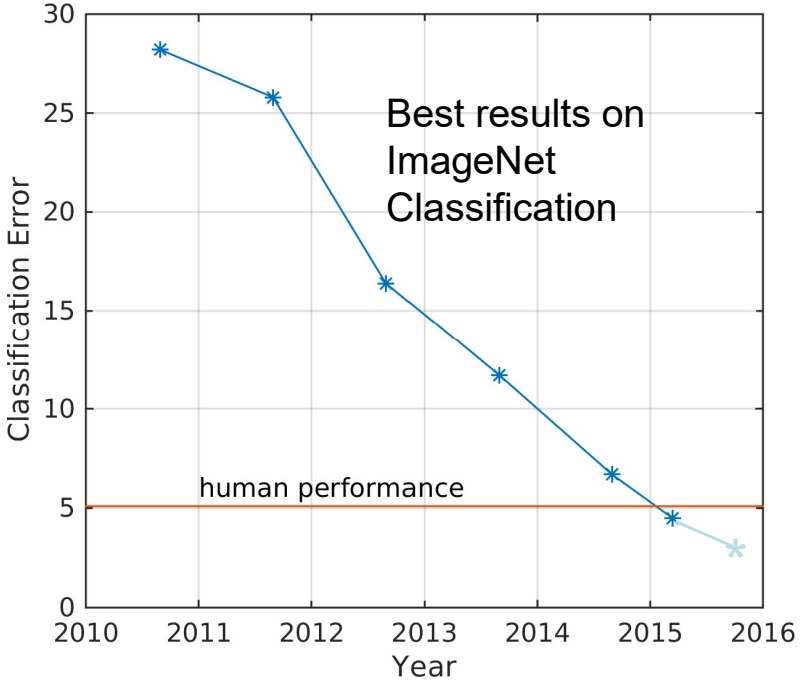
Image Processing

Class 11
Deep Learning for Biomedical Image Analysis

Recap: Deep Learning with Artificial Neural Networks

Deep convolutional neural networks surpassed human-level performance in Feb. 2015



GT: mountain tent
1: sleeping bag
2: mountain tent
3: parachute
4: ski
5: flagpole

GT: geyser
1: geyser
2: volcano
3: sandbar
4: breakwater
5: leatherback turtle

K. He, X. Zhang, S. Ren, J. Sun: “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.” (2015) arXiv:1502.01852 [cs.CV]

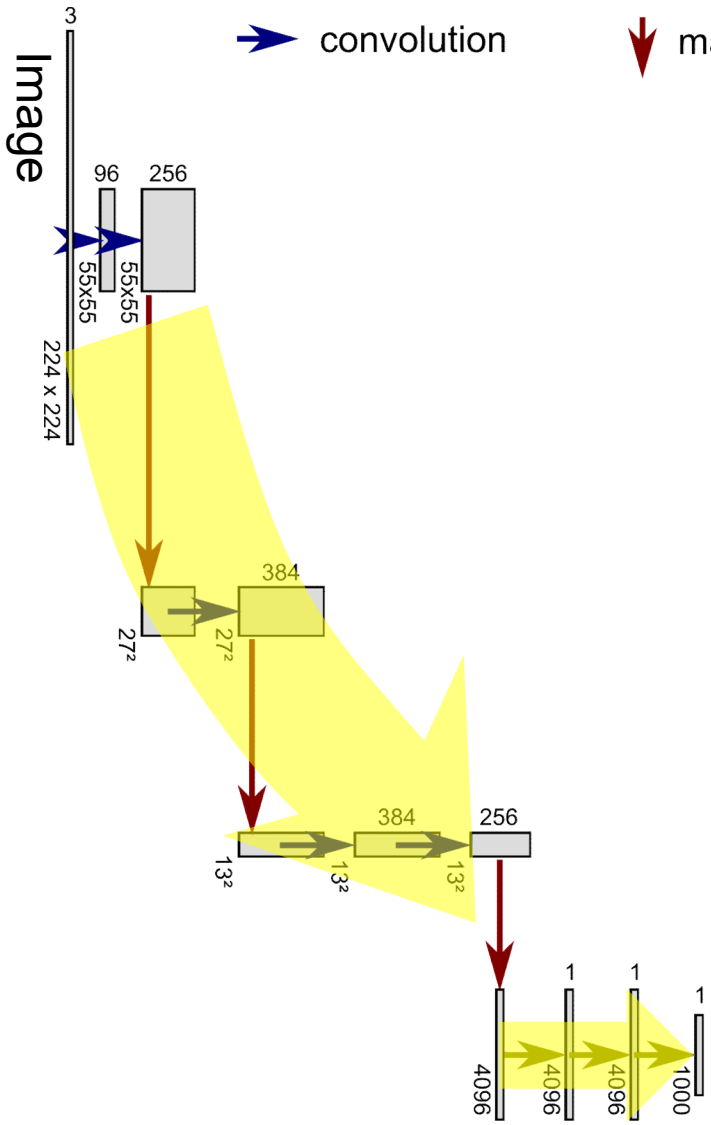
K. He, X. Zhang, S. Ren, J. Sun: “Deep Residual Learning for Image Recognition.” (2015) arXiv:1512.03385 [cs.CV]

Training a neural network for image classification requires many labeled images

- E.g. “ImageNet”: **1.2 Mio. images** from **1000 object classes**

[O. Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge.” (2014) arXiv:1409.0575 [cs.CV]]

Neural Networks for Image Segmentation: AlexNet



Feature extraction

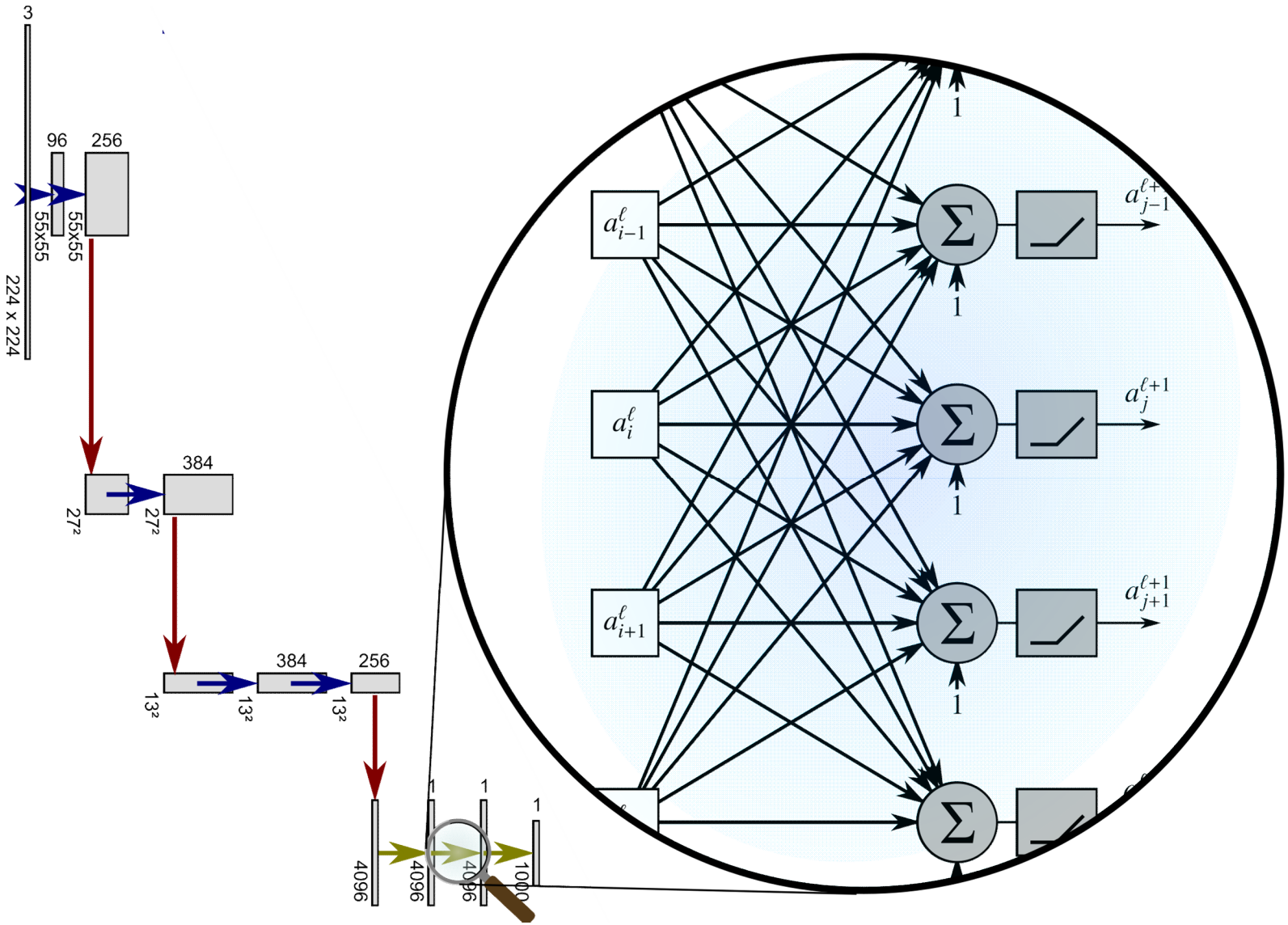
- Step-wise non-linear transformation of image content
  - Learned Filters (Convolution)
  - Subsampling (Max-Pooling)

Classification

- Final representation is mapped to class label
  - Fully connected network

Both **automatically learned** from image/label pairs only!

Fully Connected Networks for Classification

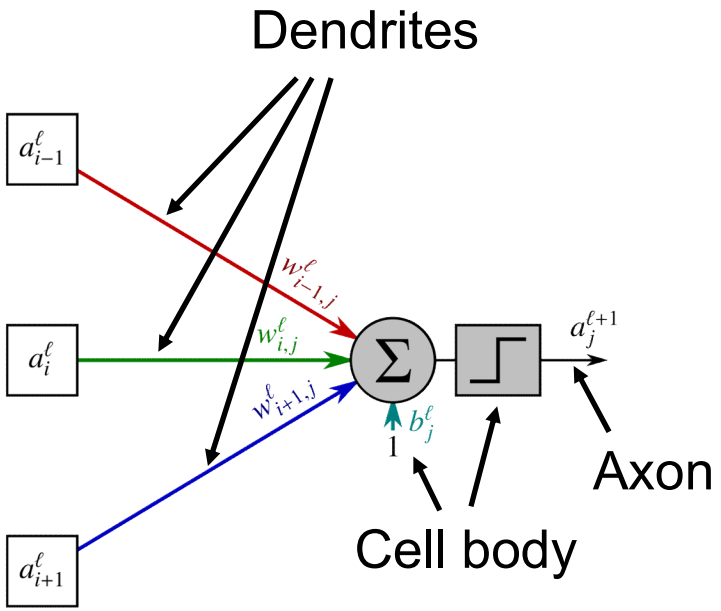


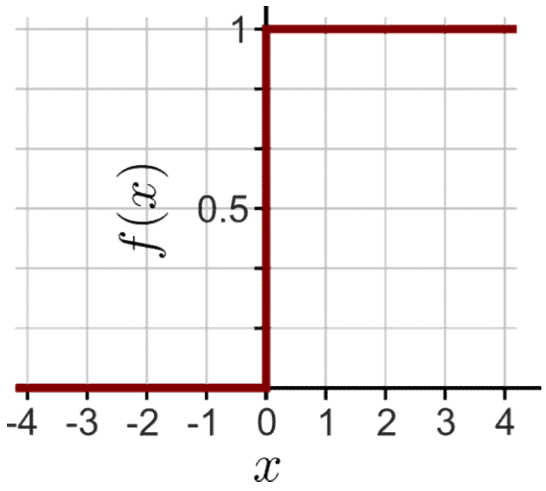
The Neuron (Perceptron)

$$a_j^{\ell+1} = f\left(w_{i-1,j}^{\ell} a_{i-1}^{\ell} + w_{i,j}^{\ell} a_i^{\ell} + w_{i+1,j}^{\ell} a_{i+1}^{\ell} + b_j^{\ell}\right)$$

4 learnable parameters

Stimuli (from previous neurons)



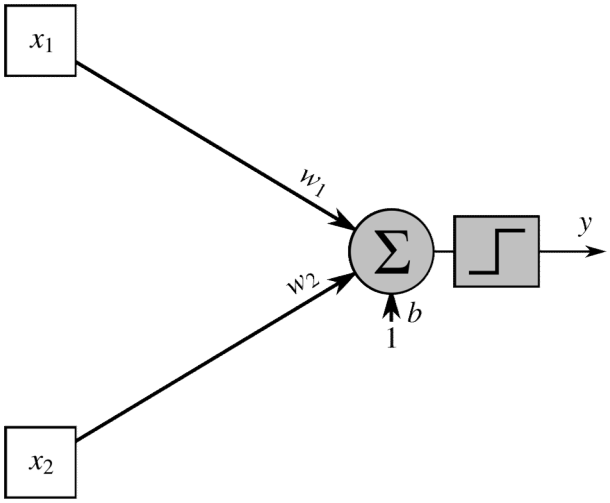


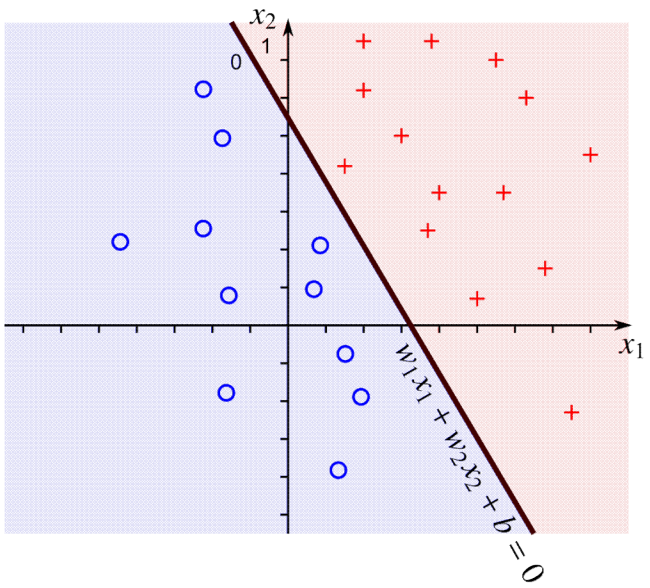
Activation function

- Neuron sums weighted stimuli from previous neurons
- Then transforms the result into binary decision (neuron fires)

Idea: Divide Input Space in Two Halves

- Weights of a **single neuron model a line** in 2D space
  - Inputs left to the line → class 0 (circle)
  - Inputs right of the line → class 1 (cross)

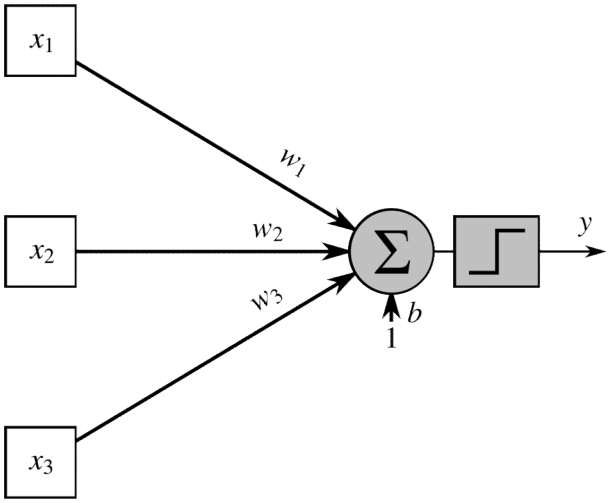


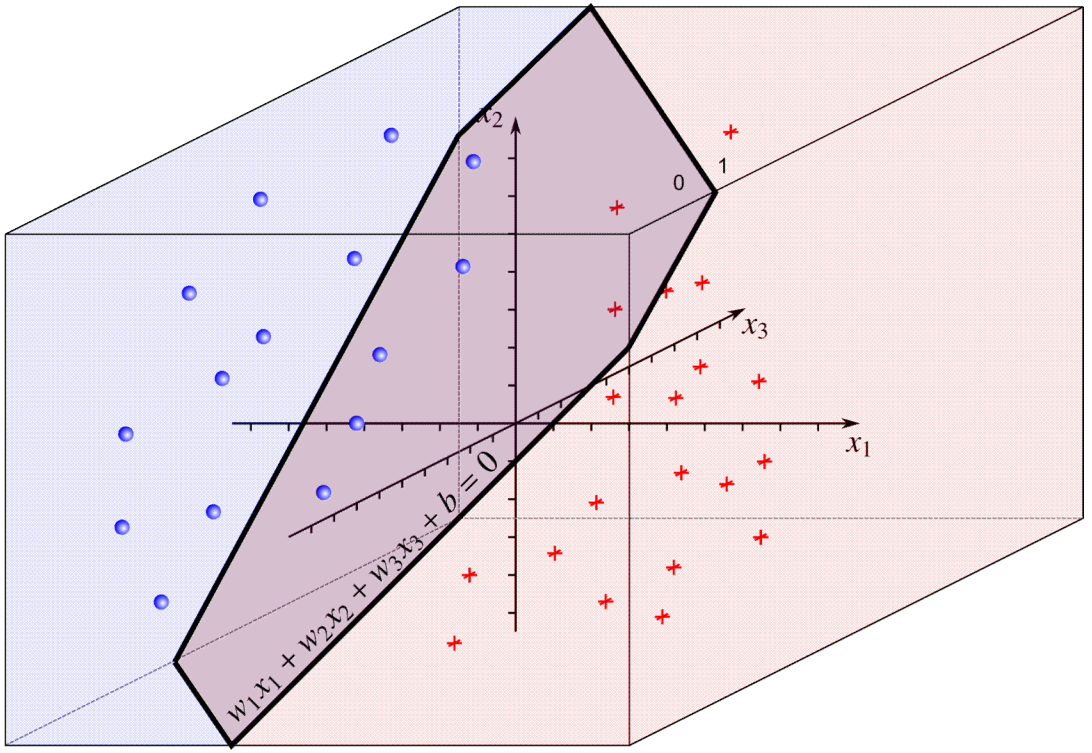


Input / Feature space

Idea: Divide Input Space in Two Halves

- Weights of a **single neuron model a plane** in 3D space
  - Inputs left to the plane → class 0 (sphere)
  - Inputs right of the plane → class 1 (cross)

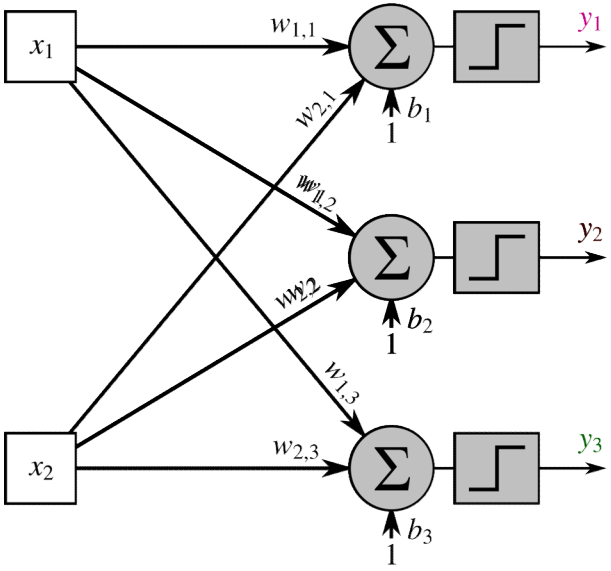


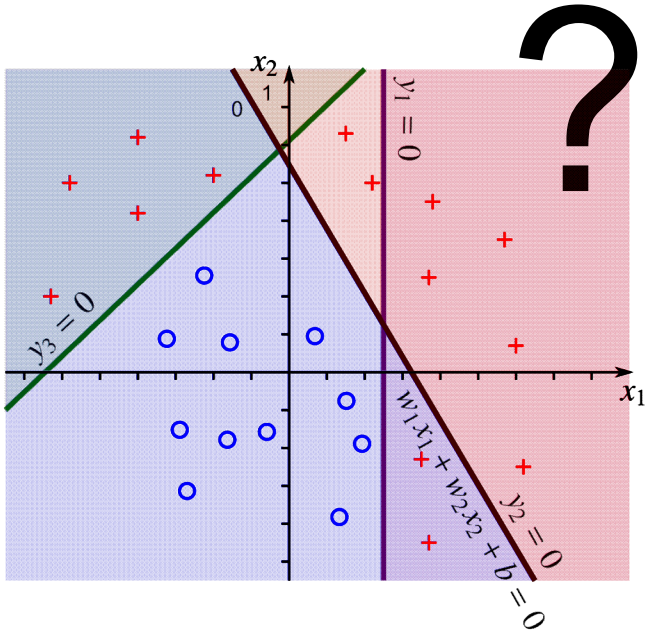


Input / Feature space

Idea: Divide Input Space in Two Halves

- What if inputs are **not linearly separable**?
- Add more lines = Add **more output neurons**!
- **Postpone classification** to following layers
- **3 layers** are sufficient to learn **classification of arbitrary inputs**!

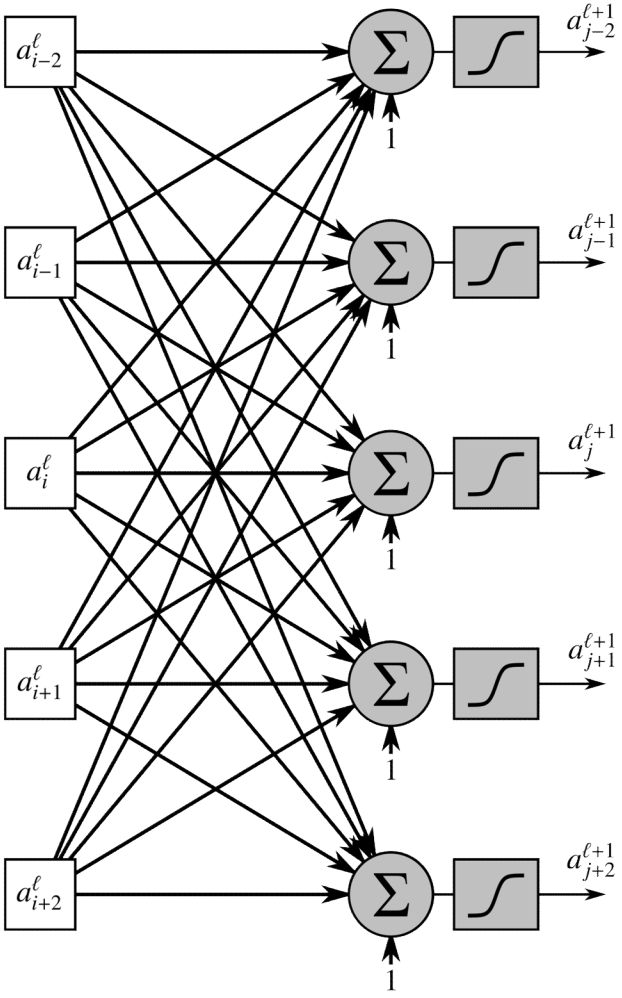




Input / Feature space

Fully Connected Networks

Stimuli (from previous neurons)



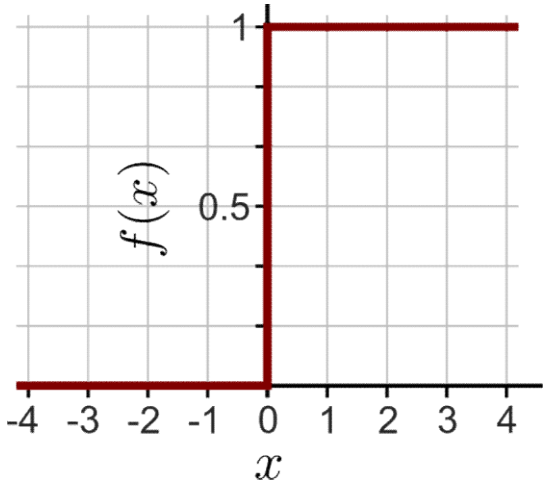
$$a_j^{\ell+1} = f\left(\sum_i w_{i,j}^{\ell} a_i^{\ell} + b_j^{\ell}\right)$$

5x5+5 = 30 learnable parameters!

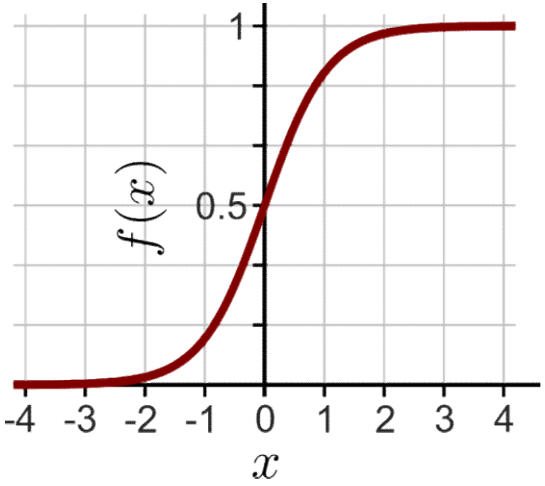
Many parameters
→ High capacity

- Can learn arbitrary dependencies between inputs
- Tends to **learn by heart** (over-fits training data)
- **Bad generalization** to unseen data

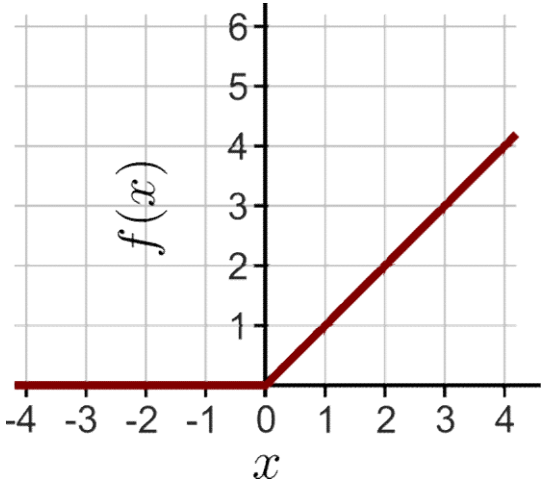
Differentiable Activation Functions



Step Activation



Sigmoid Activation

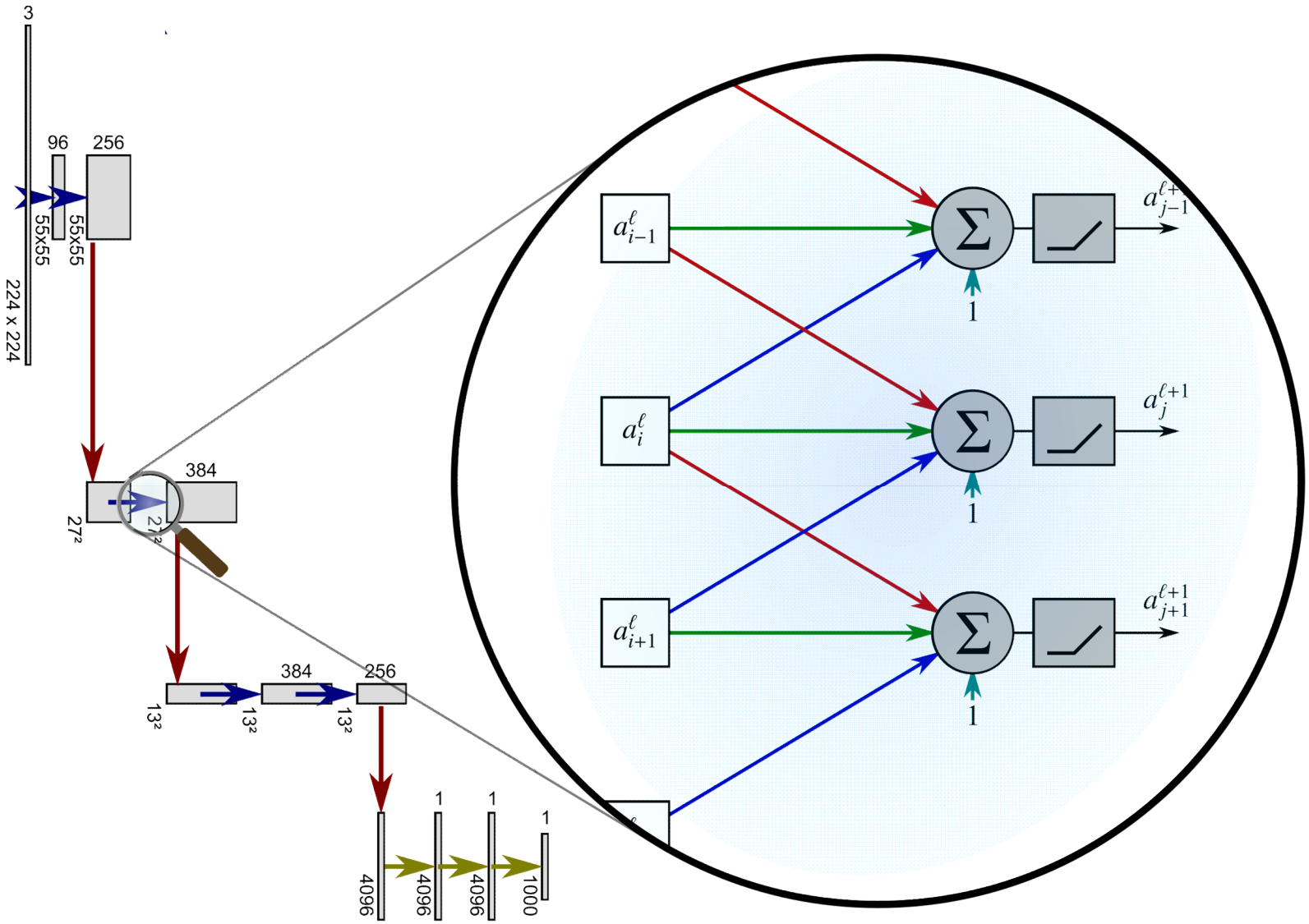


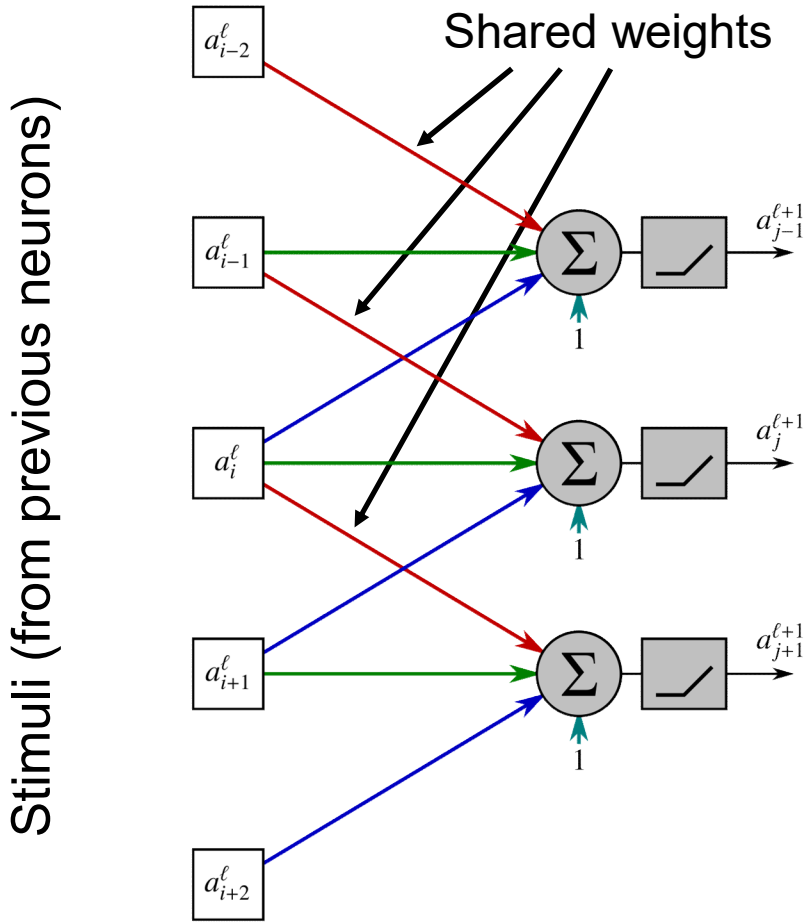
Rectified Linear Unit

Many optimizers rely on derivatives

- **Derivative of step is zero**, except for **x=0** where it is **undefined**
- Idea: Use **smooth step approximation**: Sigmoid
  - Downside: Quite expensive to evaluate and differentiate
- New Idea: Use a **mainly linear function** (ReLU)
  - Easy derivatives (derivative at x=0 must be defined to be 0 or 1)

Convolutional Networks for Feature Extraction





$$a_i^{\ell+1} = f\left(w_{-1}^{\ell} a_{i-1}^{\ell} + w_0^{\ell} a_i^{\ell} + w_1^{\ell} a_{i+1}^{\ell} + b^{\ell}\right)$$

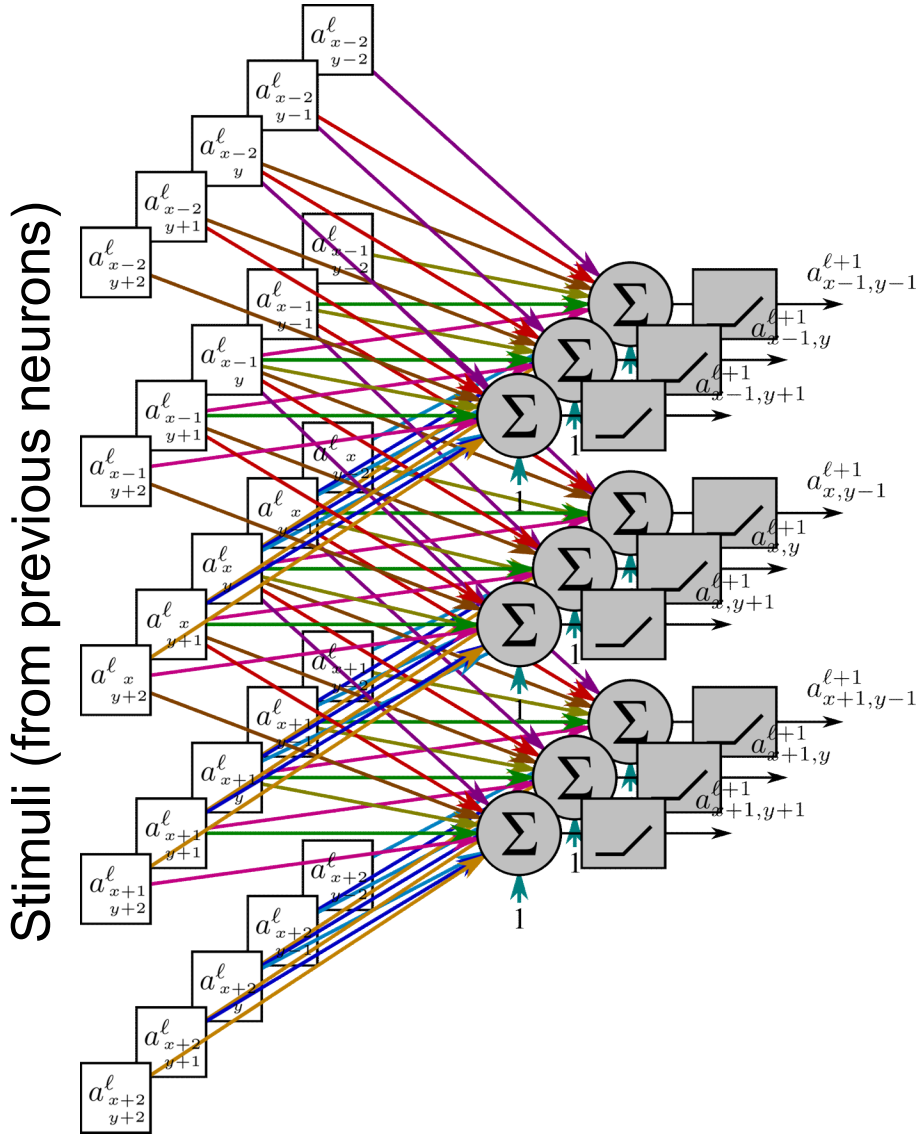
Only 4 learnable parameters!

Exploits image structure

Lower capacity

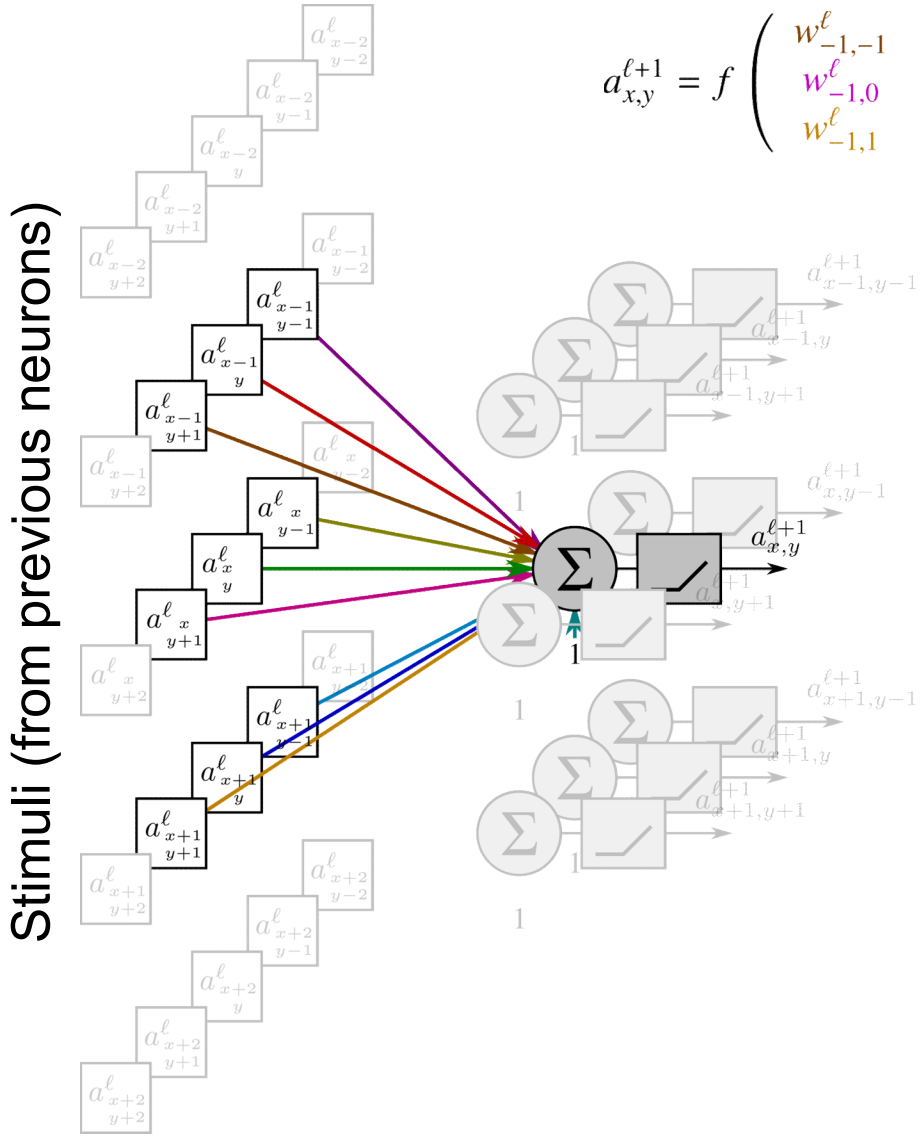
- Must learn **underlying concept**
- **Better generalization**

2D Convolutional Neuronal Networks



2D Convolutional Neuronal Networks

$$a^{\ell+1}_{x,y} = f\left(\begin{array}{ccccccccc} w^{\ell}_{-1,-1} & a^{\ell}_{x-1,y-1} & + & w^{\ell}_{0,-1} & a^{\ell}_{x,y-1} & + & w^{\ell}_{1,-1} & a^{\ell}_{x+1,y-1} & + \\ w^{\ell}_{-1,0} & a^{\ell}_{x-1,y} & + & w^{\ell}_{0,0} & a^{\ell}_{x,y} & + & w^{\ell}_{1,0} & a^{\ell}_{x+1,y} & + \\ w^{\ell}_{-1,1} & a^{\ell}_{x-1,y+1} & + & w^{\ell}_{0,1} & a^{\ell}_{x,y+1} & + & w^{\ell}_{1,1} & a^{\ell}_{x+1,y+1} & + \end{array} b^{\ell} \right)$$



10 learnable parameters!

Learns a non-linear filter

- Coefficients = Weights

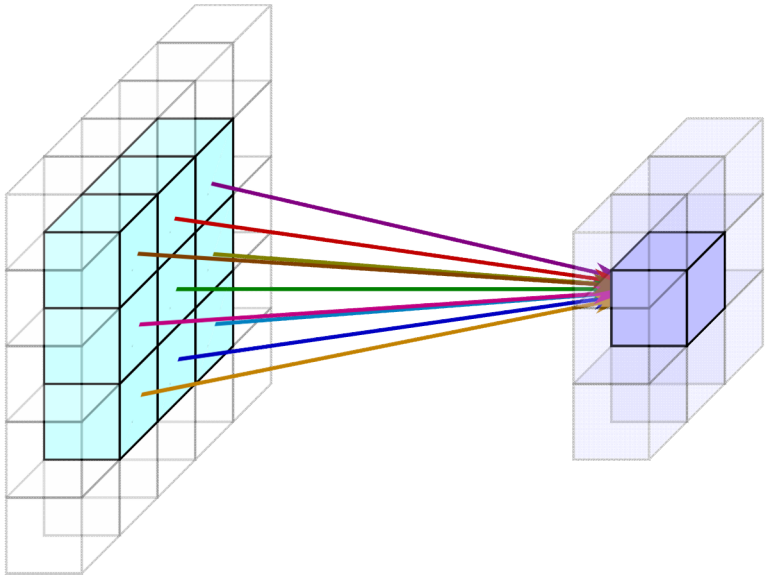
Filter size adjustable

- Here: 3x3
- Common: 3x3, …, 11x11

2D Convolutional Neuronal Networks

$$a_{x,y}^{\ell+1} = f\left(\begin{array}{ccccccccccc} w_{-1,-1}^{\ell} & a_{x-1,y-1}^{\ell} & + & w_{0,-1}^{\ell} & a_{x,y-1}^{\ell} & + & w_{1,-1}^{\ell} & a_{x+1,y-1}^{\ell} & + & & \\ w_{-1,0}^{\ell} & a_{x-1,y}^{\ell} & + & w_{0,0}^{\ell} & a_{x,y}^{\ell} & + & w_{1,0}^{\ell} & a_{x+1,y}^{\ell} & + & & \\ w_{-1,1}^{\ell} & a_{x-1,y+1}^{\ell} & + & w_{0,1}^{\ell} & a_{x,y+1}^{\ell} & + & w_{1,1}^{\ell} & a_{x+1,y+1}^{\ell} & + & b^{\ell} \end{array} \right)$$

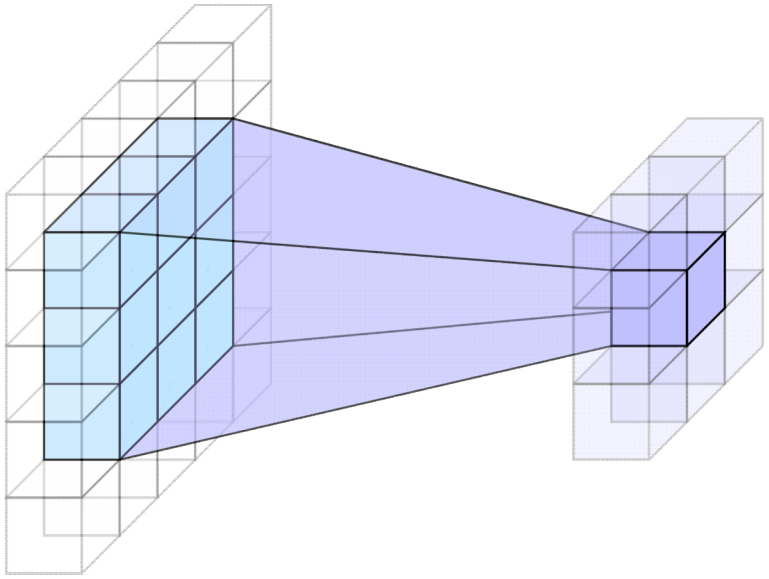
Stimuli (from previous neurons)



2D Convolutional Neuronal Networks

$$a_{x,y}^{\ell+1} = f\left(\sum_{x'=-\lfloor \frac{k_x}{2} \rfloor}^{\lfloor \frac{k_x}{2} \rfloor} \sum_{y'=-\lfloor \frac{k_y}{2} \rfloor}^{\lfloor \frac{k_y}{2} \rfloor} w_{x',y'}^{\ell} a_{x+x',y+y'}^{\ell} + b^{\ell}\right)$$

Stimuli (from previous neurons)



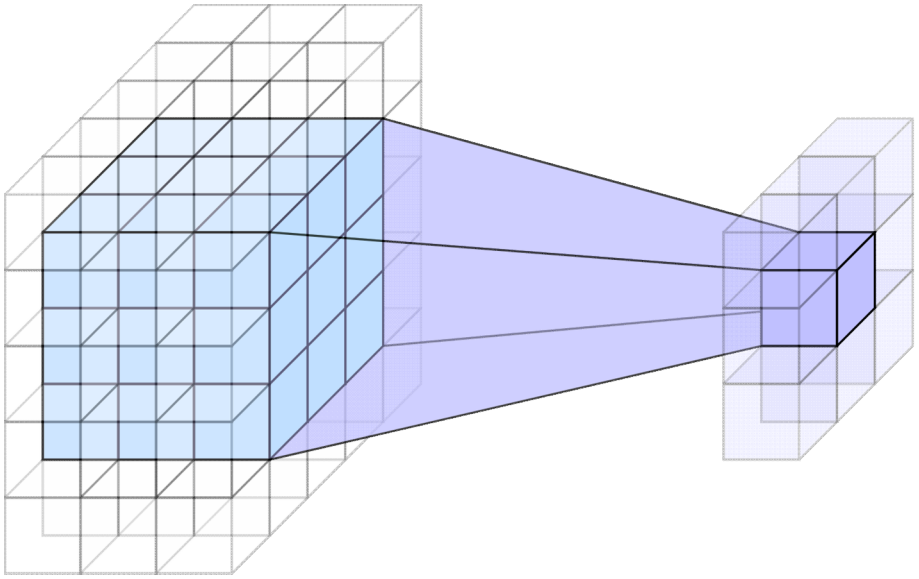
Single input channel:

- Nonlinear 2D-filter

2D Convolutional Neuronal Networks – Multiple Channels

$$a_{x,y}^{\ell+1} = f\left(\sum_{c_{\text{in}}=1}^{C_{\text{in}}} \sum_{x'=-\lfloor \frac{k_x}{2} \rfloor}^{\lfloor \frac{k_x}{2} \rfloor} \sum_{y'=-\lfloor \frac{k_y}{2} \rfloor}^{\lfloor \frac{k_y}{2} \rfloor} w_{x',y'}^{\ell,c_{\text{in}}} a_{x+x',y+y'}^{\ell,c_{\text{in}}} + b^{\ell}\right)$$

Stimuli (from previous neurons)



Single input channel:

- Nonlinear 2D-filter

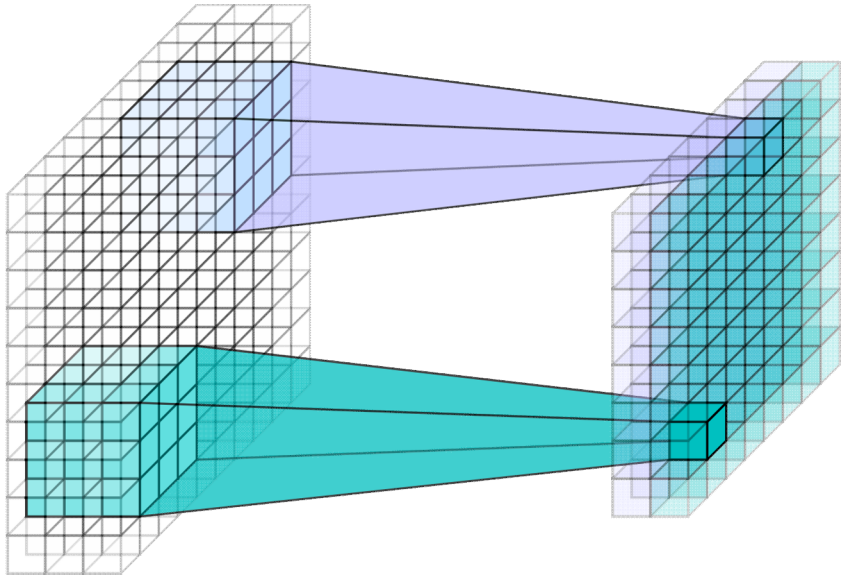
Multiple input channels:

- Simply extend filter along channel dimension

2D Convolutional Neuronal Networks – Multiple Filters

$$a_{x,y}^{\ell+1,c_{\text{out}}} = f\left(\sum_{c_{\text{in}}=1}^{C_{\text{in}}} \sum_{x'=-\lfloor \frac{k_x}{2} \rfloor}^{\lfloor \frac{k_x}{2} \rfloor} \sum_{y'=-\lfloor \frac{k_y}{2} \rfloor}^{\lfloor \frac{k_y}{2} \rfloor} w_{x',y'}^{\ell,c_{\text{in}},c_{\text{out}}} a_{x+x',y+y'}^{\ell,c_{\text{in}}} + b^{\ell,c_{\text{out}}}\right)$$

Stimuli (from previous neurons)



Single input channel:

- Nonlinear 2D-filter

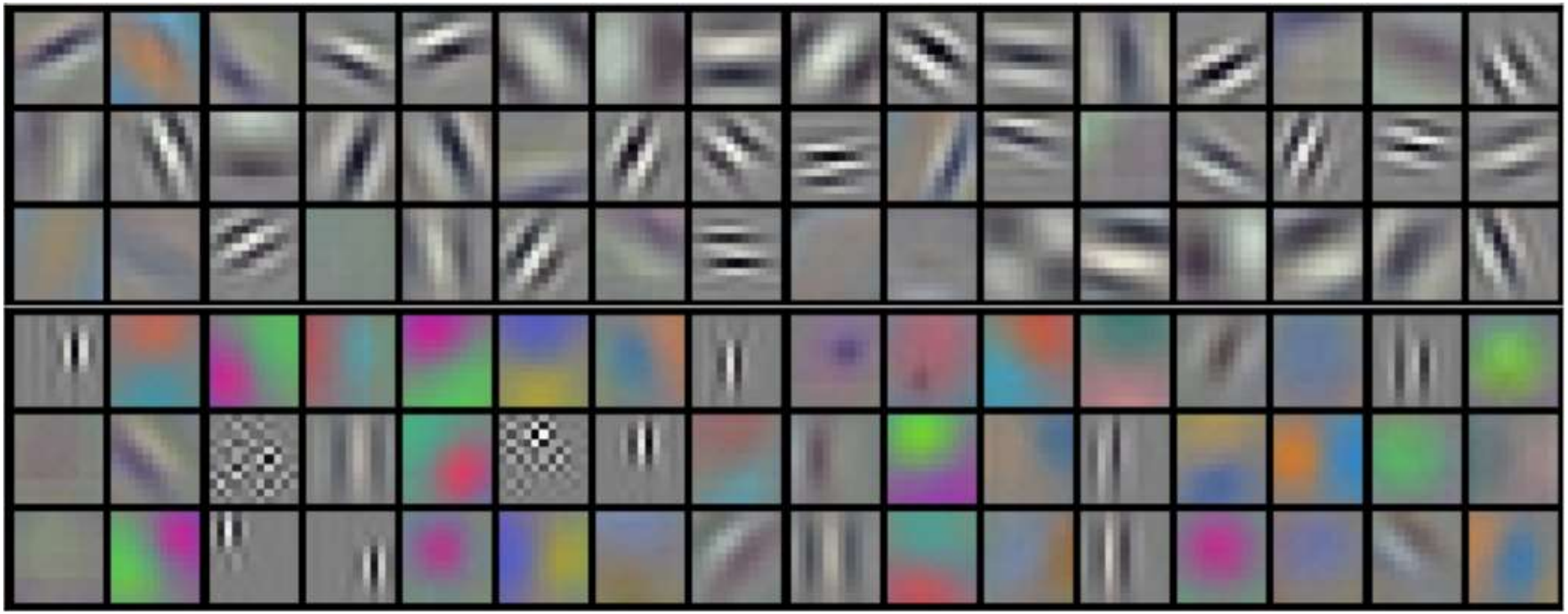
Multiple input channels:

- Simply extend filter along channel dimension

Multiple filters:

- Generate one output channel per filter

Learned Filters (First Layer)



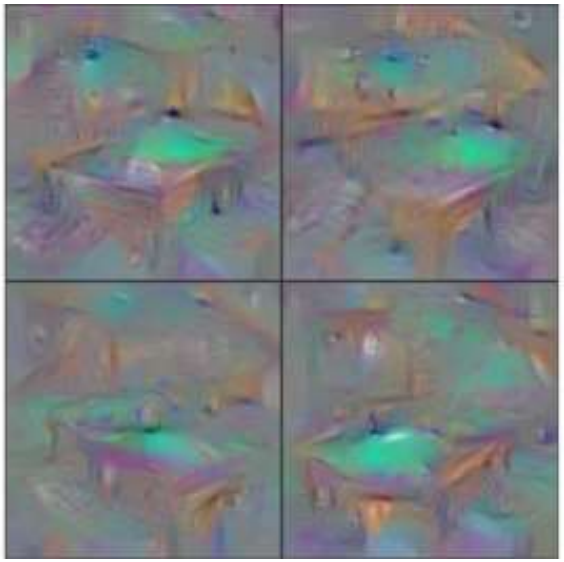
[Krizhevski et al., NIPS 2012]

Filters in the first layer encode local structure

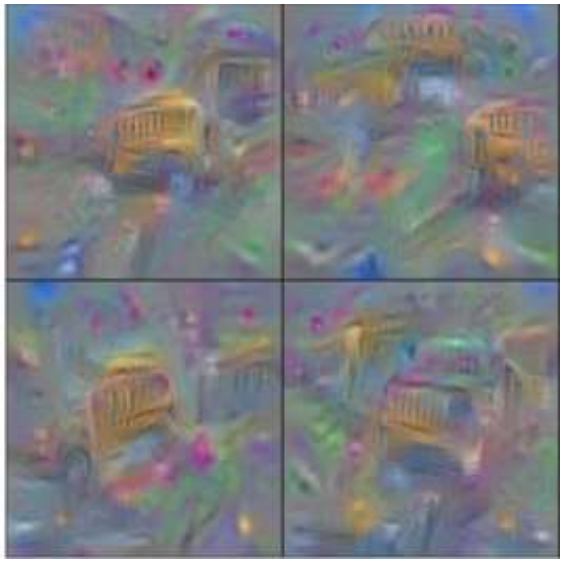
- Edges (variable frequency and orientation), texture, color blobs, …
- Similar to the first ganglions in the eye‘s retina
- Deeper layers harder to interpret

Learned Filters (Deeper Layers)

Flamingo



Billard table



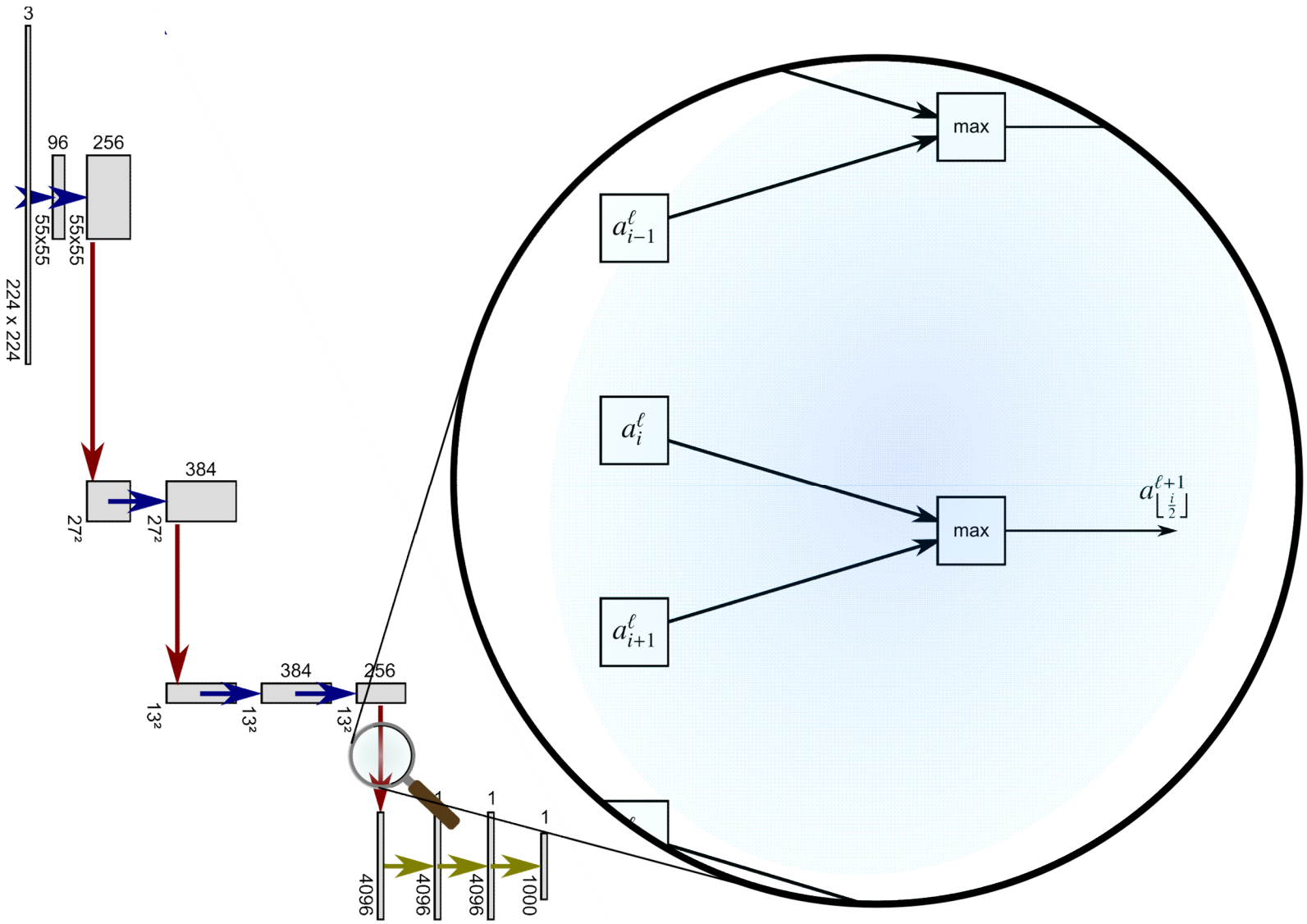
School bus

[Yosinski et al., ICML 2015]

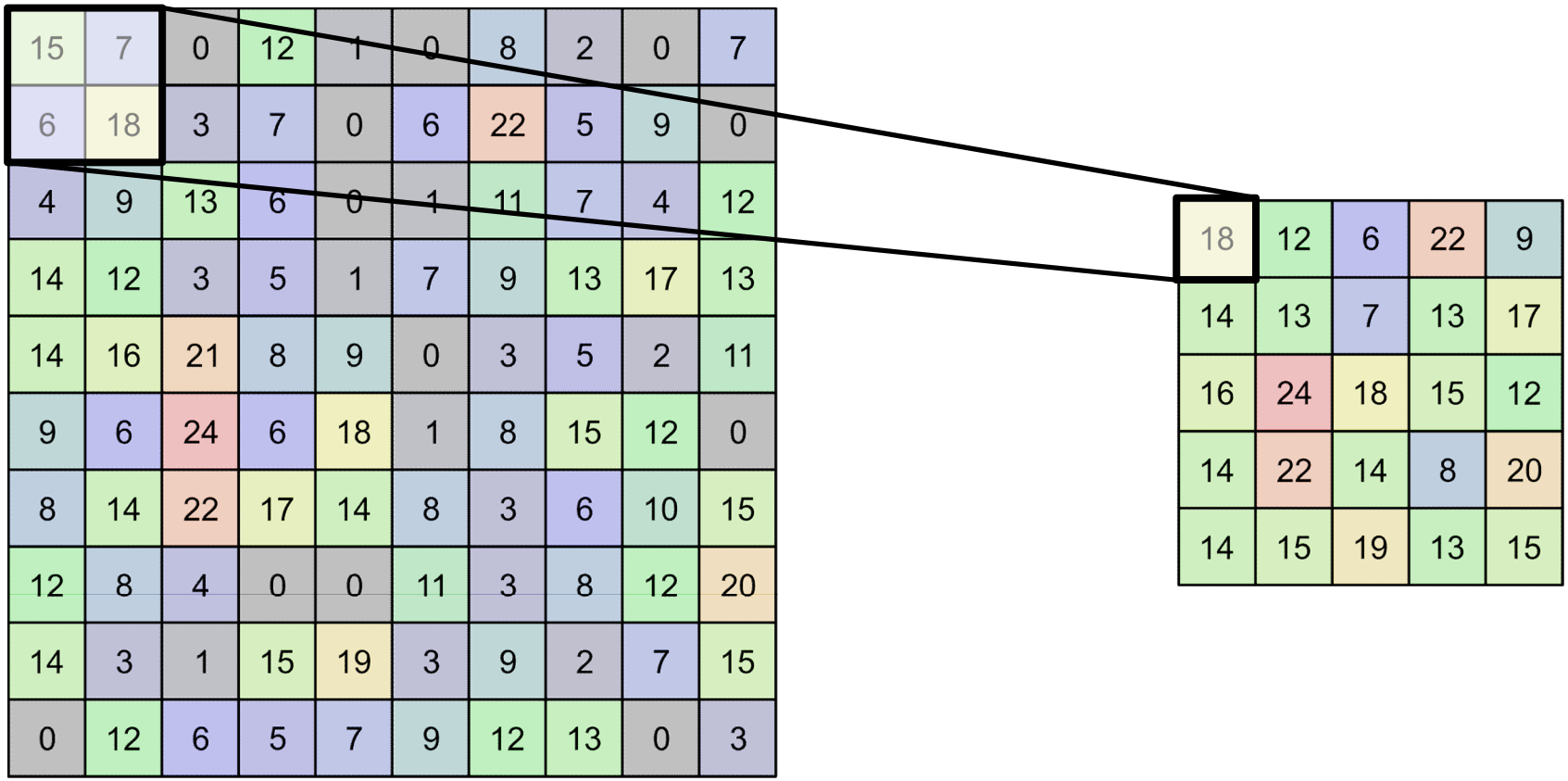
Filters combine abstract outputs of previous layers

- Don‘t show filter coefficients
- Generate images maximizing the response of selected neurons
  - ➢ See what the neuron “wants” to see

Max-Pooling to Increase Field of View



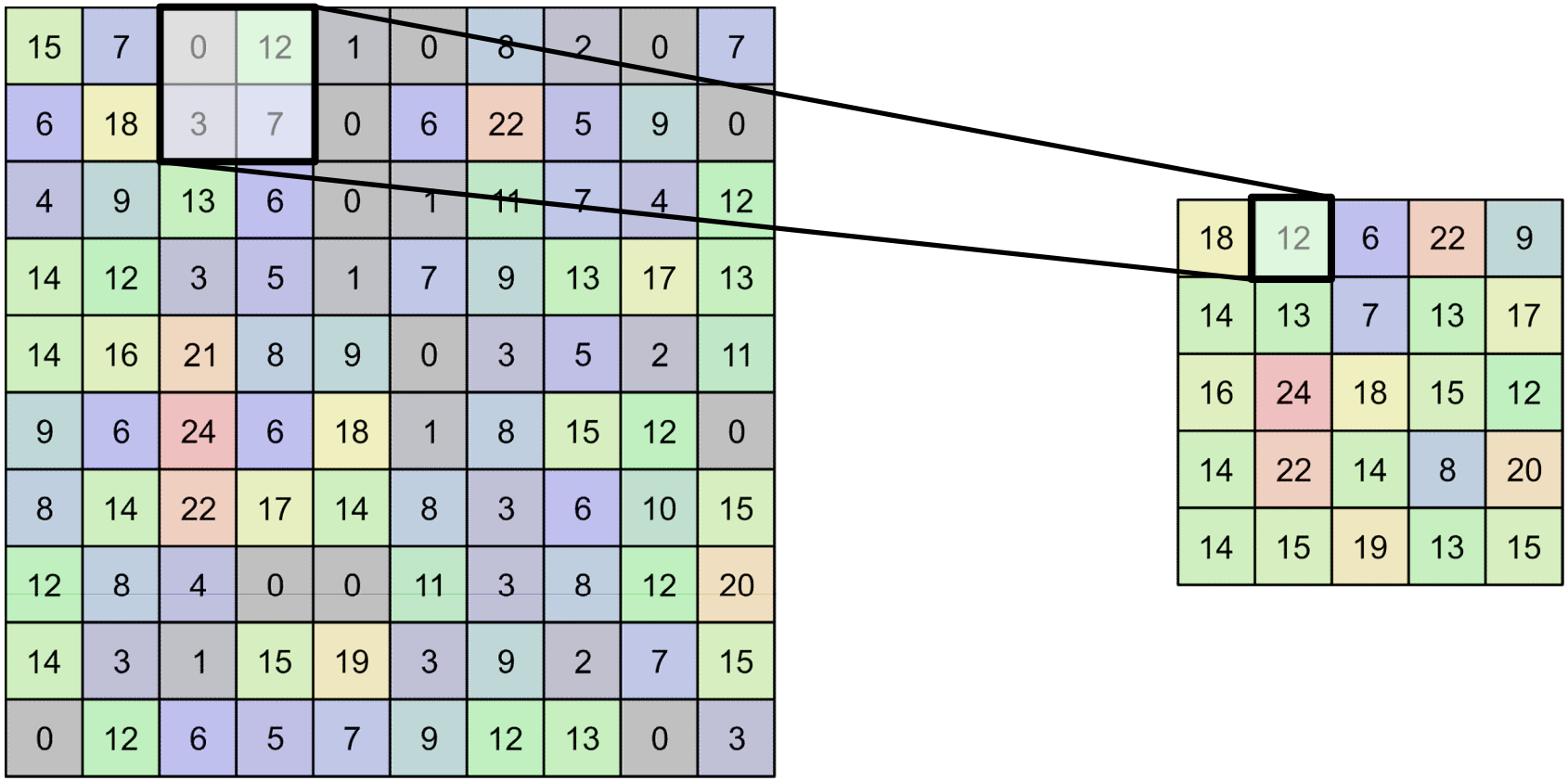
2D Max-Pooling



Simple sub-sampling

- → **No learned parameters**
- → **Reduces output dimensions** by factor 2
- → **Doubles the field-of-view** of output neurons for each dimension

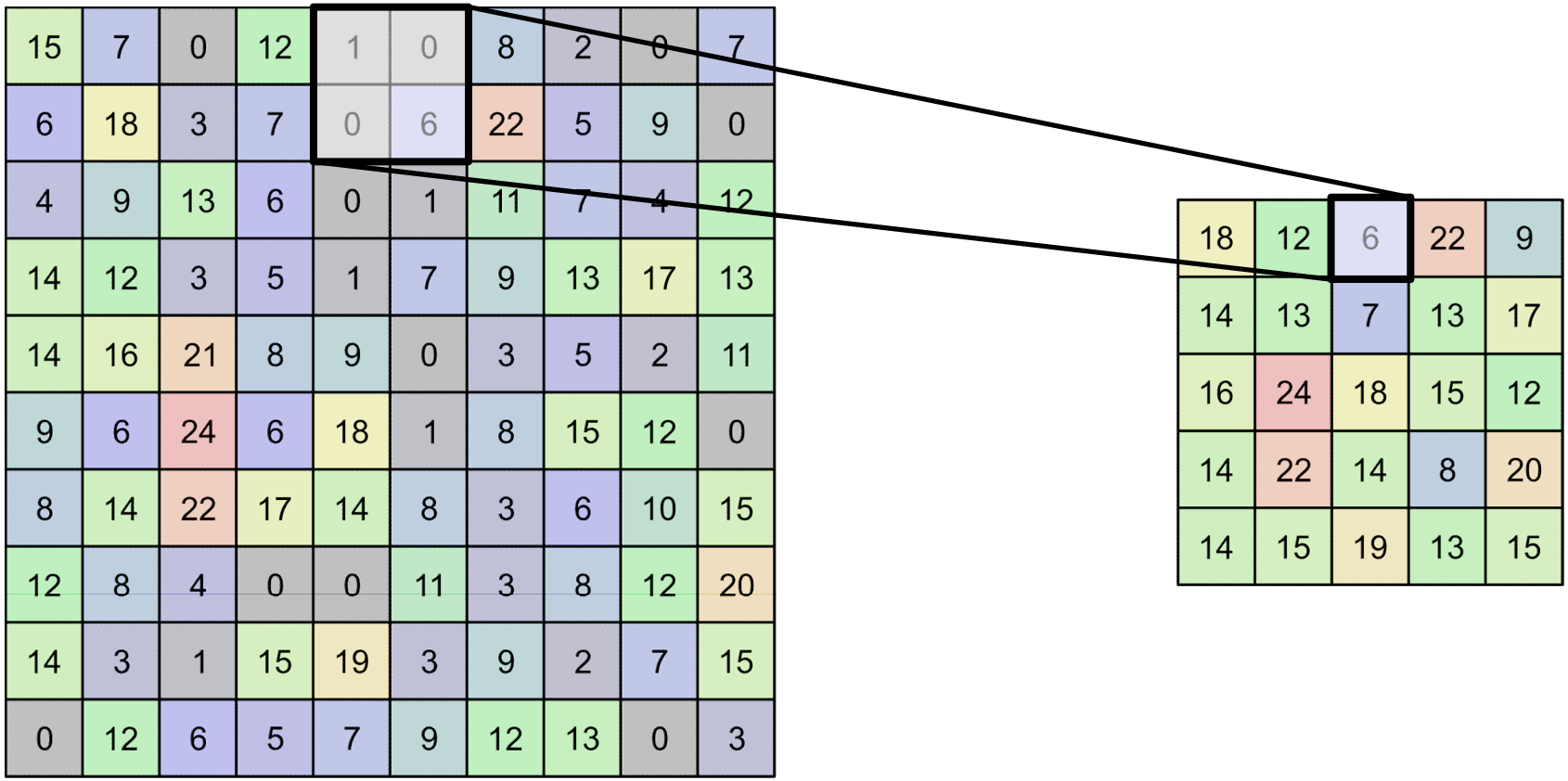
2D Max-Pooling



Simple sub-sampling

- → **No learned parameters**
- → **Reduces output dimensions** by factor 2
- → **Doubles the field-of-view** of output neurons for each dimension

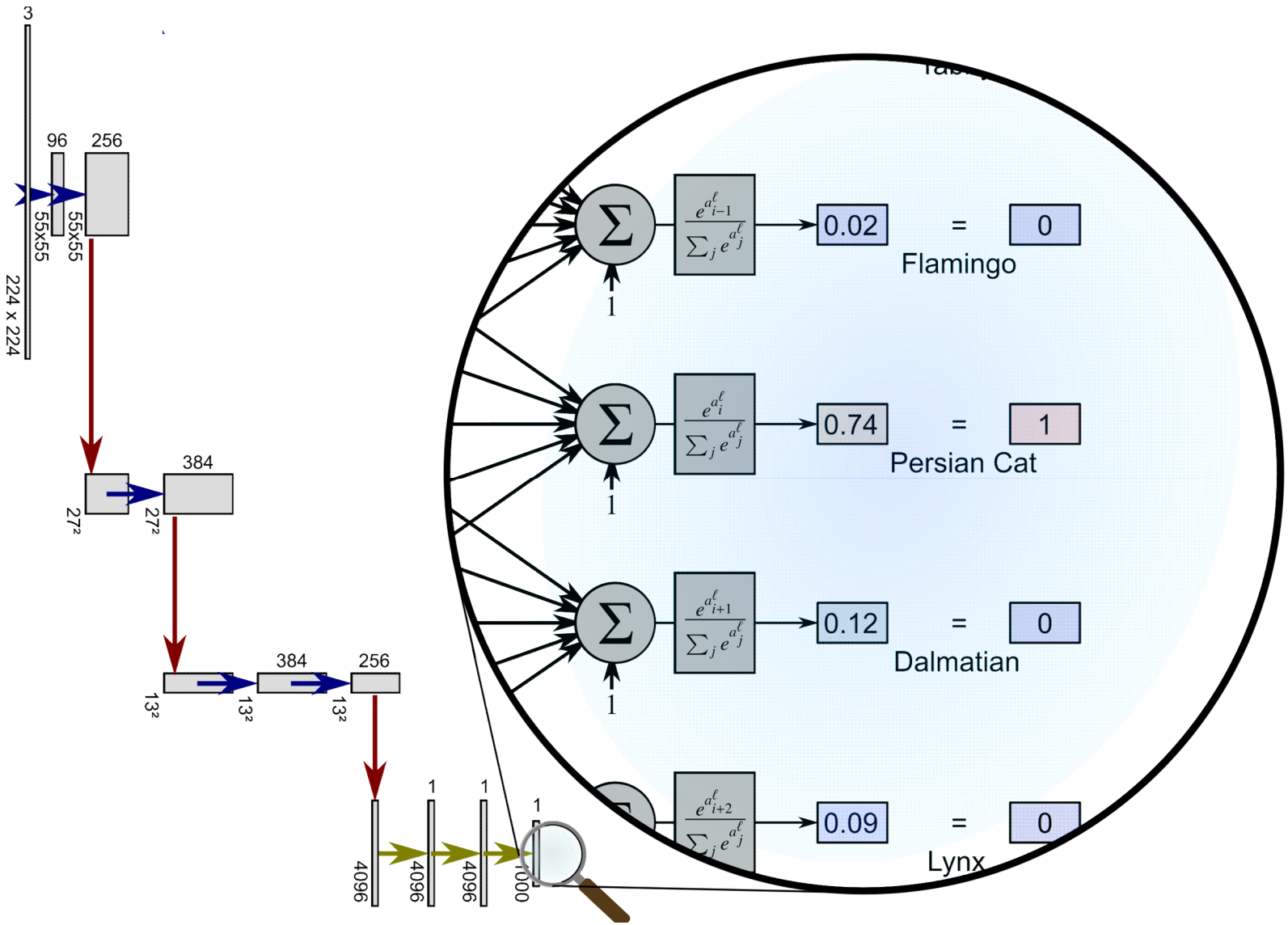
2D Max-Pooling



Simple sub-sampling

- → **No learned parameters**
- → **Reduces output dimensions** by factor 2
- → **Doubles the field-of-view** of output neurons for each dimension

The Objective Function



Training a Neural Network – The Objective Function

Store class label as {0,1} vector

1 = this class

0 = other class

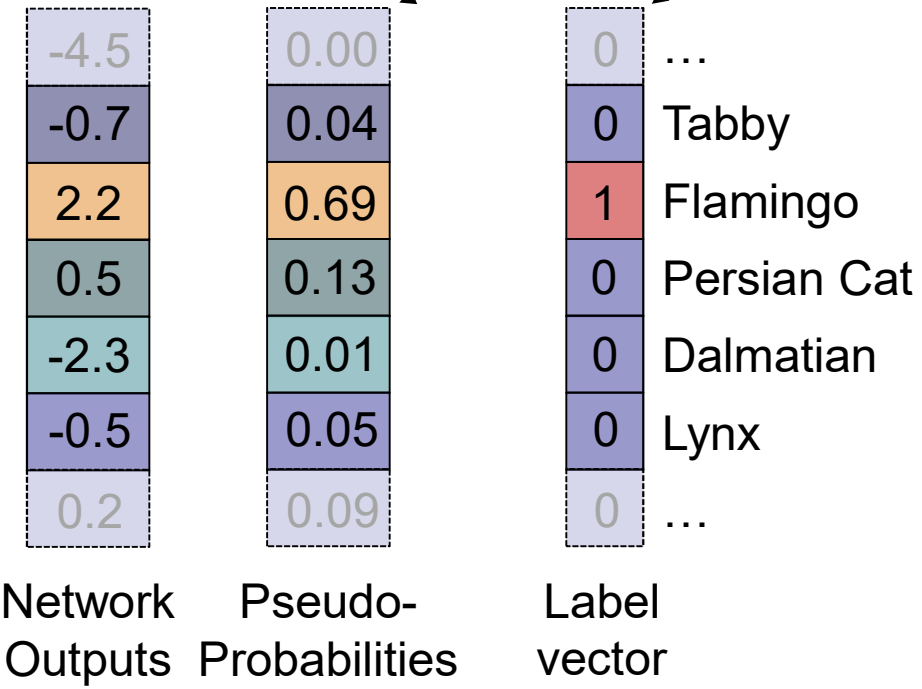
1. Convert **network outputs** to **pseudo probabilities** in [0,1]
2. **Compare** pseudo probabilites with label vector
3. **Sum** over all comparisons

→ **"Energy" E**

Minimize E using an optimizer

Parameters: all network weights

$$p_c = \frac{e^{a_c^\ell}}{\sum_{k=1}^{C} e^{a_k^\ell}} \qquad E = \sum_{c=1}^{C} \log(p_c) \cdot t_c$$



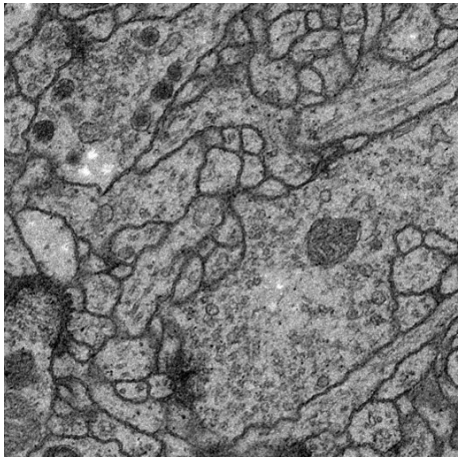
Application to (Real) Biomedical Problems

Can CNNs Also Solve Real Problems?

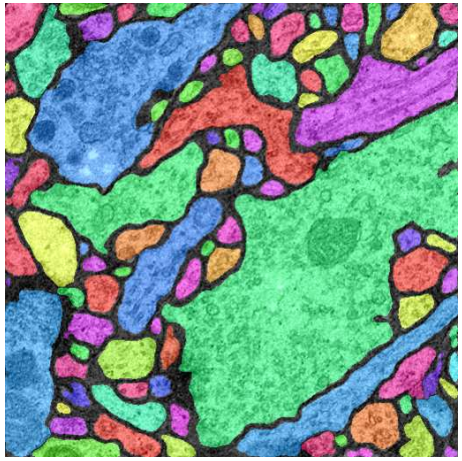
Image classification is of limited use in biology!

More interesting:

- Image segmentation
- Detection of specific structures



EM image



Segmentation

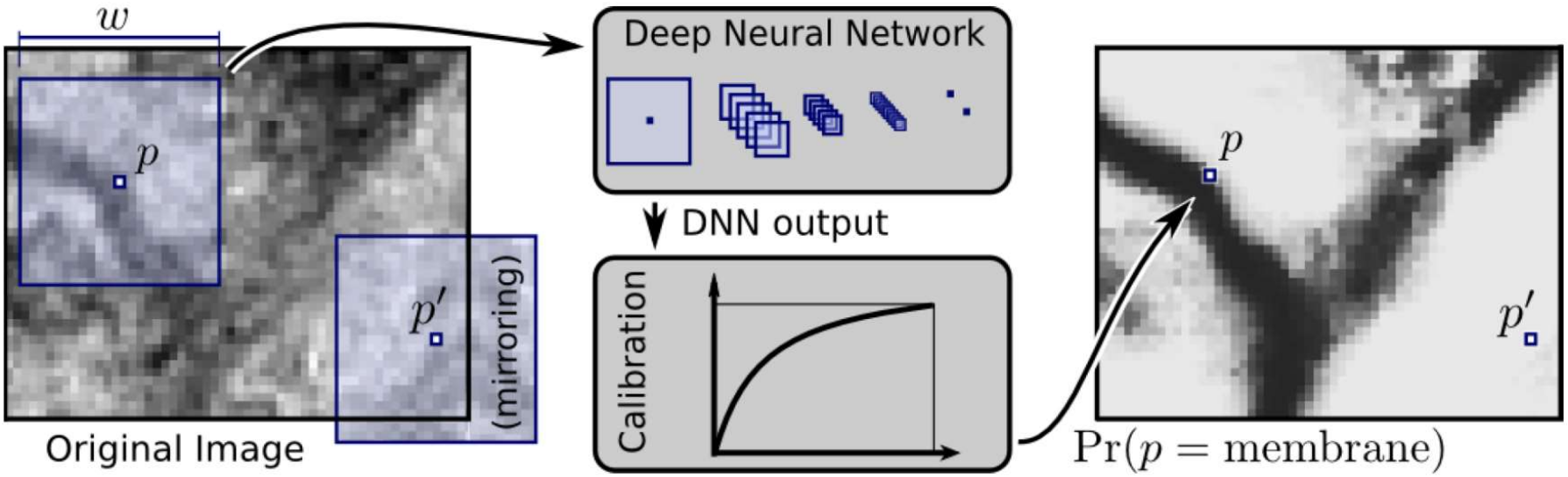
What is the difference?

- Classification: **One class label per image**
- Segmentation/Detection: **One class label per pixel**

Naïve Idea: Use classification network with sliding window

- One network evaluation per pixel ➔ very expensive, but works!

Pixel-wise classification using Sliding Windows



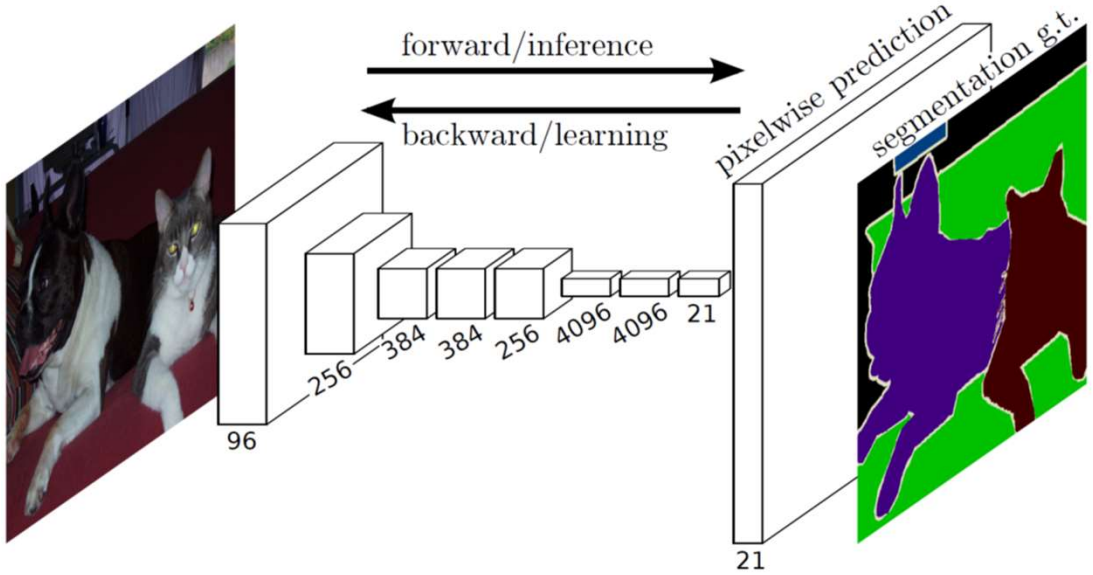
D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber - Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images (NIPS 2012)

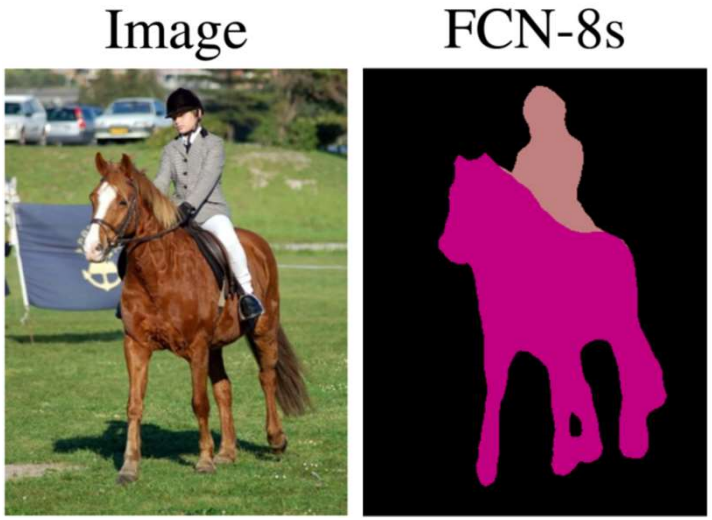
Winner of the EM segmentation challenge (ISBI 2012)

- ✓ Advantage: intuitive
- × One network evaluation per pixel ➔ extremely slow!
- × Evaluations highly redundant

Idea: Exploit redundancy ➔ parallelize classification

Fully convolutional networks



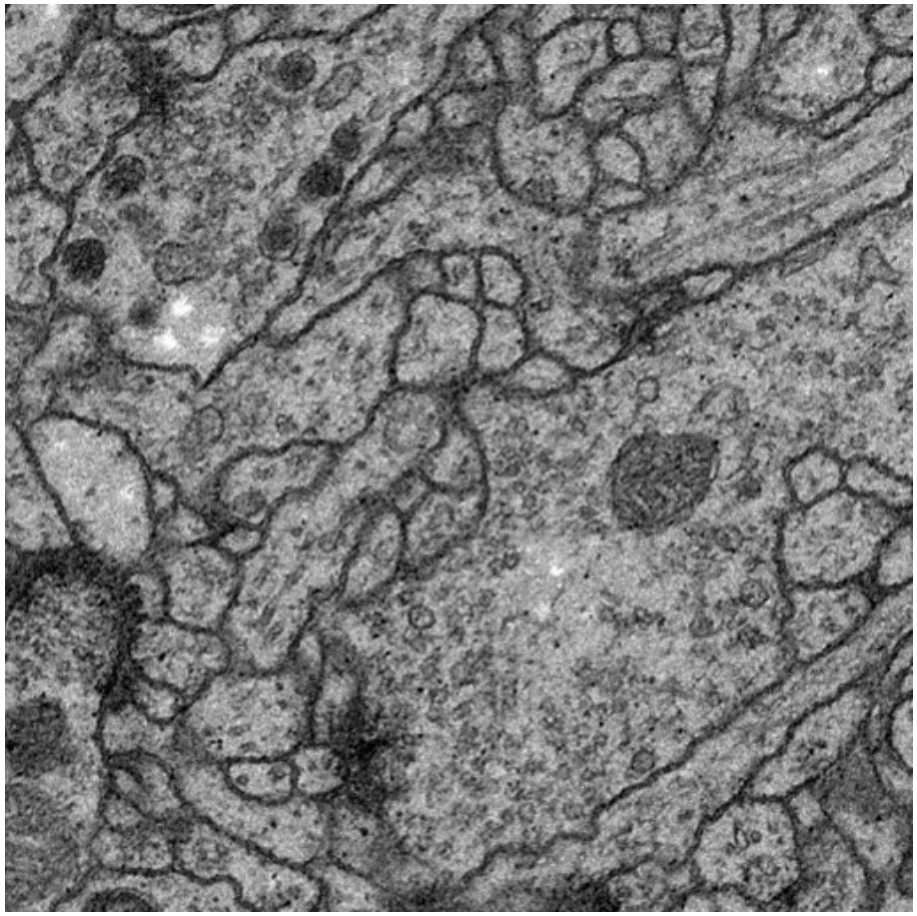


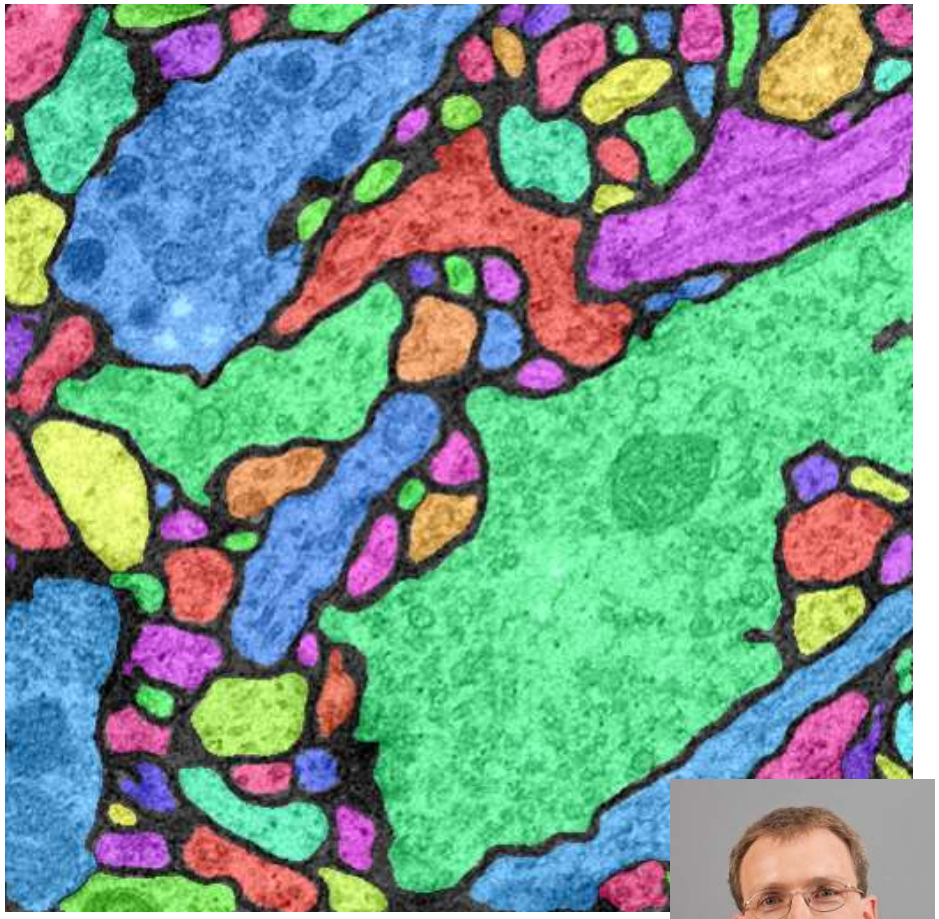
J. Long, E. Shelhamer, T. Darrell: "Fully Convolutional Networks for Semantic Segmentation". CVPR 2015, arXiv:1411.4038 [cs.CV]

Replace fully connected layers by up-sampling steps

- **Outputs a (lower-resolution) image** instead of a class label
- Uses a **pre-trained classification network**
- Only **one feature channel per class** at lowest resolution
- Uses **zero-padded convolutions** ➔ no tiling possible

Applications



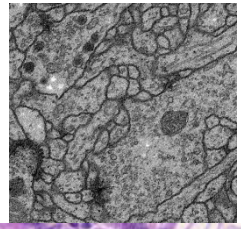


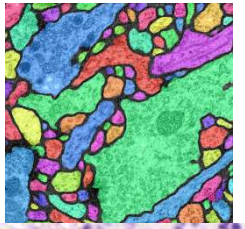
Membrane Segmentation in electron microscopic (EM) images

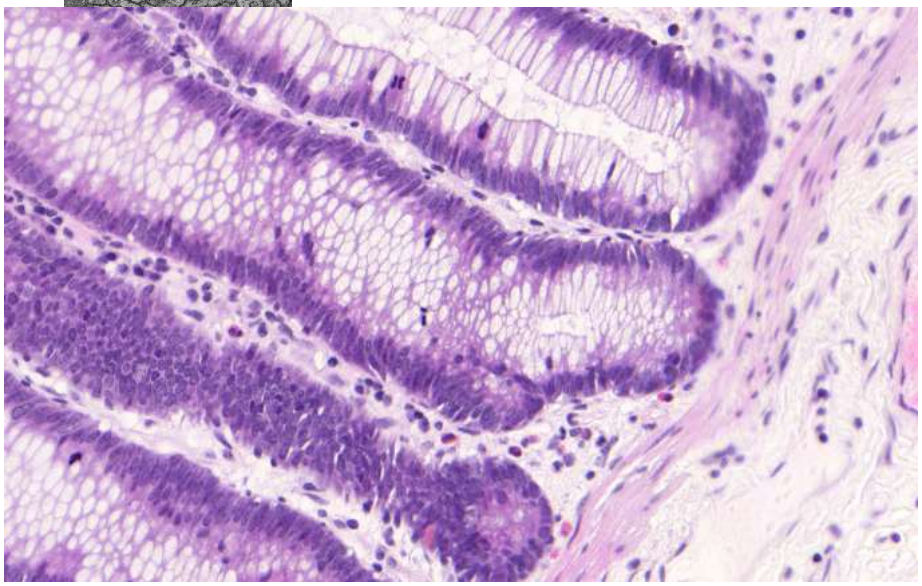
- 2D/3D, two classes: Membrane / Rest
- Challenges: similar structures, weak contours, topology preservation

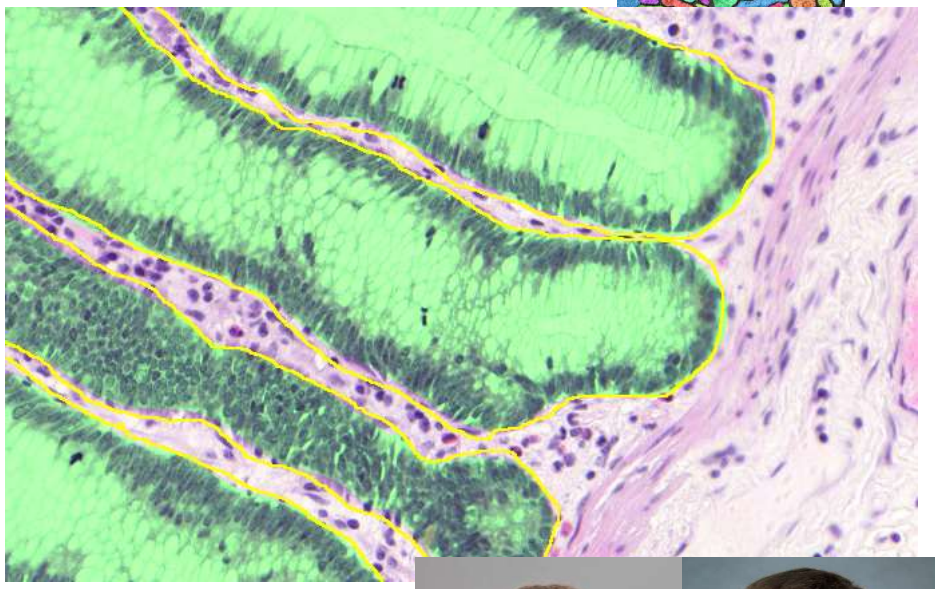
Olaf Ronneberger

Applications





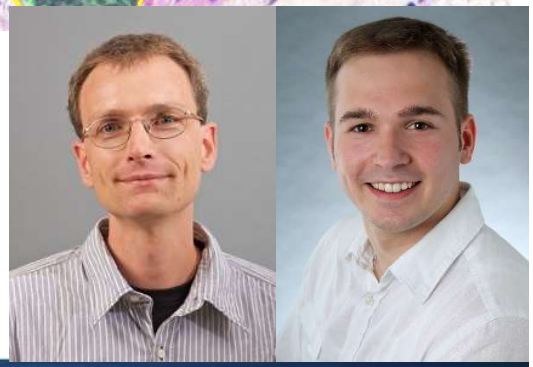




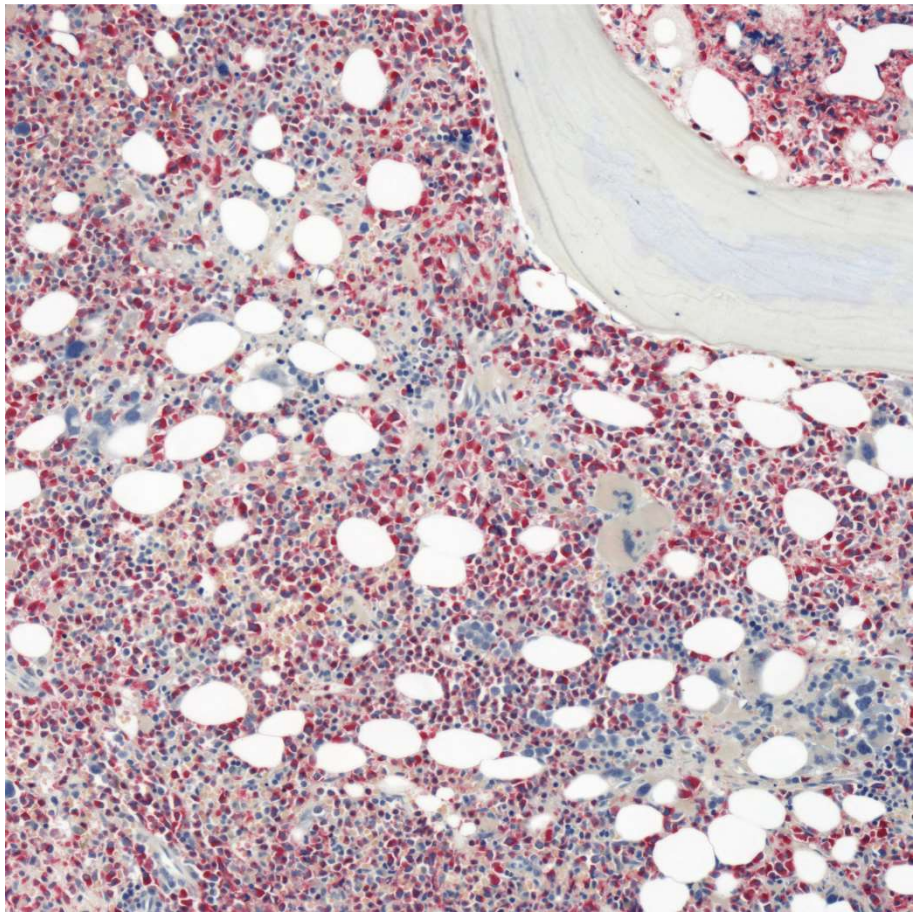
Gland Segmentation in histologically stained tissue sections

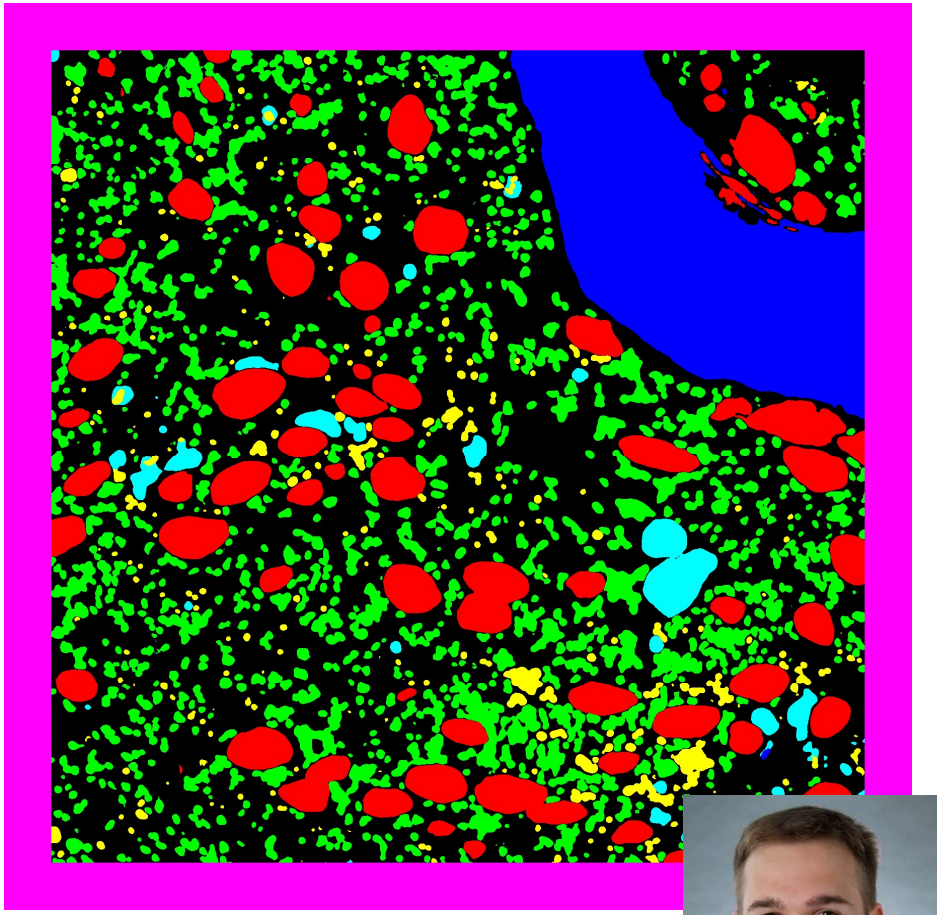
- 2D, two classes: Gland / Rest
- Challenges: Similar textures, large structures

Olaf Ronneberger & Anton Böhm



Applications



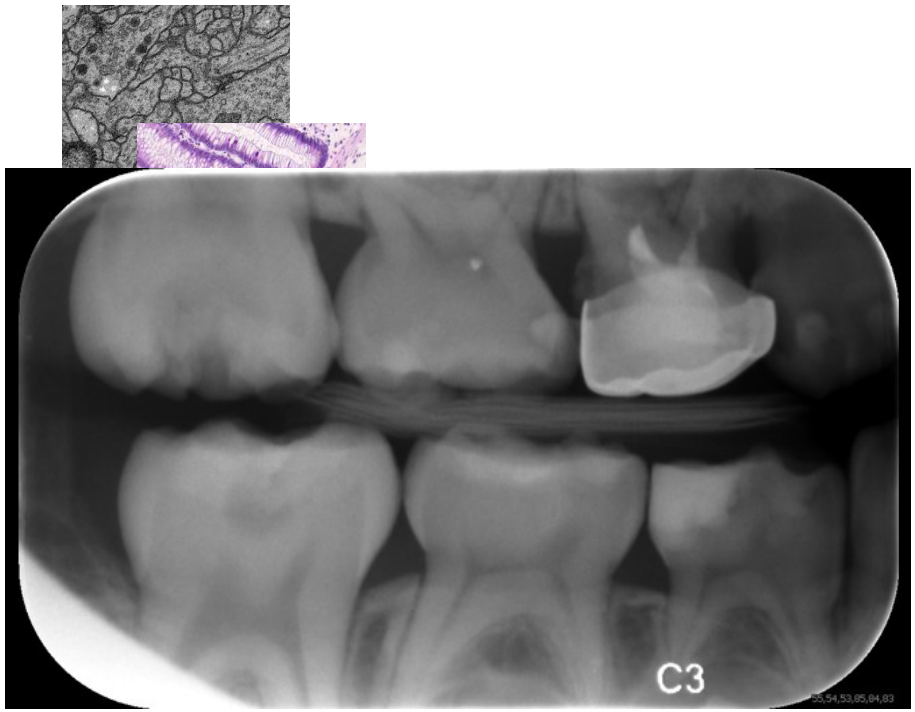


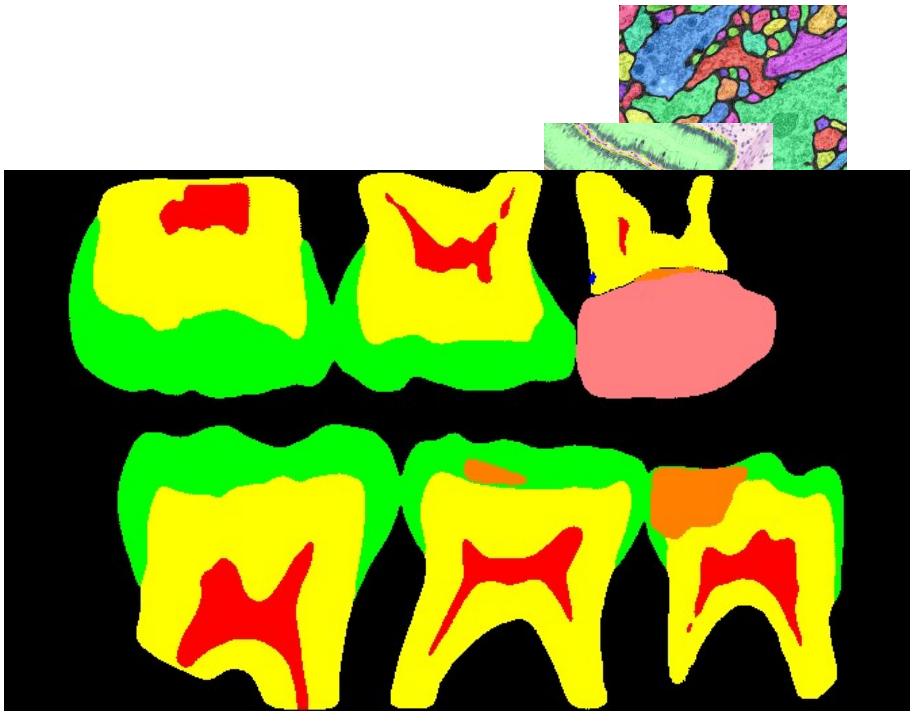
Segmentation and classification in histologically stained tissue sections

- 2D, 6 classes: Granulo-, Erythropoesis, Megacaryocyte, Fat, Bone, BG
- Challenges: Similar textures, large structures, variable staining quality

Anton Böhm

Applications



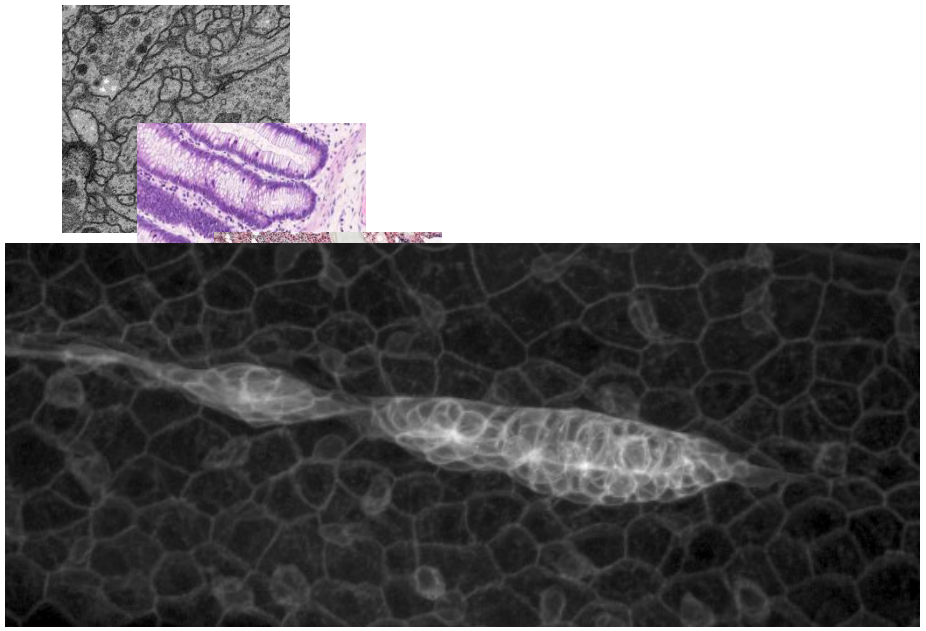


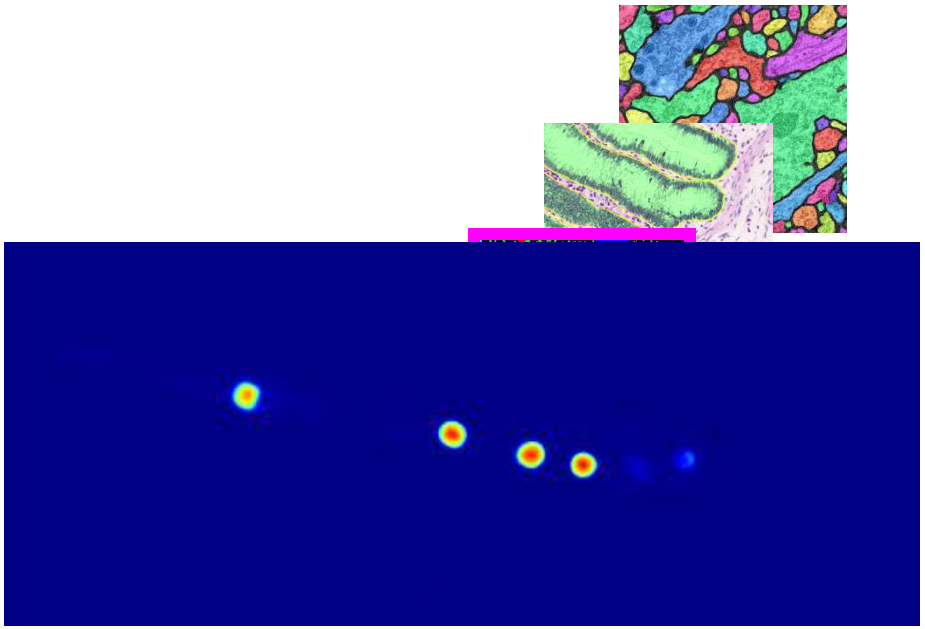
Segmentation and classification in dental x-ray images

- 2D, 8 classes: Dentin, Enamel, Pulp, Caries, …, BG
- Challenges: Weak contrast, strong image quality variations

Olaf Ronneberger

Applications



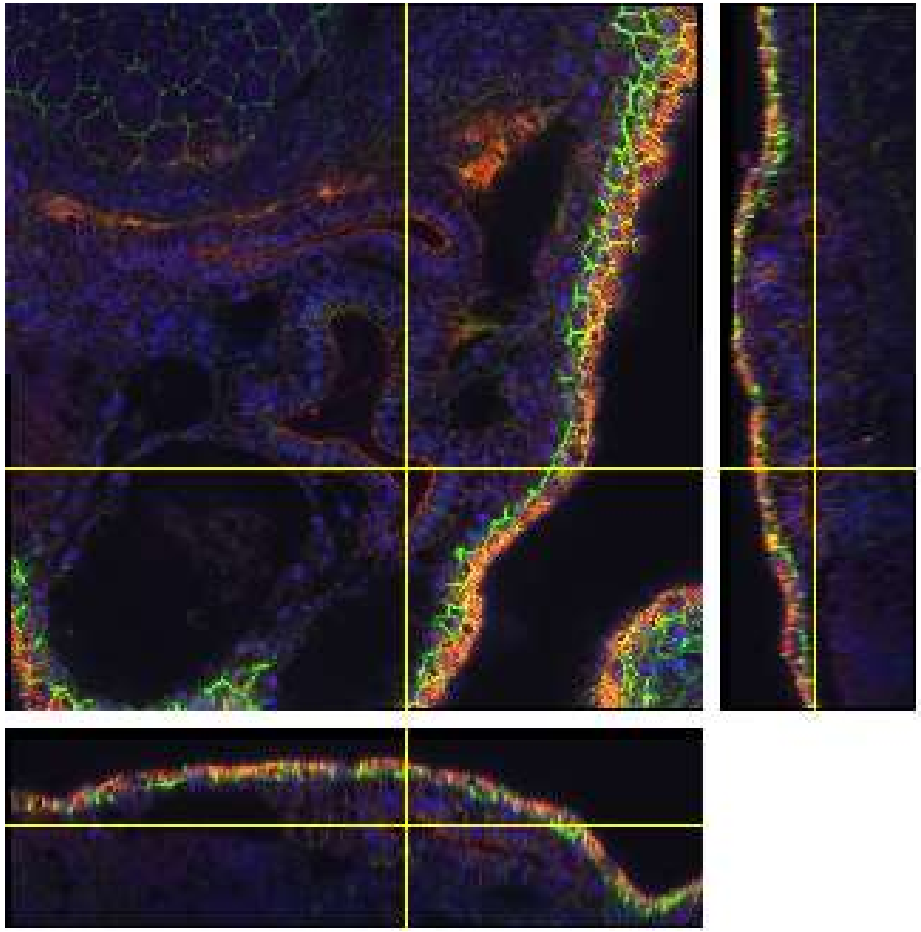


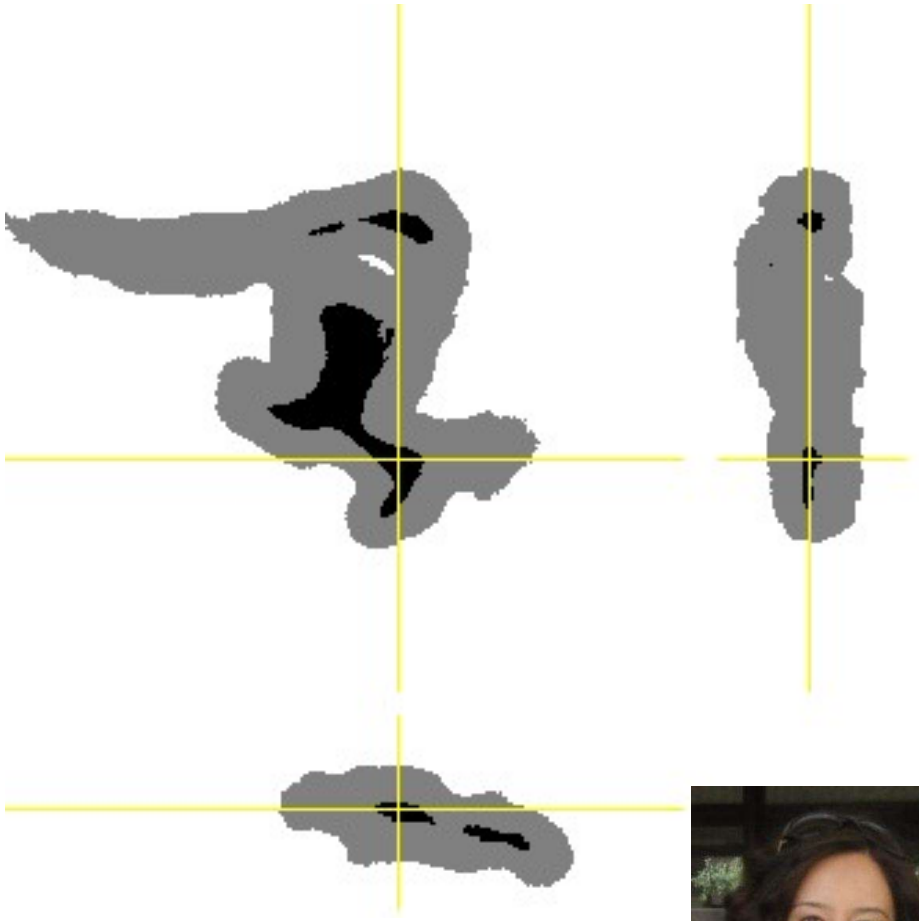
Rosette-Detection and tracking in Zebrafish primordia

- 2D+t, 2 classes: Primordium / Rest
- Challenges: Subtle cues in dense tissue

Robert Bensch

Applications



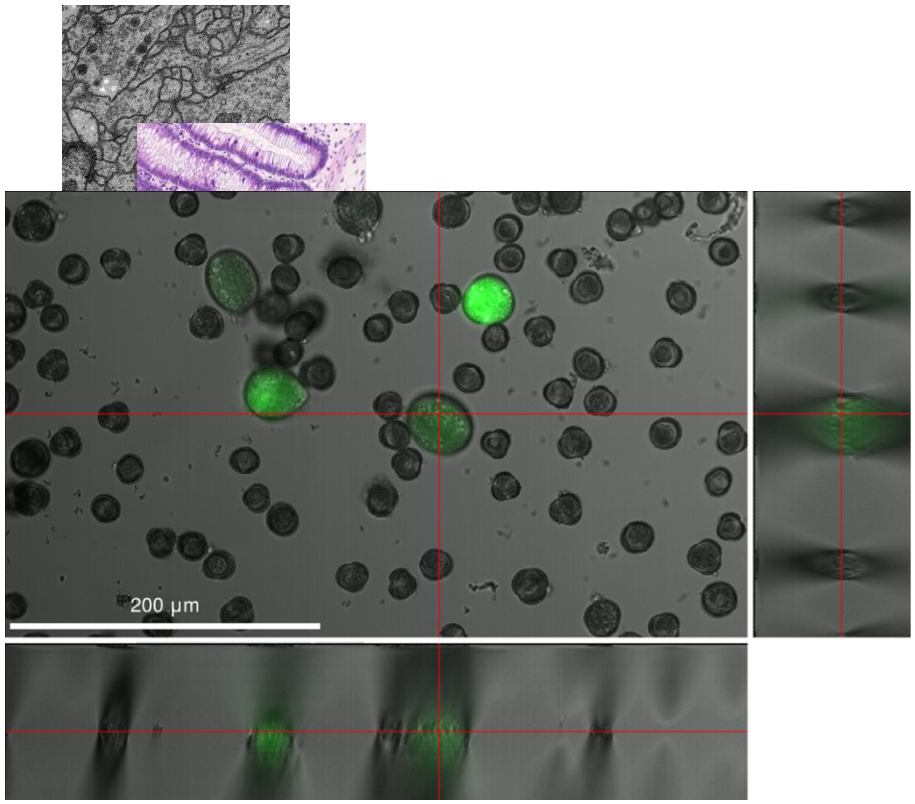


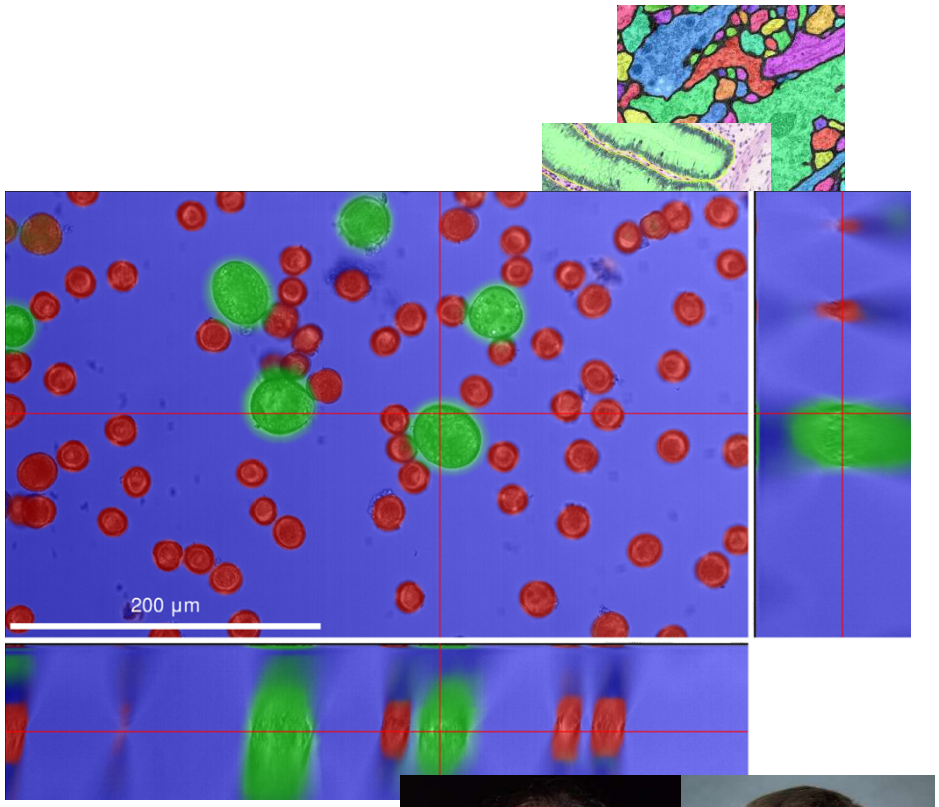
Renal tube segmentation in Xenopus

- 3D, 3 classes: Inner tube, Tube wall, Rest
- Challenges: Sparse Annotations, Structure similarities

Özgün Çiçek

Applications



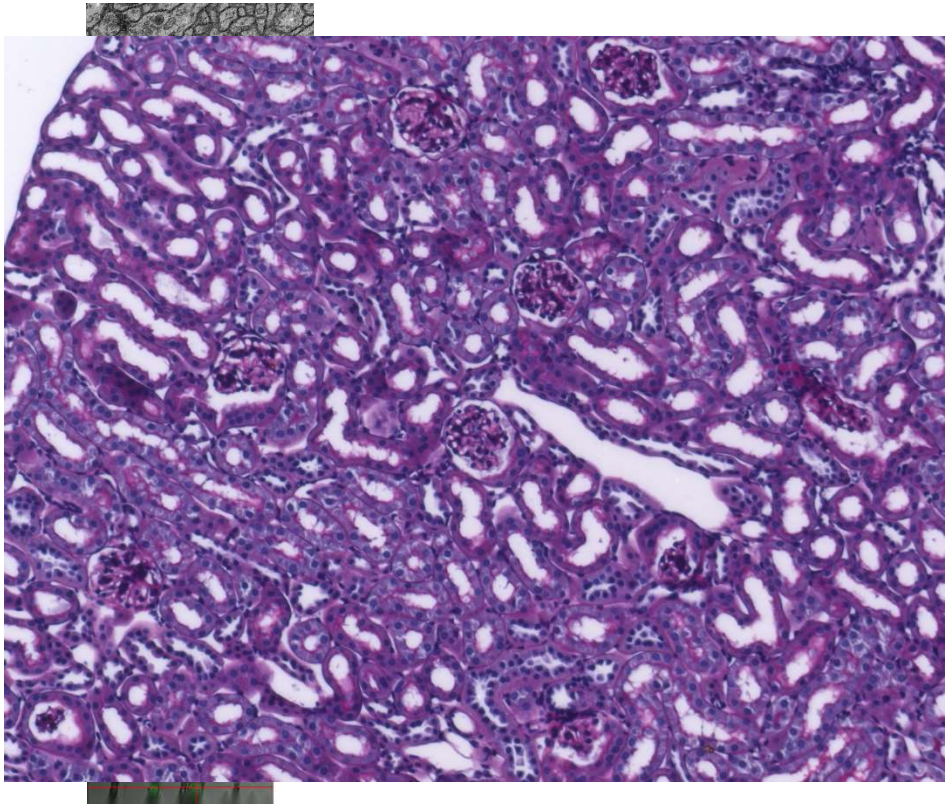


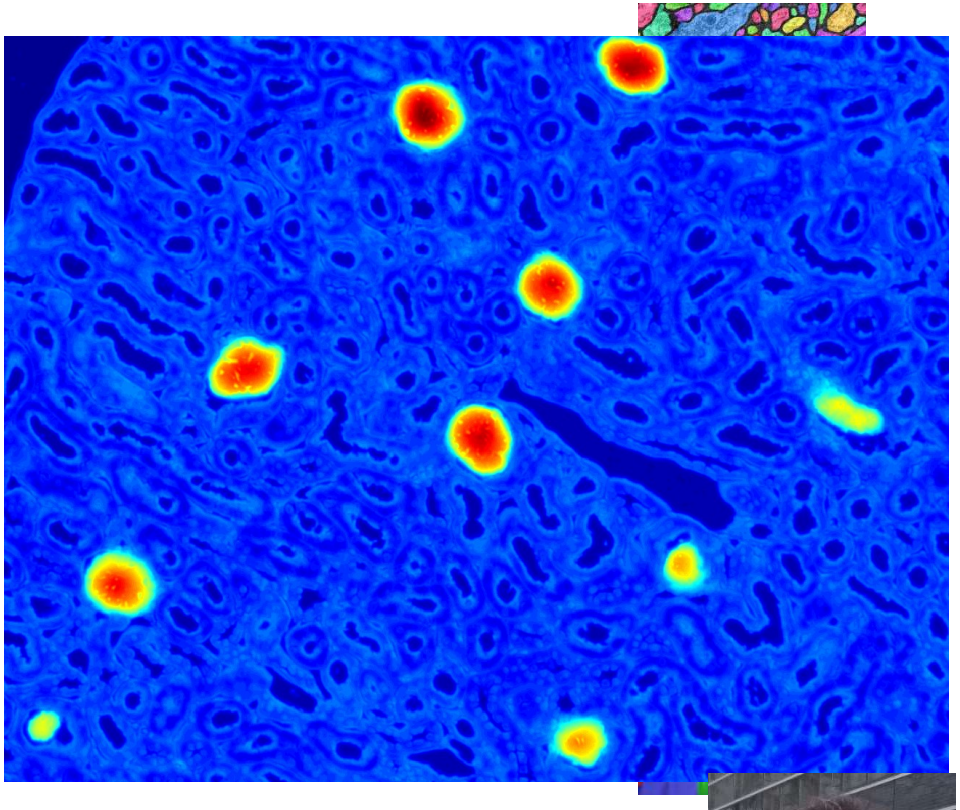
Microspore segmentation and classification (living/dead)

- 2.5D, 3 classes: Living, Dead, Background
- Challenges: Inaccurate automatic ground truth

Thorsten Falk & Anton Böhm

Applications



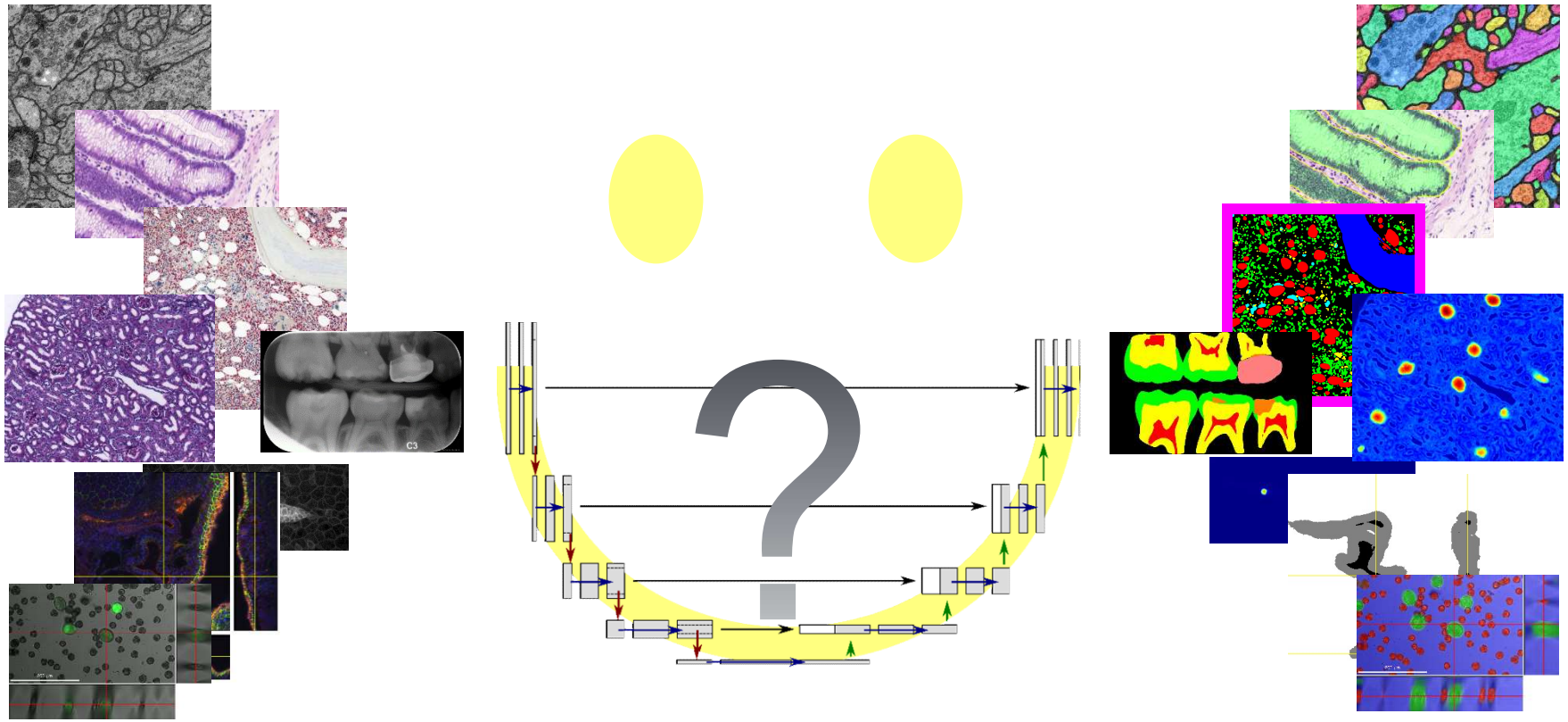


Glomeruli segmentation in histochemically stained renal tissue sections

- 2D, 2 classes: Glomerulus / Rest
- Challenges: Subtle texture differences, staining variations

Dominic Mai

Applications



Idea: Refine fully convolutional network of Long et al.

Pre-training on natural images is of little use

- **End-to-end training from little training data** required!

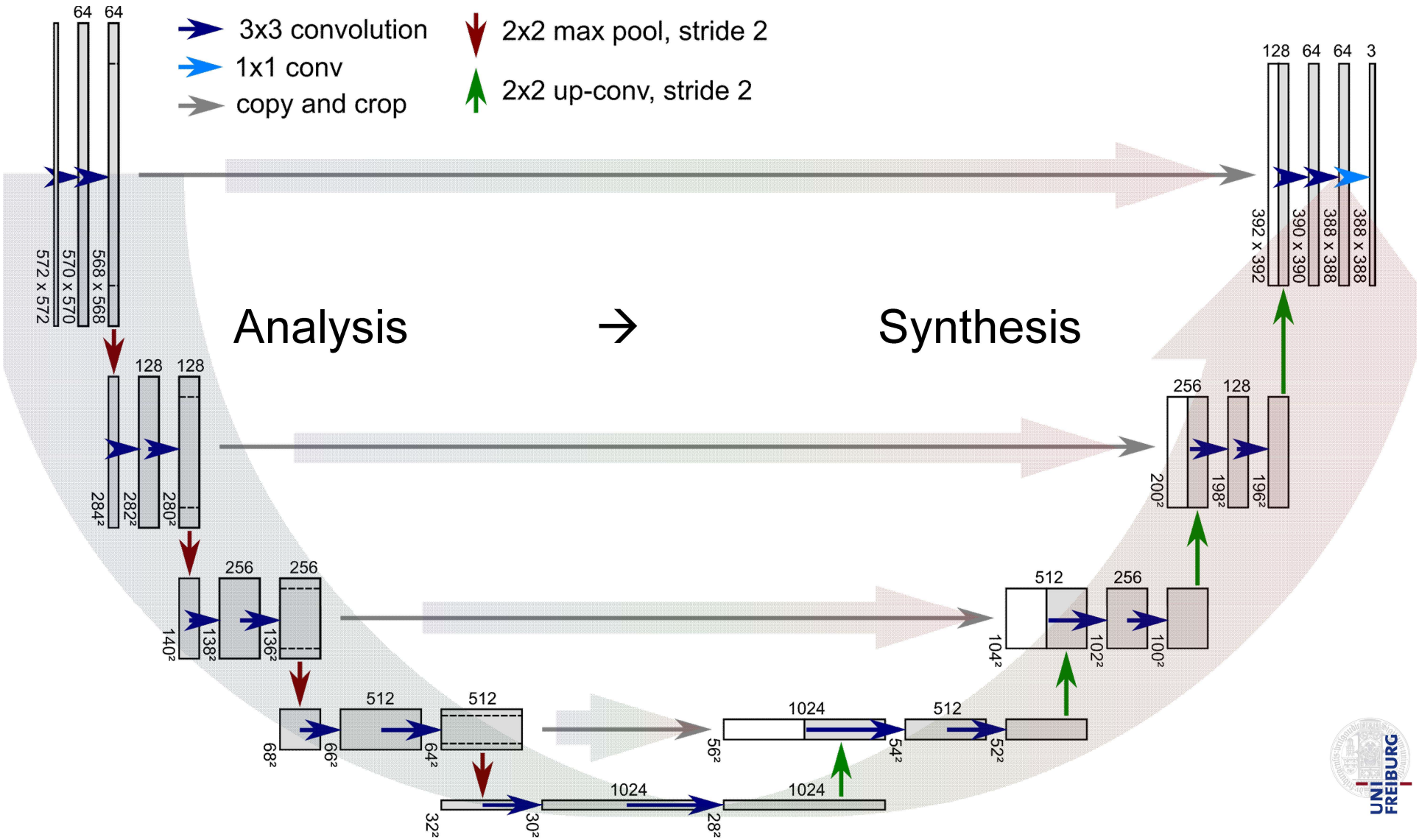
Low-dimensional feature representation sacrifices resolution

- Retain **high-dimensional features for „smart image synthesis“**
- Also use high-resolution features for synthesis (**shortcut connections**)

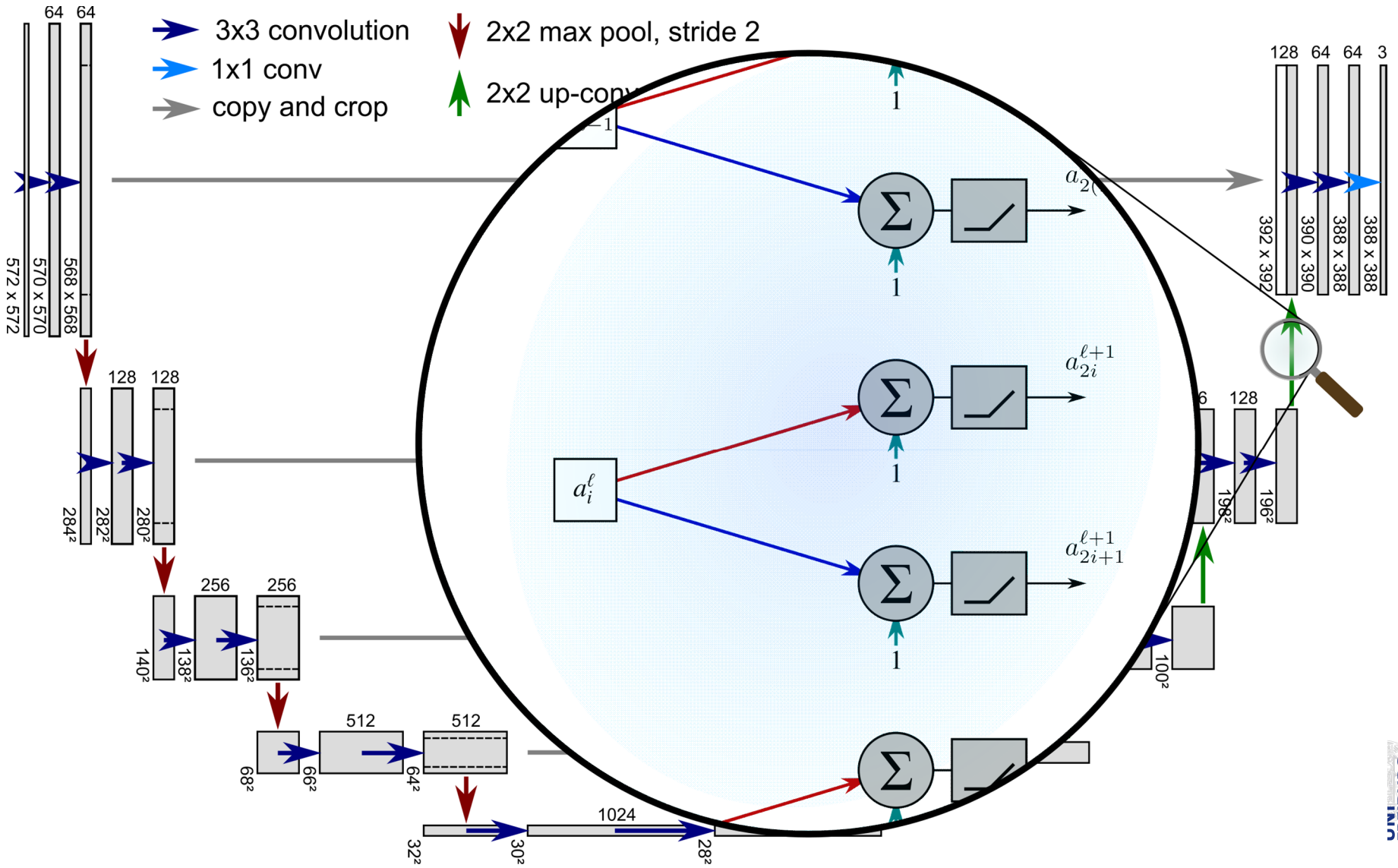
Padded convolutions restrict network to fixed input image size

- Use only **valid part of convolutions** to allow **segmentation of arbitrarily sized images**

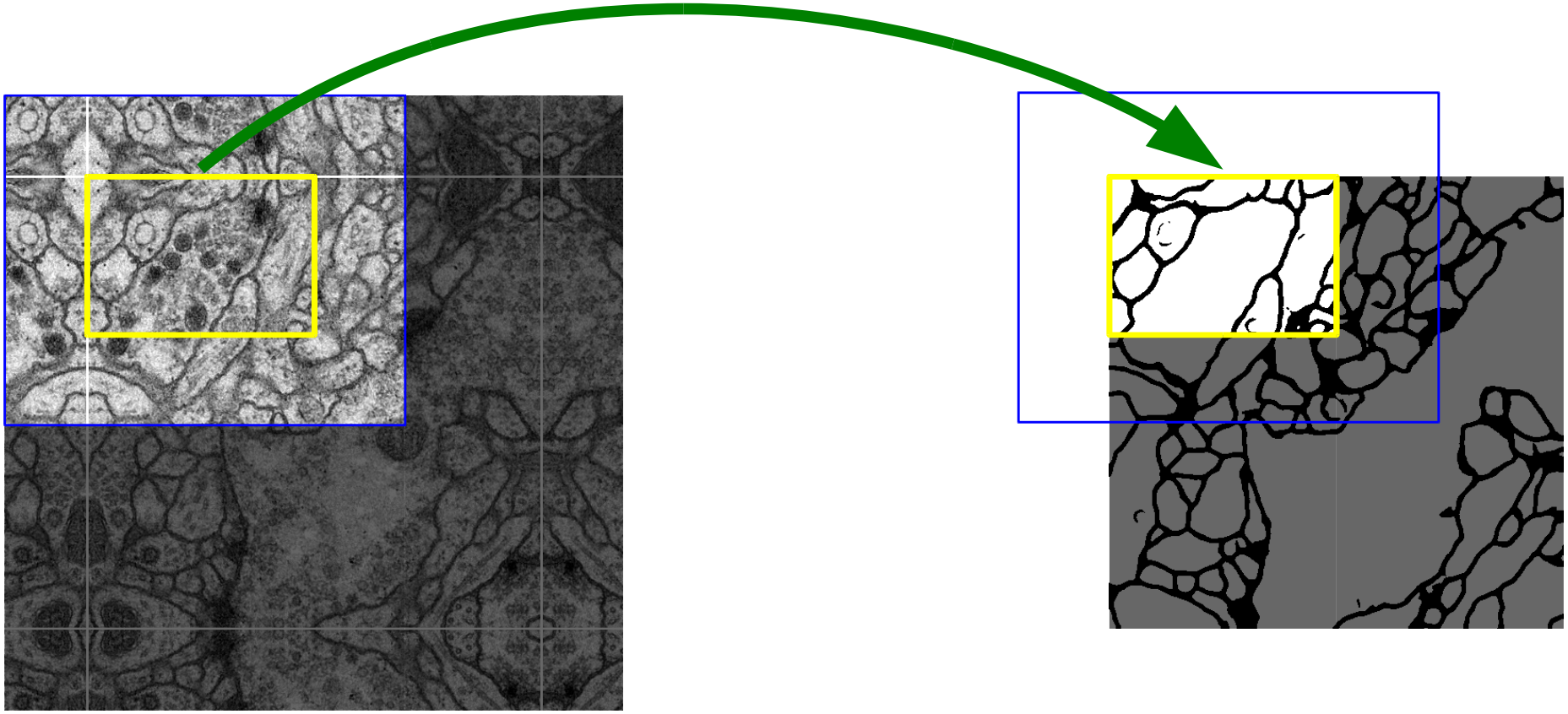
CNN for Image Analysis and Synthesis: The U-Net



Up-Convolutions to Recover Original Resolution



Boundary Treatment



- Segmentation of the yellow area needs input data of the blue area
- Raw data extrapolation by mirroring

Generation of Training Data

Many images with manual annotation required!?

- Depends on complexity of the problem and the network
- For simple biomedical applications few images may be sufficient

Biomedical data

- ✓ Scales are known
- ✓ No projective distortions
- ✓ No (self-)occlusion
- ✓ Position and orientation of structures often irrelevant

- ✗ High variability
  - Biological variations
  - Sample preparation
  - Imaging
- ✗ Subtle differences relevant
- ✗ Out-of-focus regions
- ✗ Ambiguous annotation

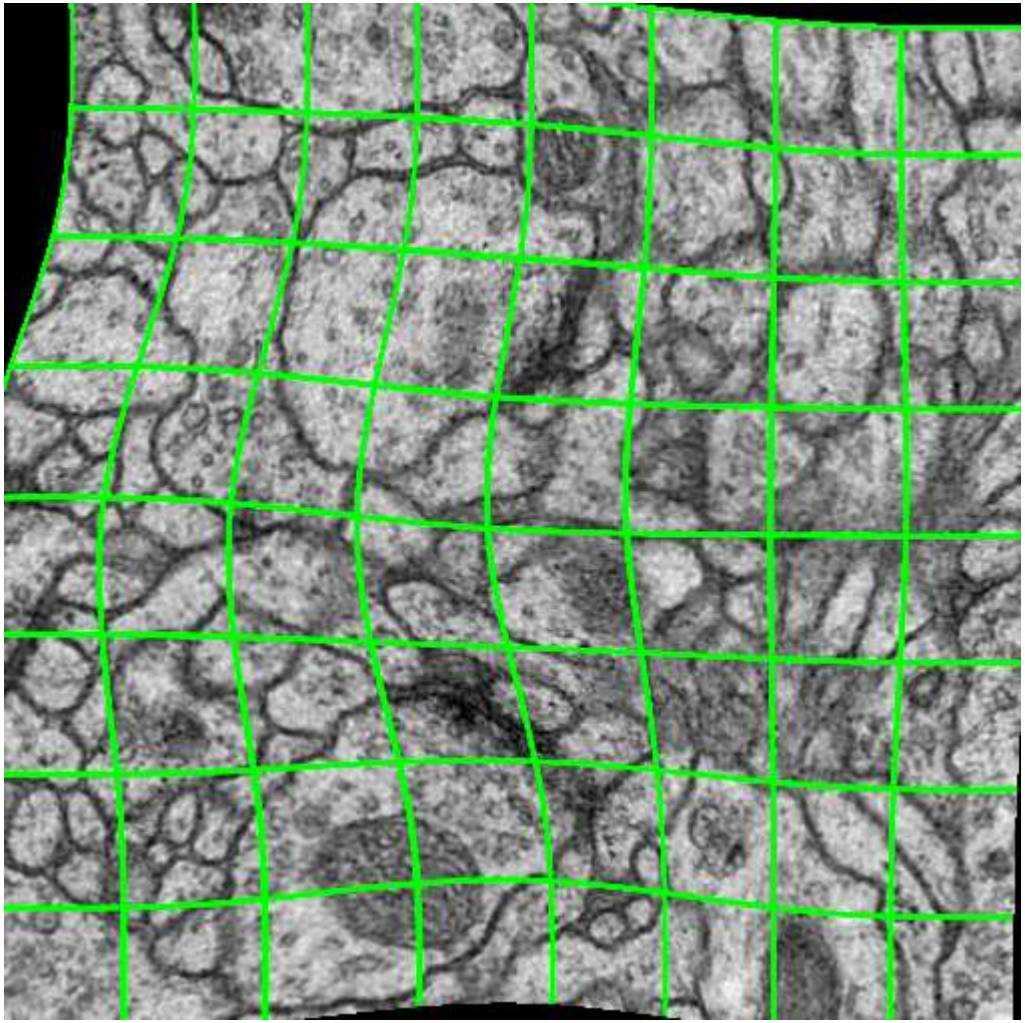
Virtual Training Samples via Data Augmentation

Many transformations do not alter biological meaning:

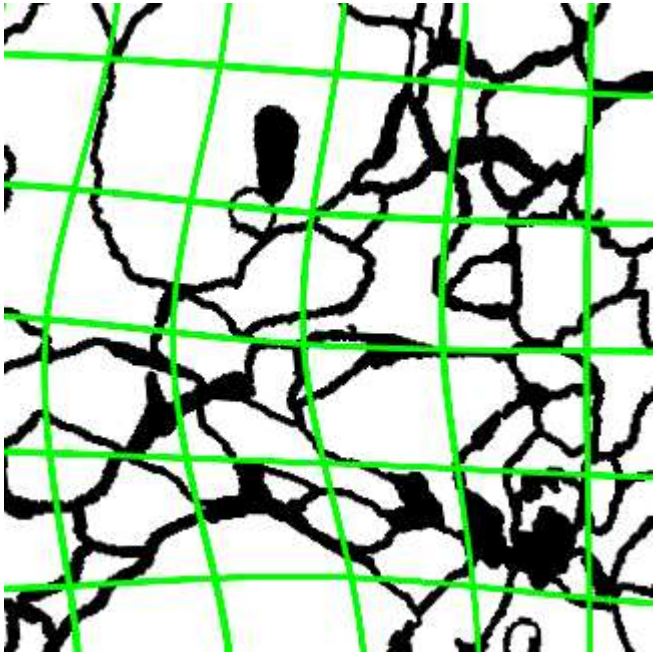
- **Shift and Rotation** irrelevant
  → Shift and rotate original images and annotations
- **Smooth shape variation** often irrelevant
  → Apply smooth deformations to original images and annotations
- Absolute **brightness and contrast** irrelevant
  → Apply different intensity transformations to original images
- **Noise** irrelevant
  → Add different noise levels to original images
- **Additional network regularization** possible
  – Dropout, Weight-Decay, …

Few annotations can yield large training set!

Virtual Training Samples via Data Augmentation

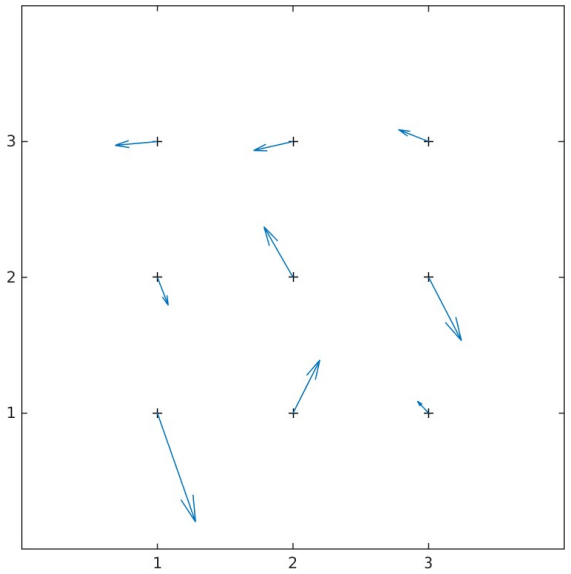


Randomly deformed image
(for visualization: no rotation, no shift, no extrapolation)

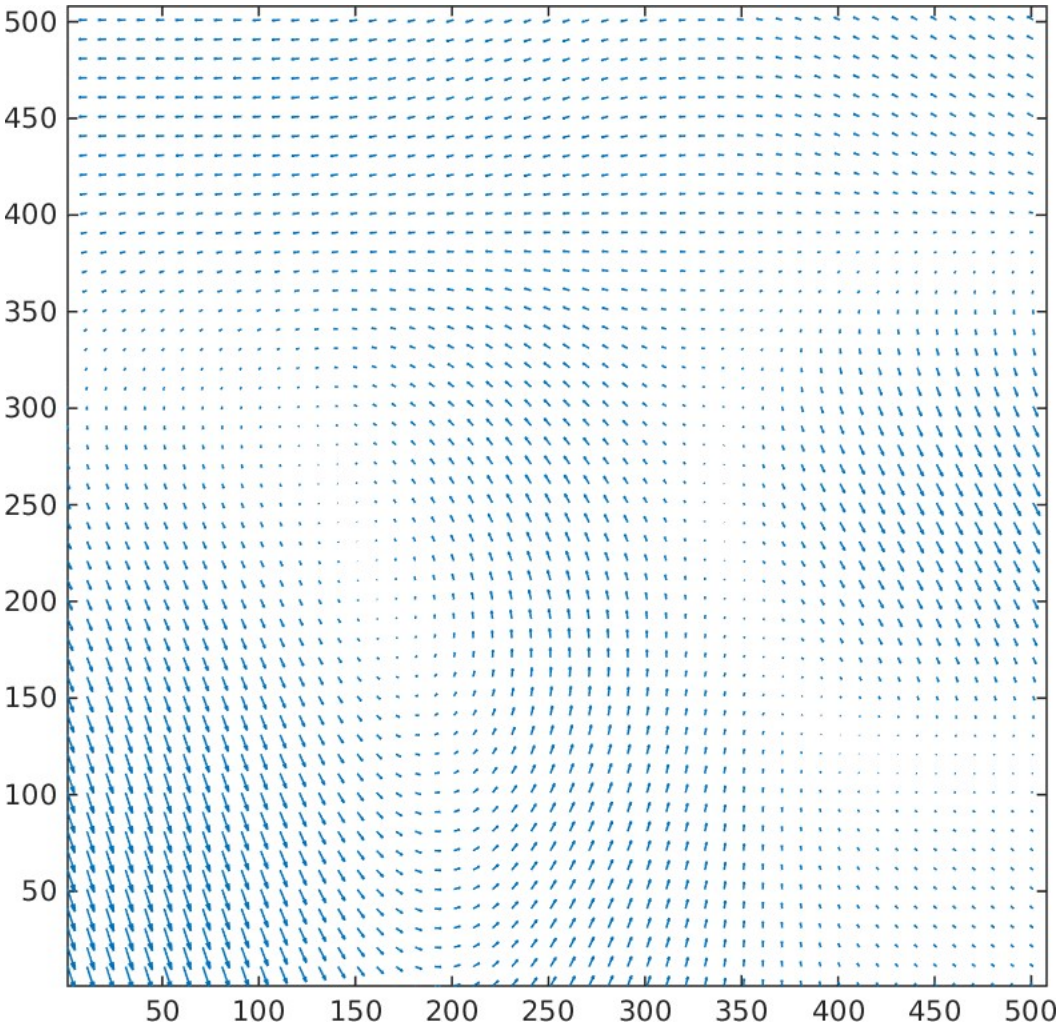


correspondingly deformed
manual labels

Deformation Field Generation

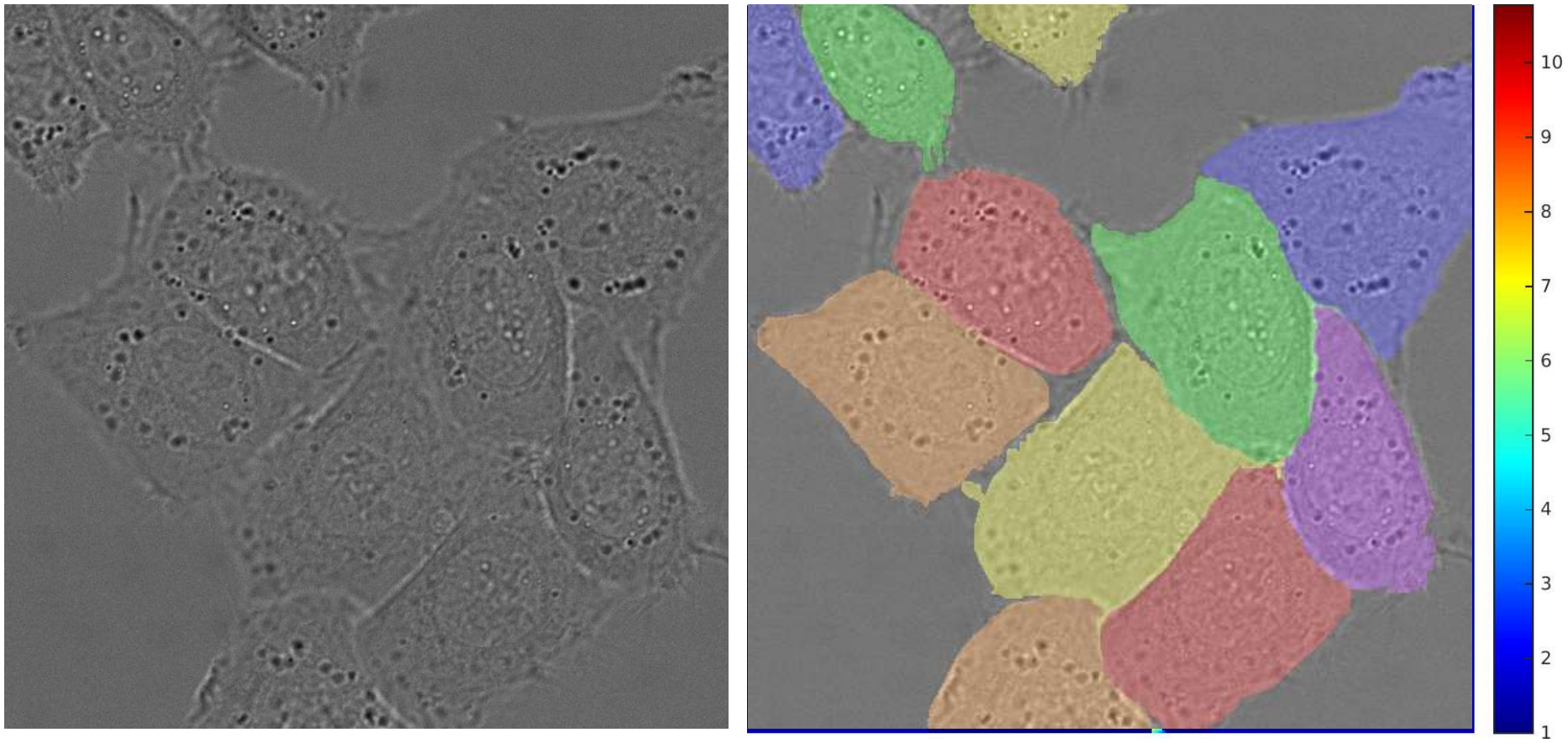


3x3 random deformation vectors



Full deformation field by **bicubic interpolation**

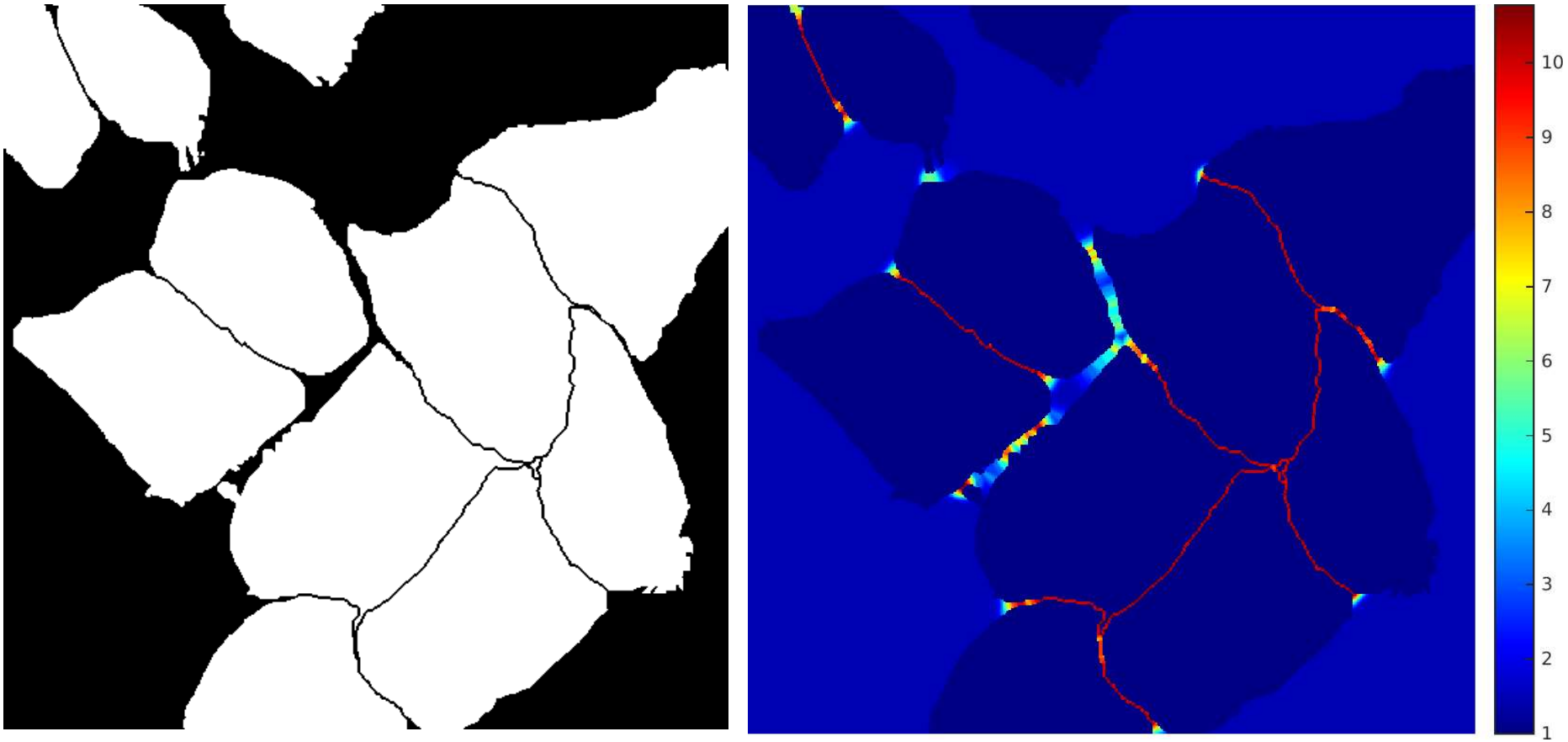
Separation of Touching Instances



Semantic segmentation vs. Instance segmentation

- Not only cell/no cell, but one instance label per cell
- If instances touch, semantic segmentation merges them

Separation of Touching Instances



Ad-hoc idea:

- Add one pixel-wide background region betweeen instances
- Give these interfaces high weight during training

Summary

Deep networks are extremely flexible

- Learn full image analysis pipeline from examples
- Allow classification, detection, segmentation, tracking, …
- Many more applications:
  - speech recognition
  - Natural language processing and automatic translation

Data augmentation allows training with few annotated images

Network training between hours and weeks

- Infeasible without parallel GPU hardware

Network evaluation is extremely fast

- Full image segmentation within seconds on a GPU

Programming Assignment

- Get familiar with the caffe neural network framework http://caffe.berkeleyvision.org
- Train a neural network for handwritten digit recognition (MNIST): http://caffe.berkeleyvision.org/gathered/examples/mnist.html This can be done on CPU or GPU
- If you don‘t want to compile caffe yourself (quite a lot of third party libraries), we provide a ready compiled version (Linux, 64bit) on the course webpage
- Optional: Do some cool stuff with the ready trained networks from the „model zoo“ on the caffe homepage. Here you might really need a GPU. The pool computers are equipped with an nVidia GTX1060.

If you want to learn more, subscribe for our lab course on deep learning and/or our Seminar/Blockseminar!

Extensions: Cell Tracking over Time