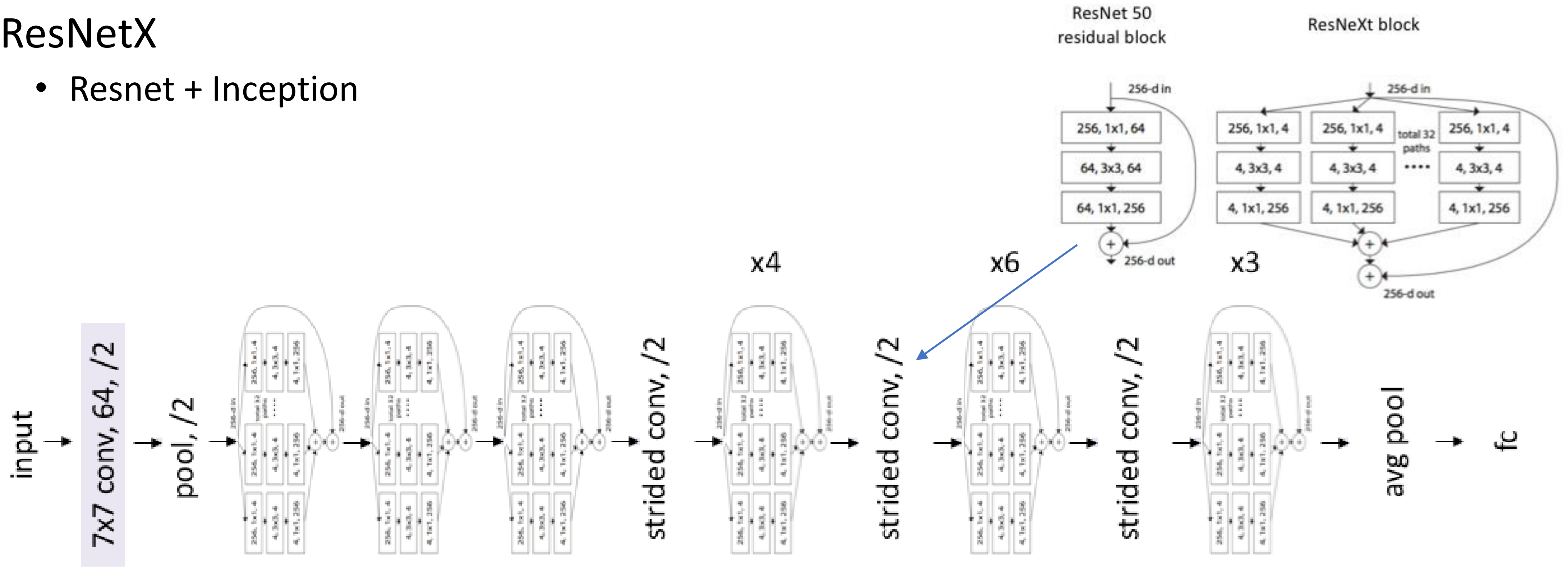




K. He, *et al.*, "Deep Residual Learning for Image Recognition", *CVPR*, 2016

Architectures and basic design concepts

- ResNetX
  - Resnet + Inception



S. Xie, *et al.*, “Aggregated Residual Transformations for Deep Neural Networks”, *CVPR*, 2017

Architectures and basic design concepts

- DenseNet
  - Pass all residuals to all layers in every block
  - Or how to take Resnet to the extreme…

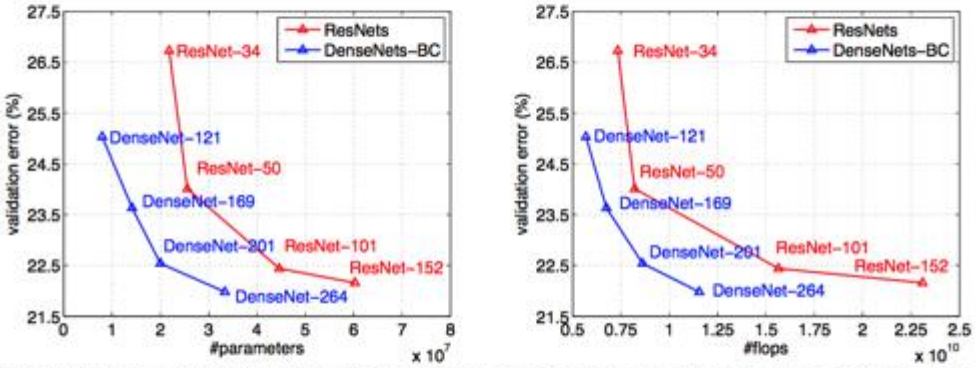
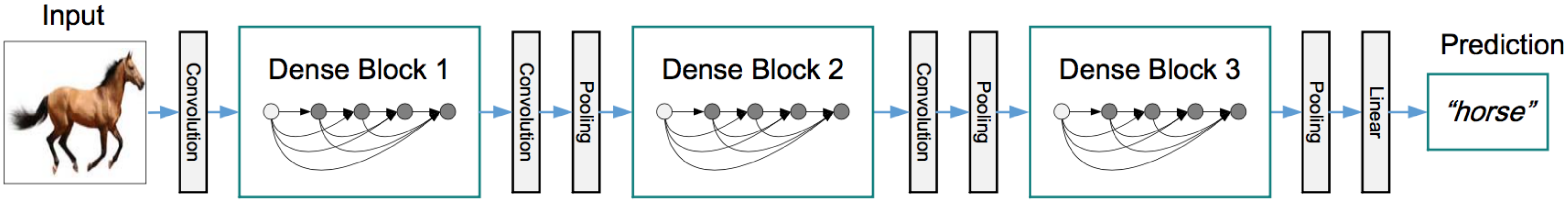


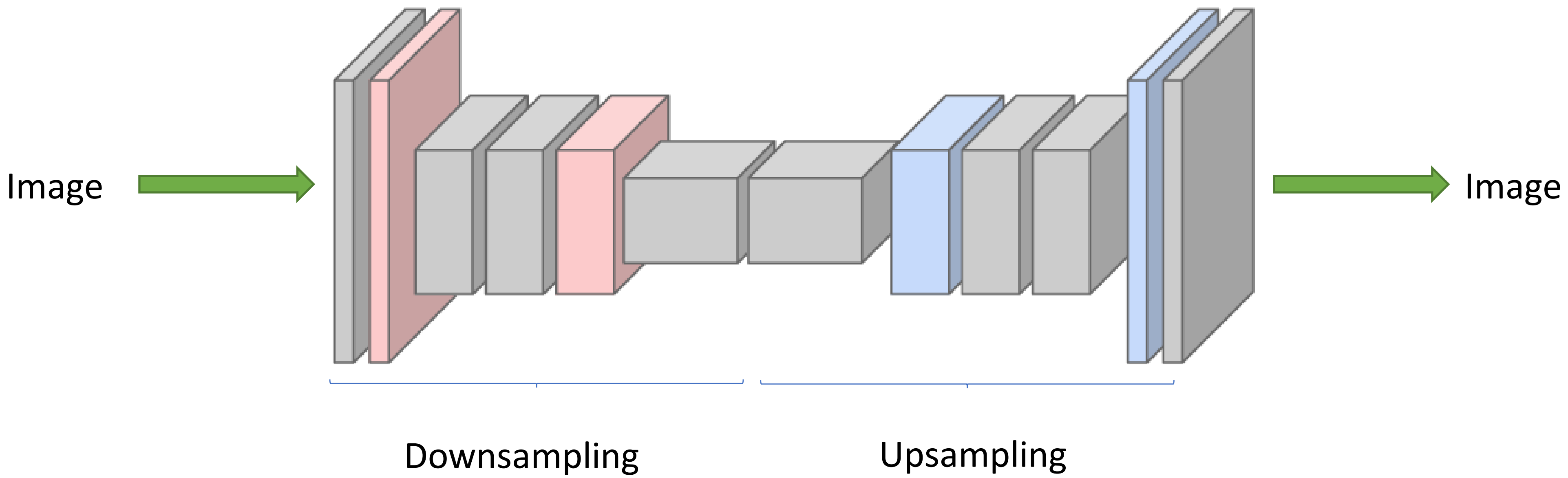
Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).



G. Huang, *et al.*, "Densely Connected Convolutional Networks", *CVPR*, 2017

Fully convolutional networks

- Fully convolutional networks
  - Normally designed to create an output image



Fully convolutional networks

❑ Fully convolutional networks
- Unpooling

Nearest Neighbor

1	2
3	4

Input: 2 x 2

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

"Bed of Nails"

1	2
3	4

Input: 2 x 2

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

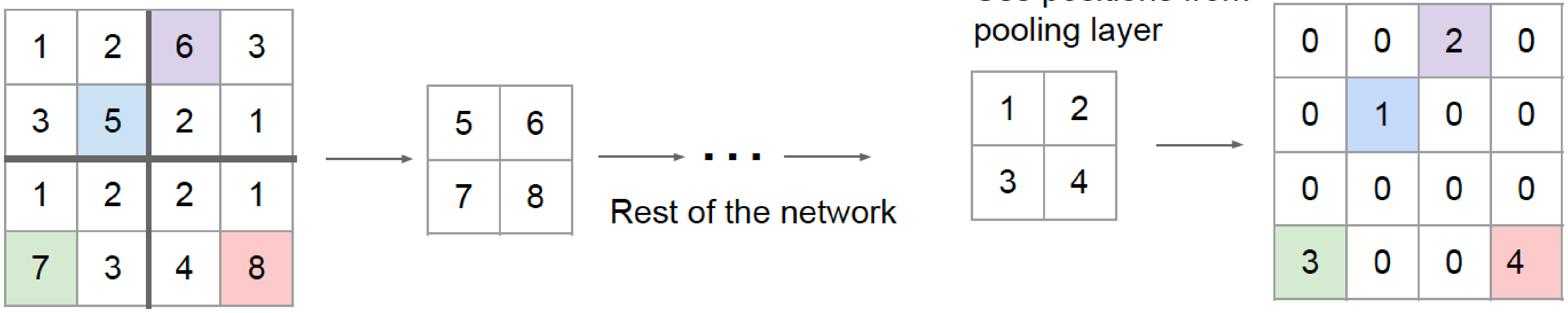
Output: 4 x 4

Fully convolutional networks

- Fully convolutional networks
  - Unpooling

Max Pooling
Remember which element was max!

Max Unpooling
Use positions from pooling layer



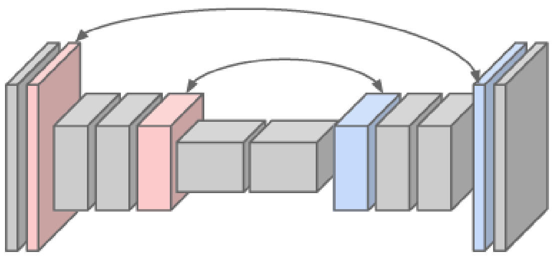
Input: 4 x 4

Output: 2 x 2

Input: 2 x 2

Output: 4 x 4

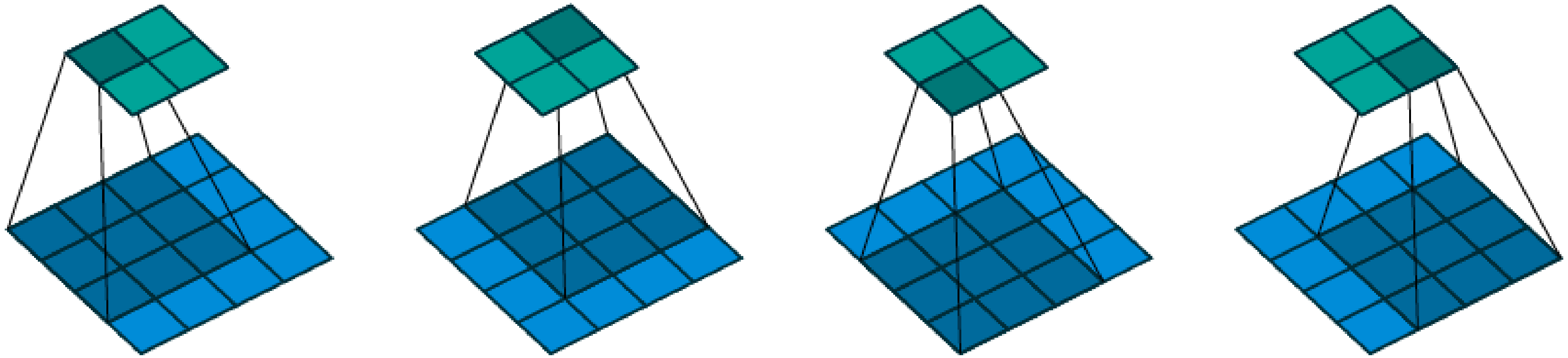
Corresponding pairs of downsampling and upsampling layers



Fully convolutional networks

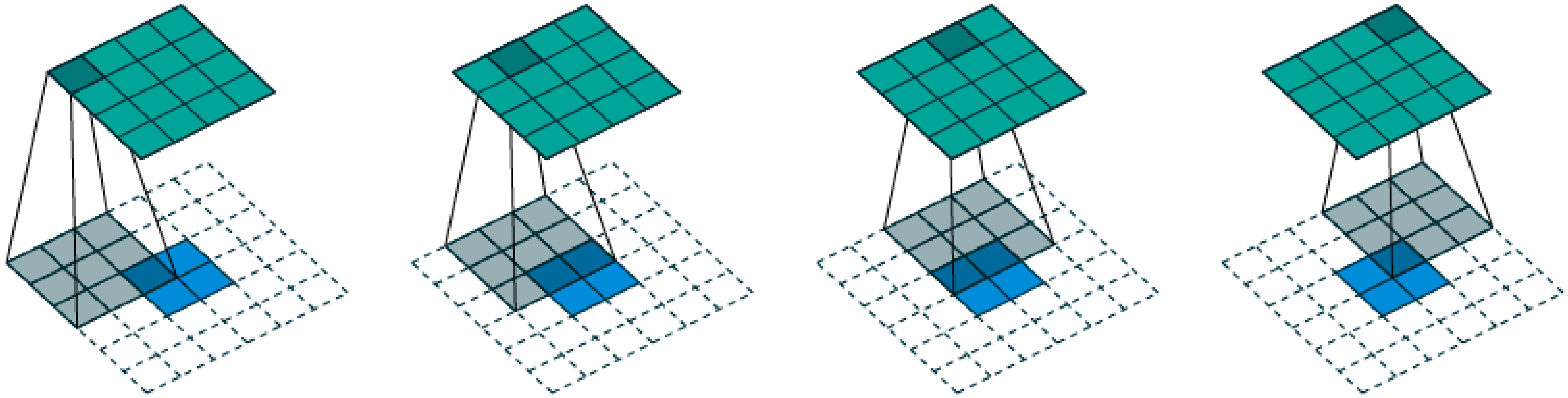
- Fully convolutional networks
  - Transposed convolution (or upconvolution)

Convolution



Input: 4×4
Kernel: 3×3
Stride: 1
Padding: 0
Output: 2×2

Transposed convolution

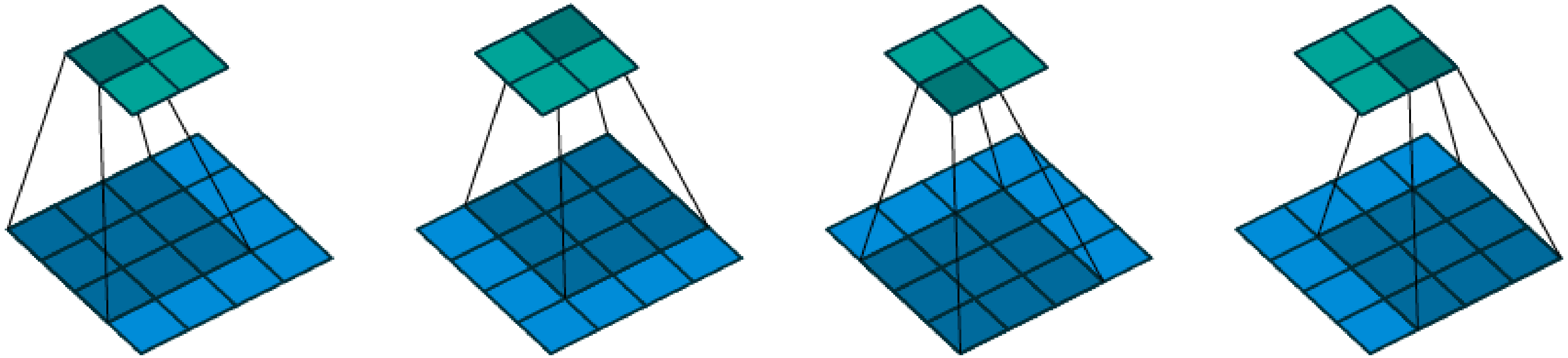


Input: 2×2
Kernel: 3×3
Stride: 1
Padding: 2
Output: 4×4

Fully convolutional networks

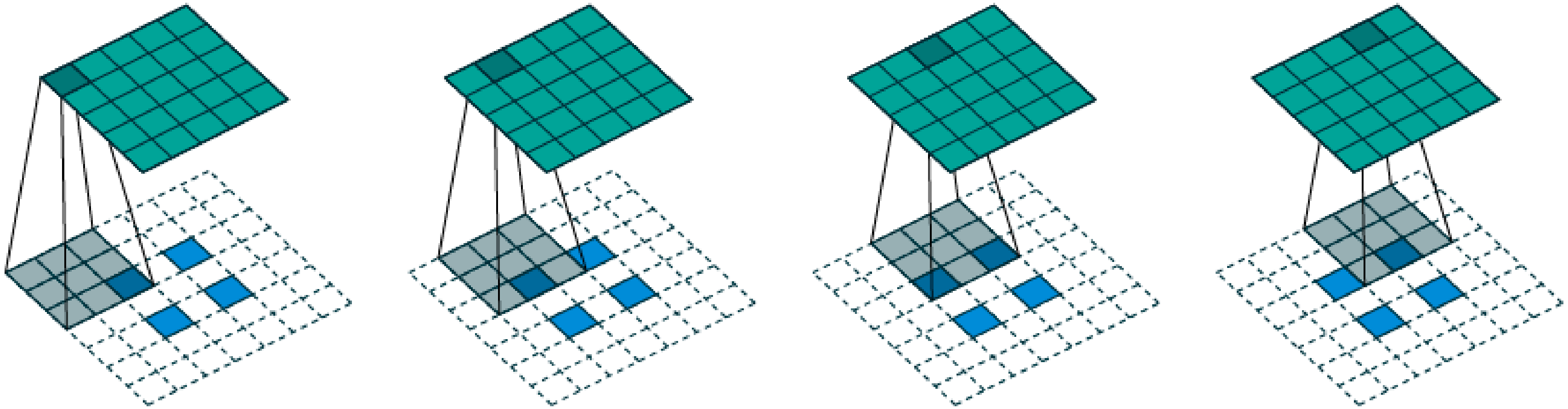
- Fully convolutional networks
  - Transposed convolution (or upconvolution)

Convolution



Input: 4×4
Kernel: 3×3
Stride: 1
Padding: 0
Output: 2×2

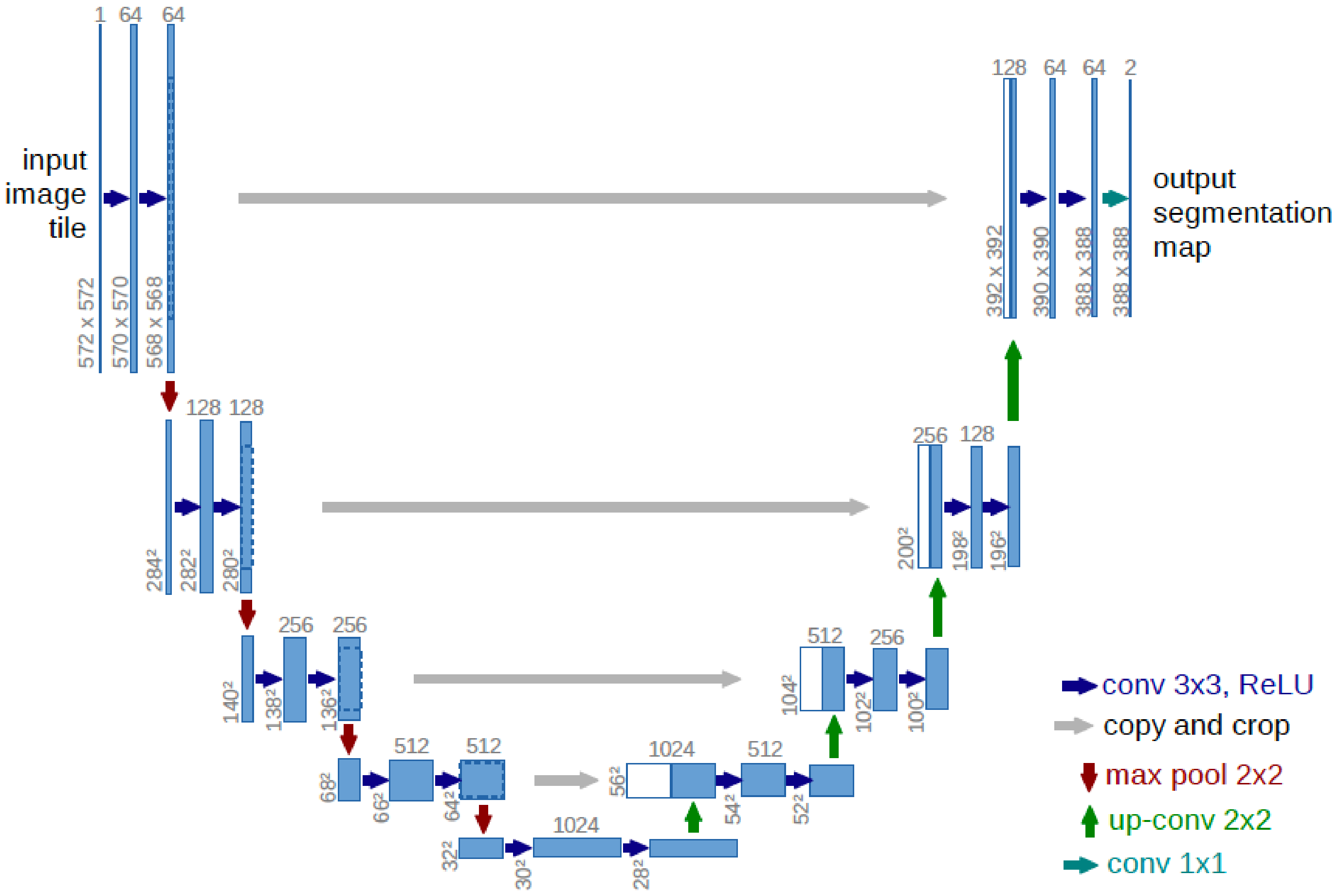
Transposed convolution



Input: 2×2
Kernel: 3×3
Stride: 1
Padding: 2
Output: 5×5

Fully convolutional networks

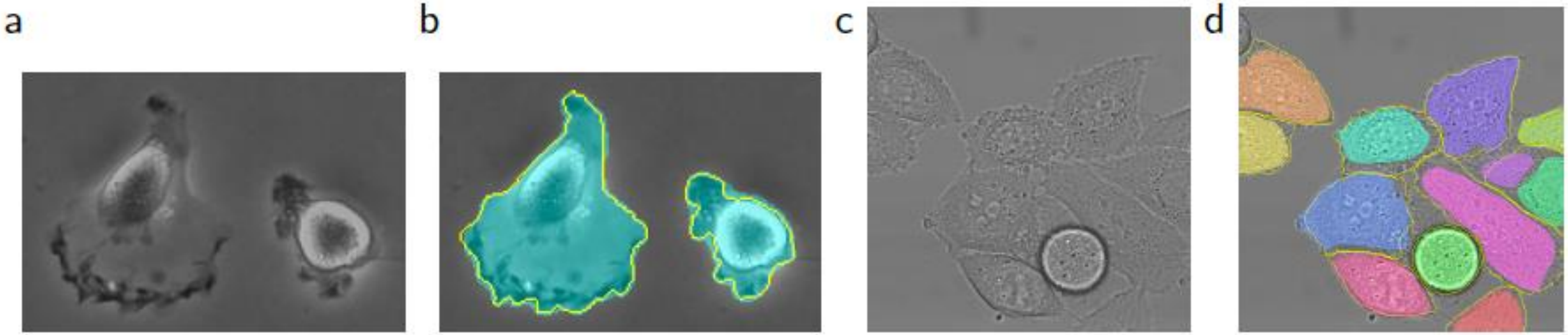
- U-net



O. Ronneberger *et al.*, 2015

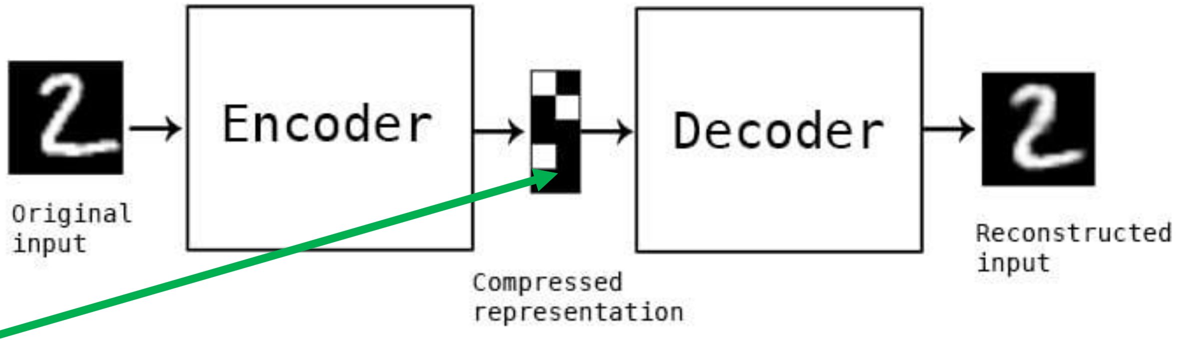
Fully convolutional networks

- U-net

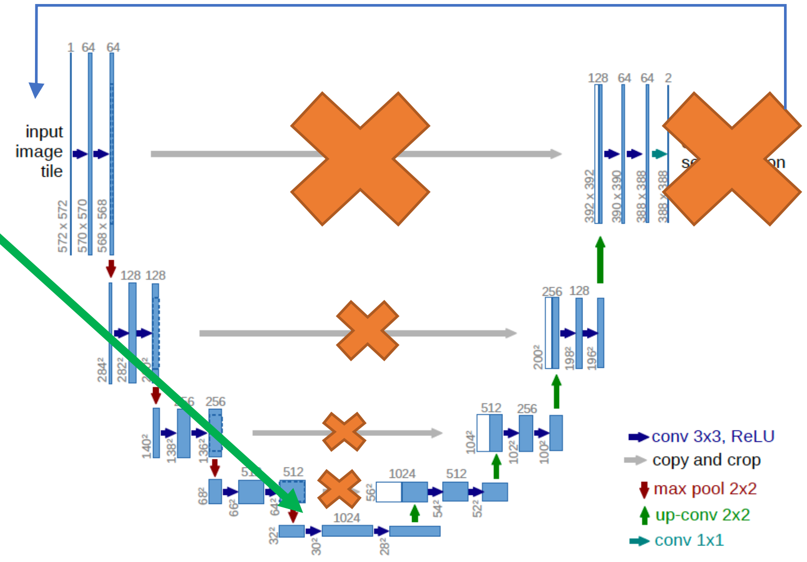


Fully convolutional networks

- Autoencoders



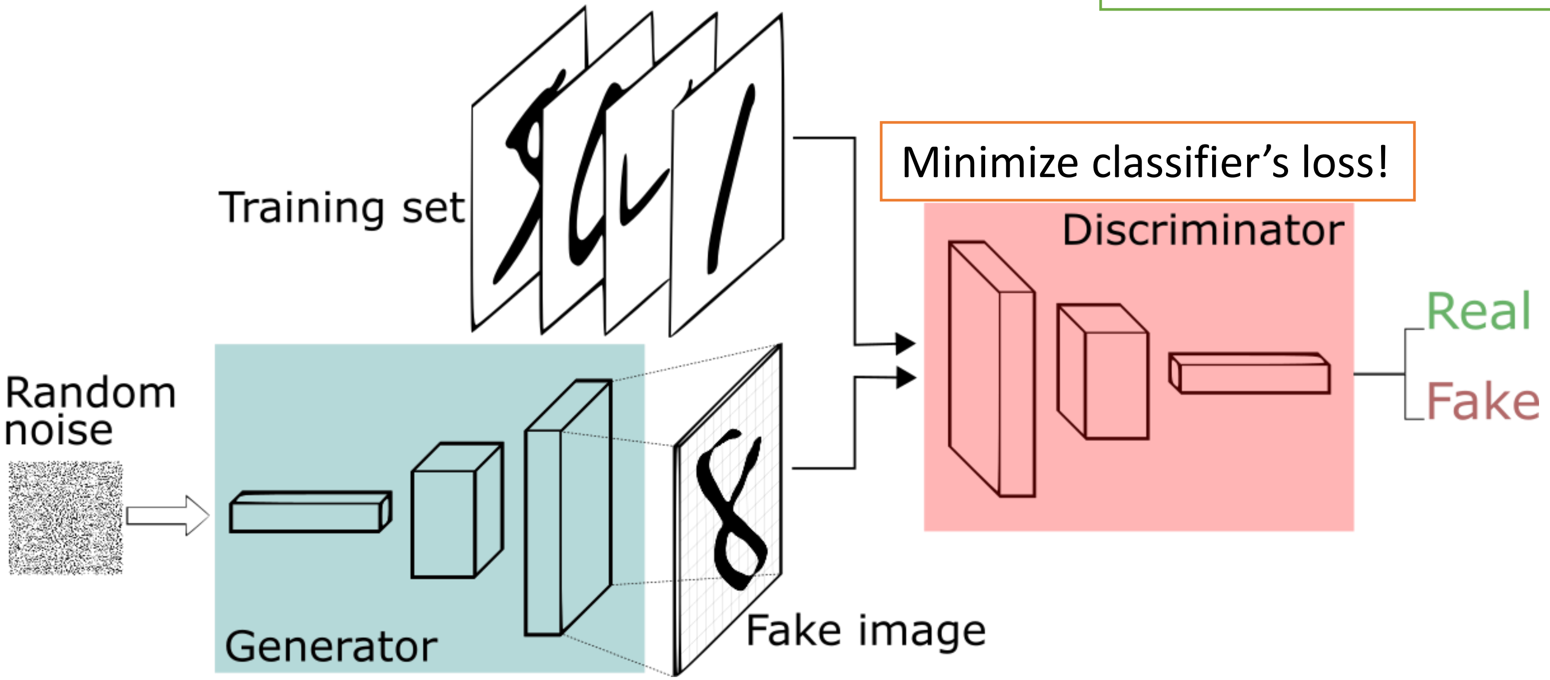
Latent space



Fully convolutional networks

- Generative adversarial networks (GANs)
  - Or learning how to fake images

Generator and discriminator must be trained independently iteratively



Minimize classifier's loss!

Maximize classifier's loss!