

# Convolutional neuronal networks

Till Korten

With Material from

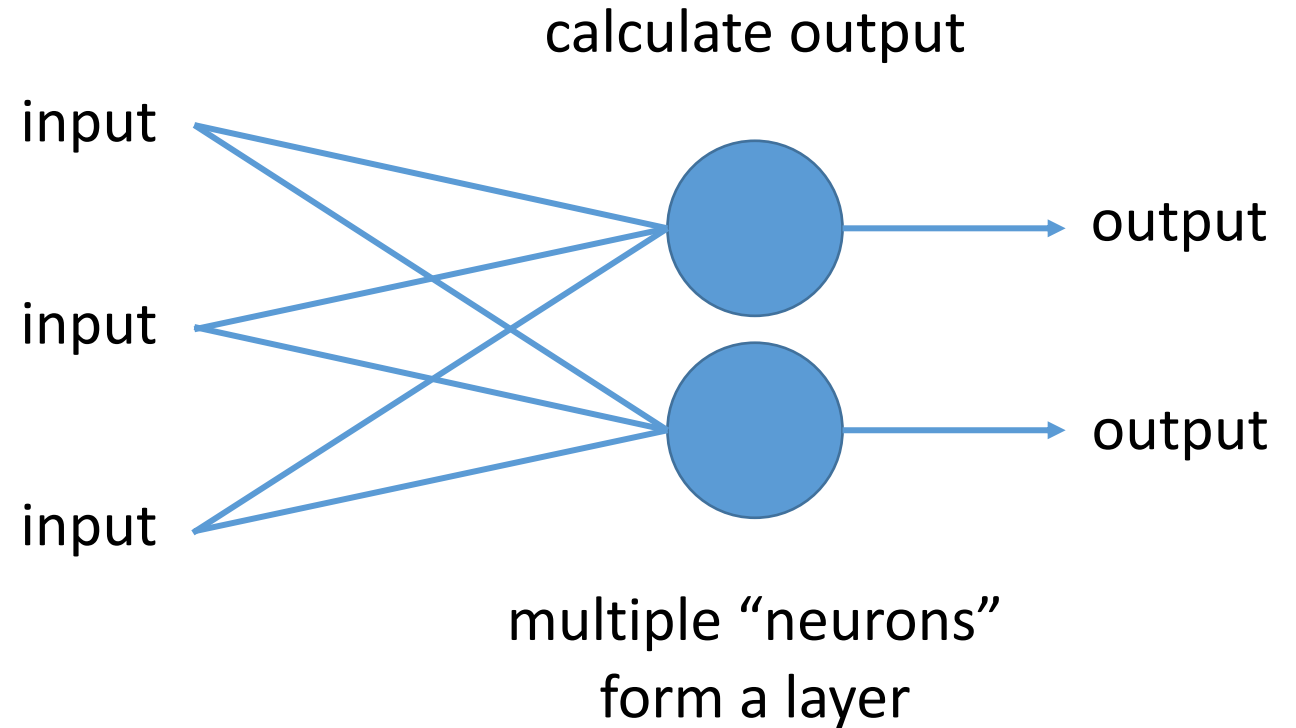
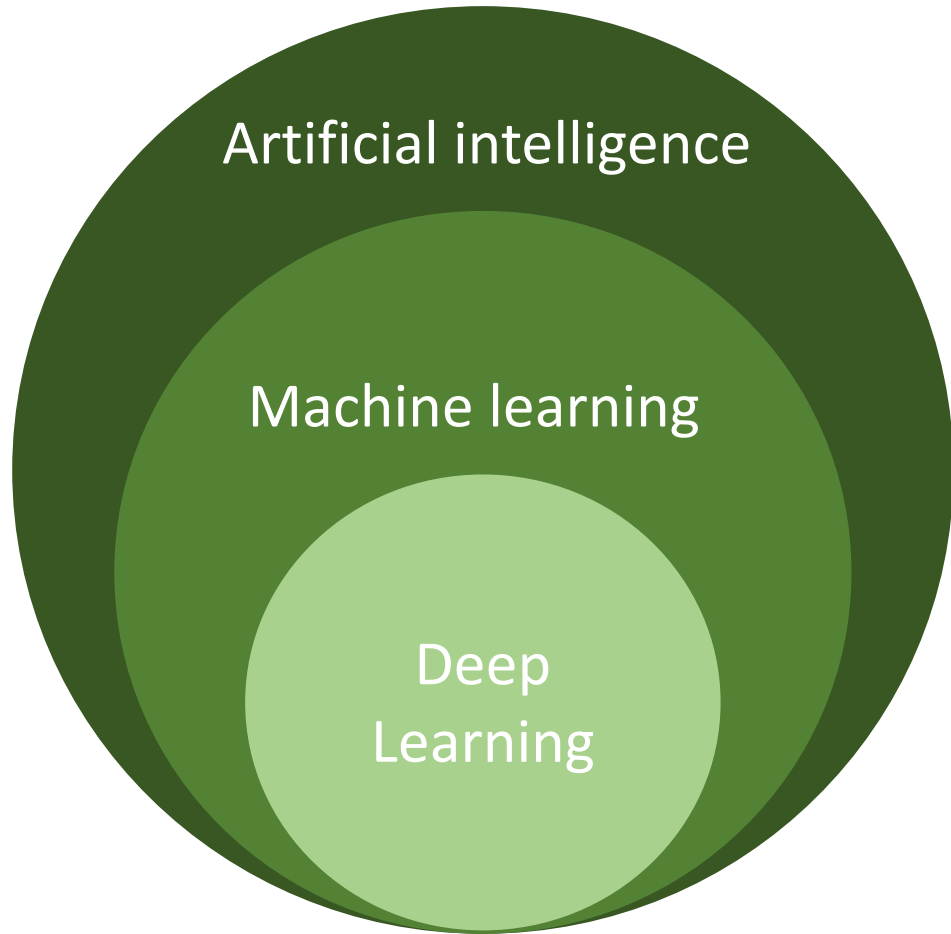
Johannes Müller, Robert Haase: PoL

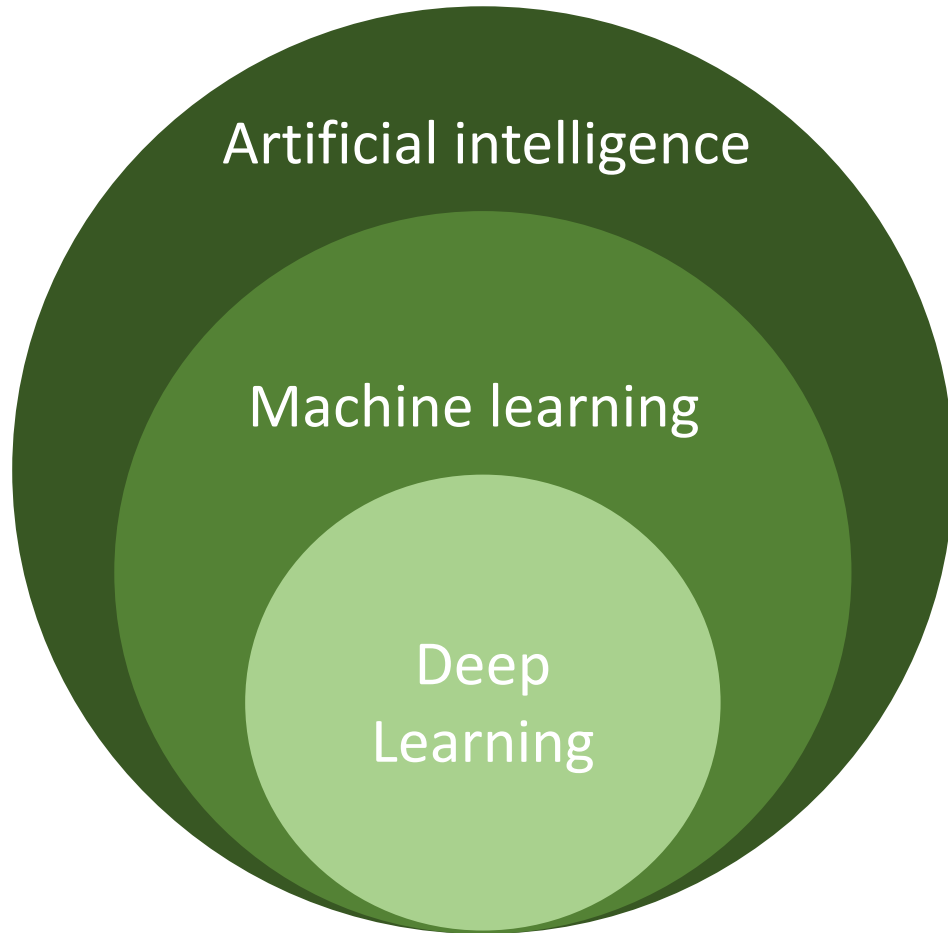
Alex Krull, Uwe Schmidt: MPI CBG

Martin Weigert: EPFL Lausanne

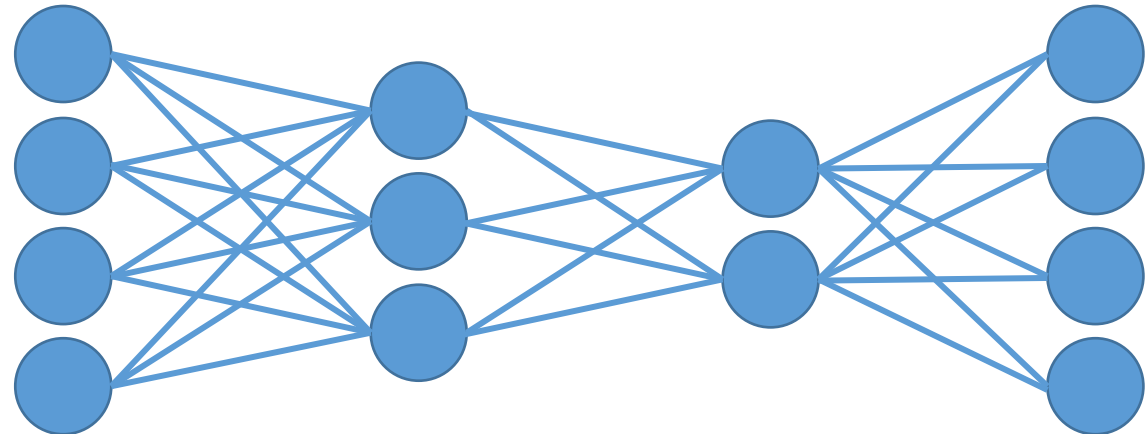
Ignacio Arganda-Carreras: Universidad del Pais Vasco

- Neural networks are composed of individual artificial “neurons”

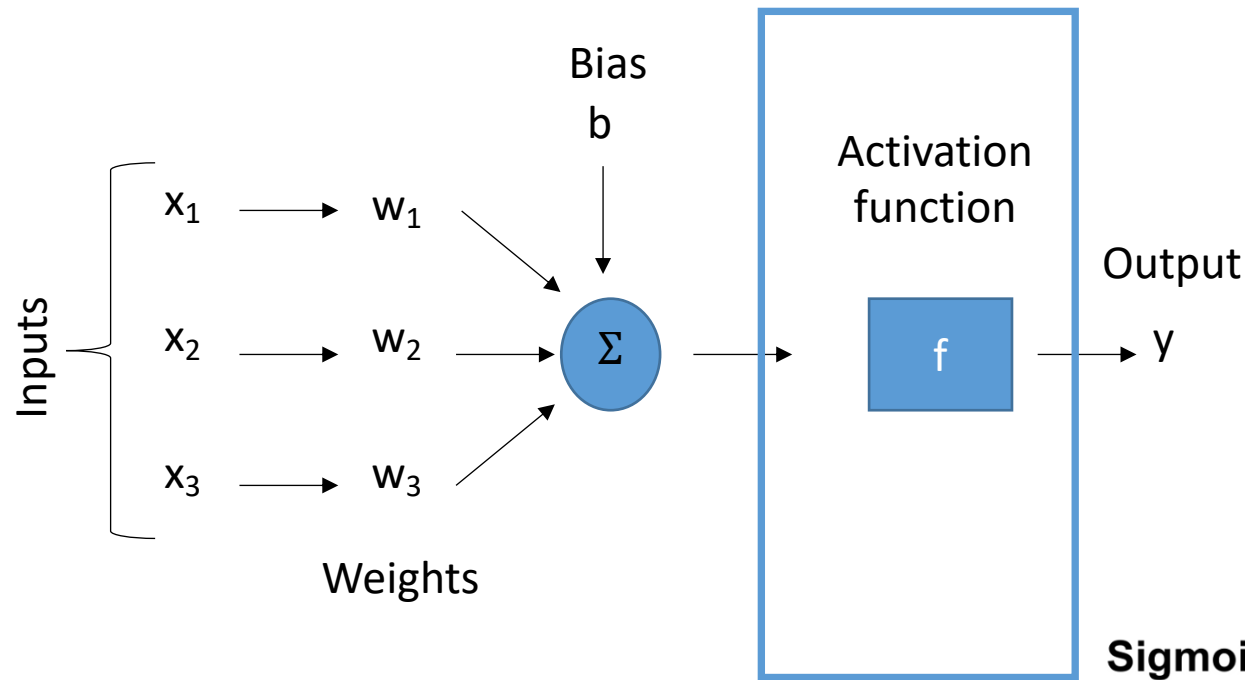




- Neural network: multiple layers of artificial “neurons”
- Deep neural network (DNN): neural network with more than one layer between input and output



# An artificial “neuron” is a mathematical function



Single neuron output calculation

$$y = w_1x_1 + w_2x_2 + w_3x_3 + b = w^T x + b$$

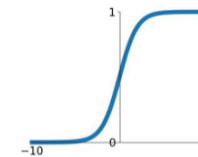
For image data, the values  $x_1, x_2, \dots$  would be

Pixel intensities

Pixel coordinates

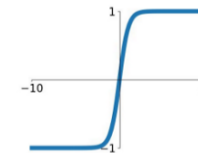
**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



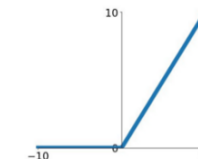
**tanh**

$$\tanh(x)$$



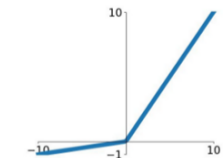
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$

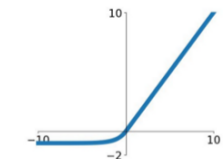


**Maxout**

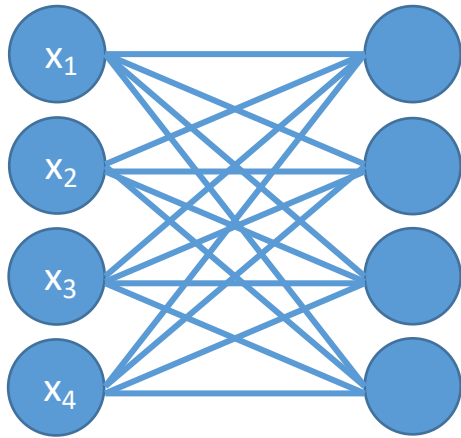
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

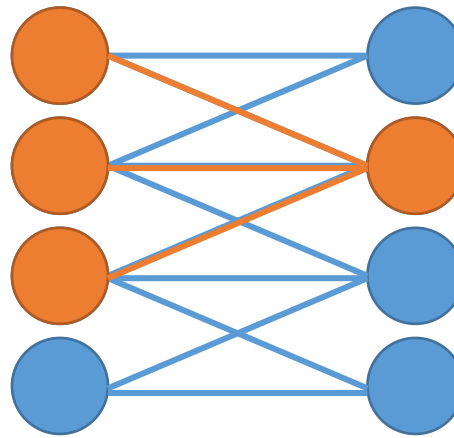
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



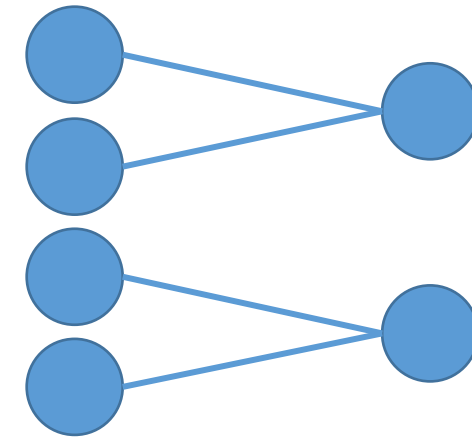
# Network layers can have different architectures



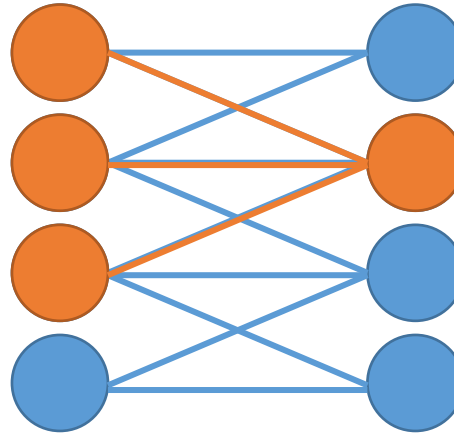
Fully connected layer



Convolutional layer



Pooling layer

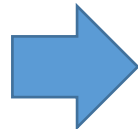


Convolutional layer

**Previously:**

Defined filter kernels

|      |     |      |
|------|-----|------|
| 1/16 | 1/8 | 1/16 |
| 1/8  | 1/4 | 1/8  |
| 1/16 | 1/8 | 1/16 |

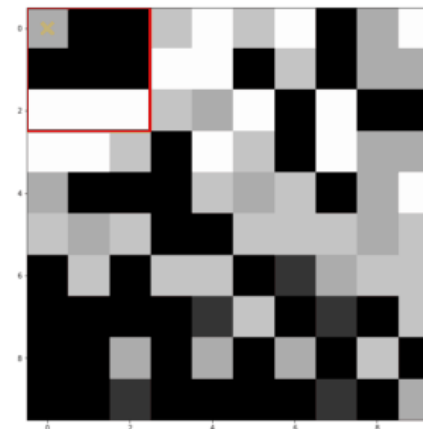


**Now:**

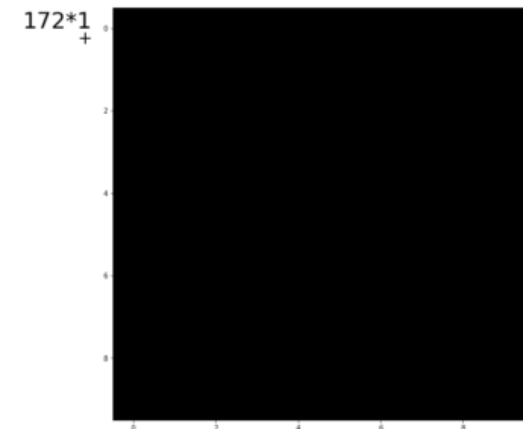
Learned filter kernels

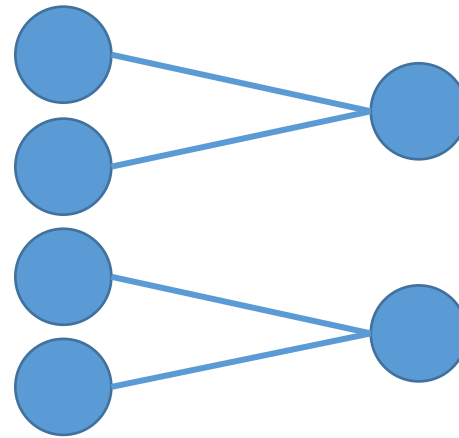
|          |          |          |
|----------|----------|----------|
| $w_{11}$ | $w_{12}$ | $w_{13}$ |
| $w_{21}$ | $w_{22}$ | $w_{23}$ |
| $w_{31}$ | $w_{32}$ | $w_{33}$ |

**Input**



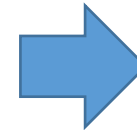
**Output**





Pooling layer

|    |    |   |    |
|----|----|---|----|
| 3  | 15 | 1 | 13 |
| 9  | 7  | 0 | 10 |
| 11 | 5  | 5 | 3  |
| 1  | 8  | 9 | 6  |



Maximal values

|    |    |
|----|----|
| 15 | 13 |
| 11 | 9  |

Average values

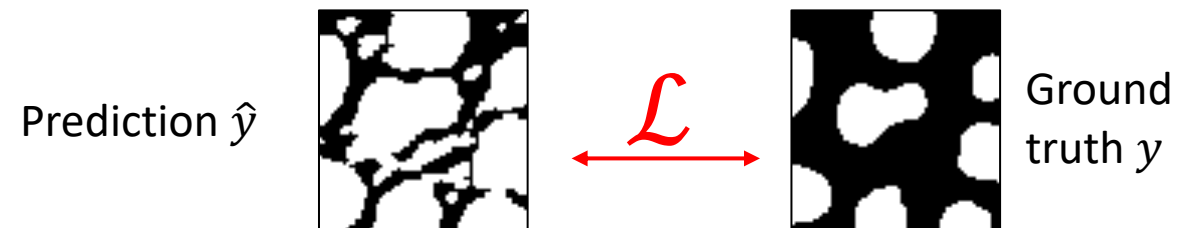
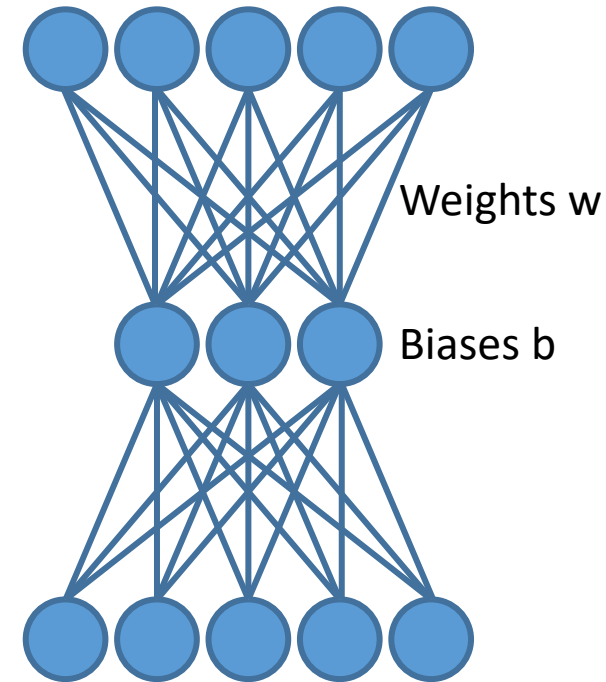
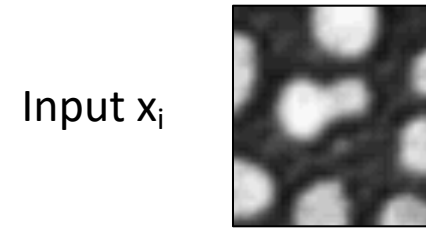
|     |     |
|-----|-----|
| 8.5 | 6.0 |
| 6.3 | 5.8 |

# The network learns by minimizing a loss function

- Learning is an optimization problem
- Step 0: Initialize the network randomly
  - Weights
  - Bias
- Step 1: Forward pass the input through the network, get an initial prediction (Images 0...M)
- Step 2: Compare the output with the ground truth, compute the error (loss function)
  - The **loss function** can be freely defined.
  - Mean squared error:

$$\mathcal{L}(y, \hat{y}) = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - y_i)^2$$

- Step 3: Update weights





The loss function can be expanded from

$$\mathcal{L}(y, \hat{y}) = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - y_i)^2$$

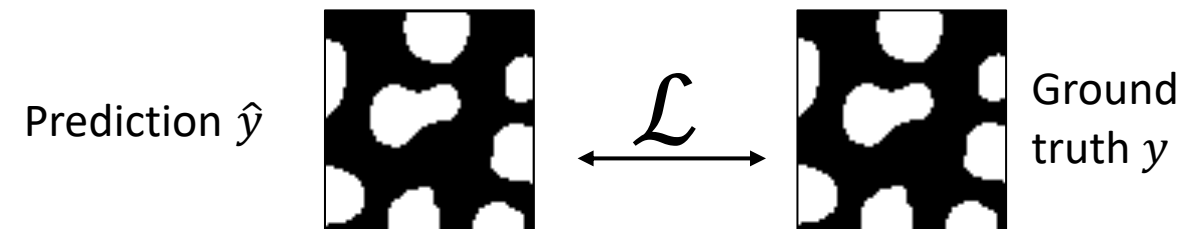
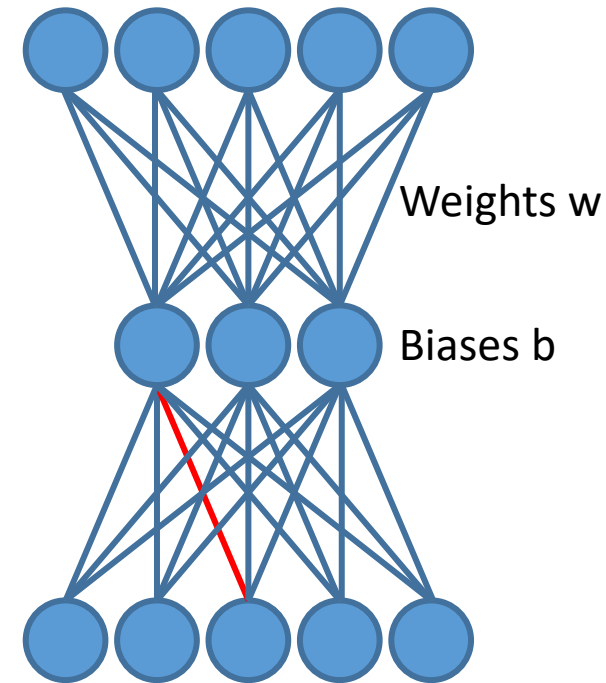
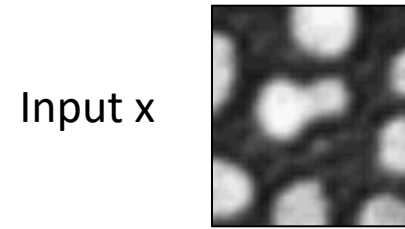
as the prediction depends on inputs  $x$  weights  $w$  and bias  $b$

$$\mathcal{L}(\hat{y}, x, w) = \frac{1}{M} \sum_{i=1}^M ((w^T x_i + b) - y_i)^2$$

We can calculate derivatives with respect to  $w$  and  $b$  to find their optimal values

→ Derivatives tell us how to change  $w$  &  $b$  in order to improve the prediction (i.e. minimize the loss function)

Repeat this for each layer, update weights  $w$



## **Popular frameworks:**

<https://www.tensorflow.org/>

<https://www.pytorch.org/>

**Hardware requirements:** Nvidia (CUDA-capable) graphics card (GPU)

**Memory:** The more GPU memory the better

## Before you start training:

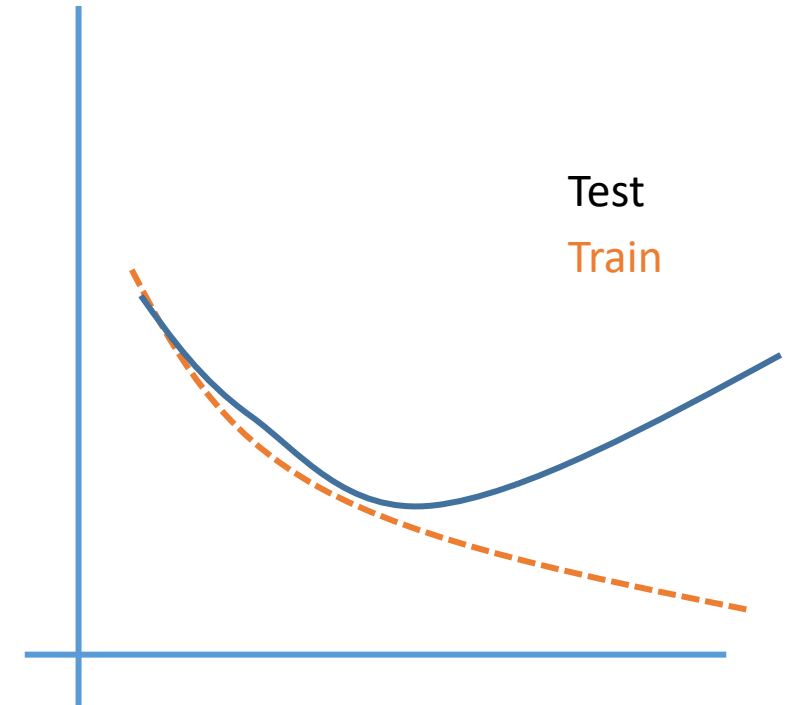
- Split data into three groups:
  - Training set
  - Testing set
  - Validation set

## During training:

- Apply network to training set:
  - Measure performance (“loss”) of network, update weights
- Apply network to testing set:
  - Measure performance of updated network, keep weights unchanged
- Improve network architecture (optimize “hyperparameters”)

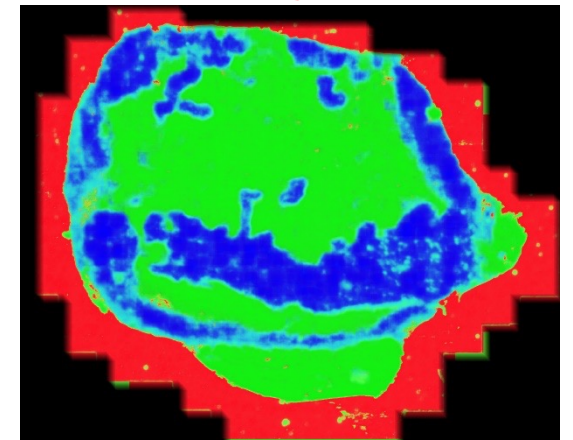
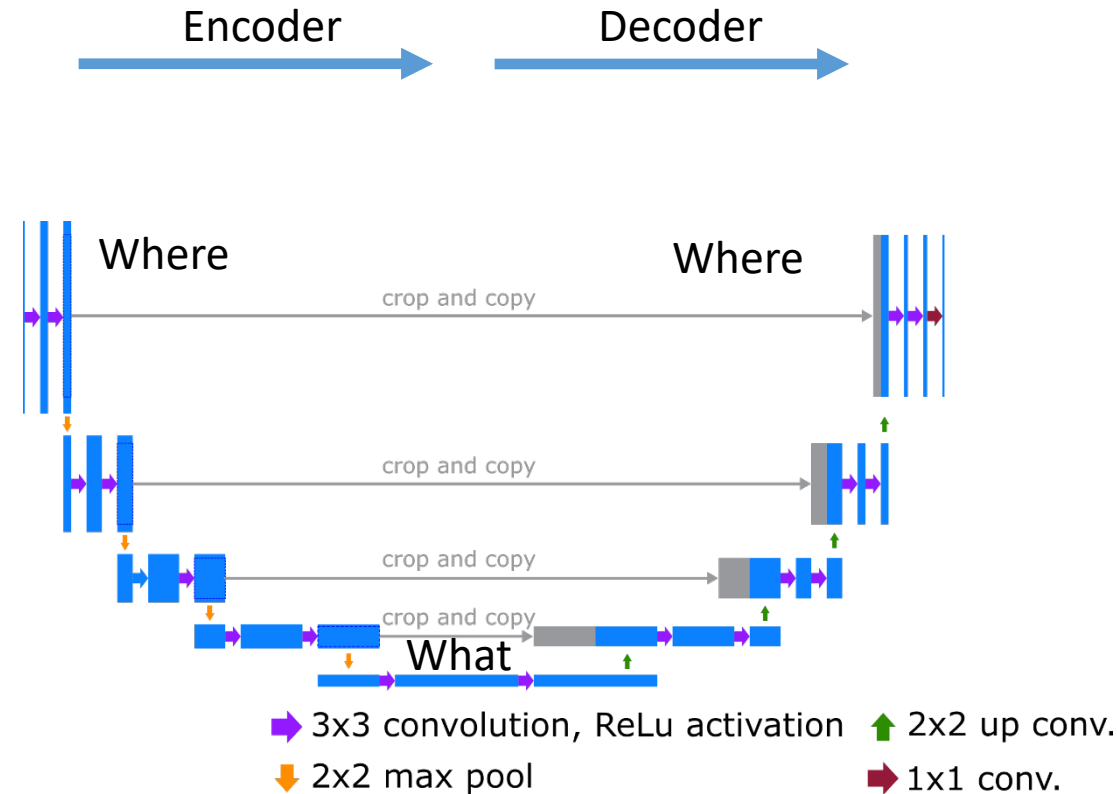
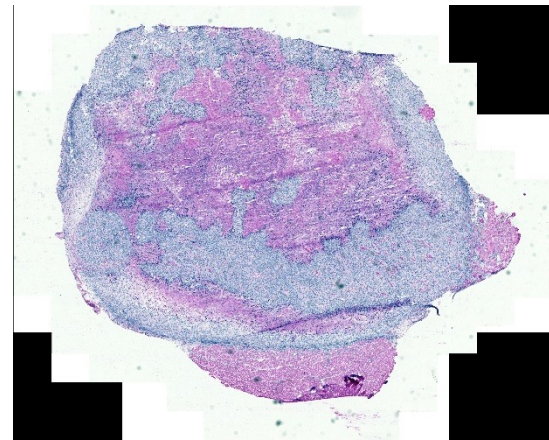
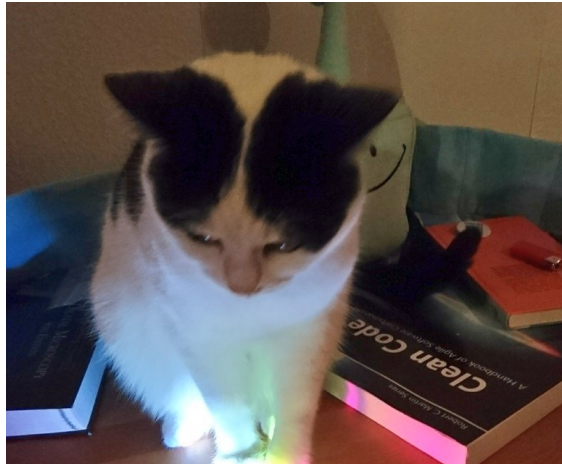
## During validation:

- Measure performance in unseen validation dataset



### Overfitting:

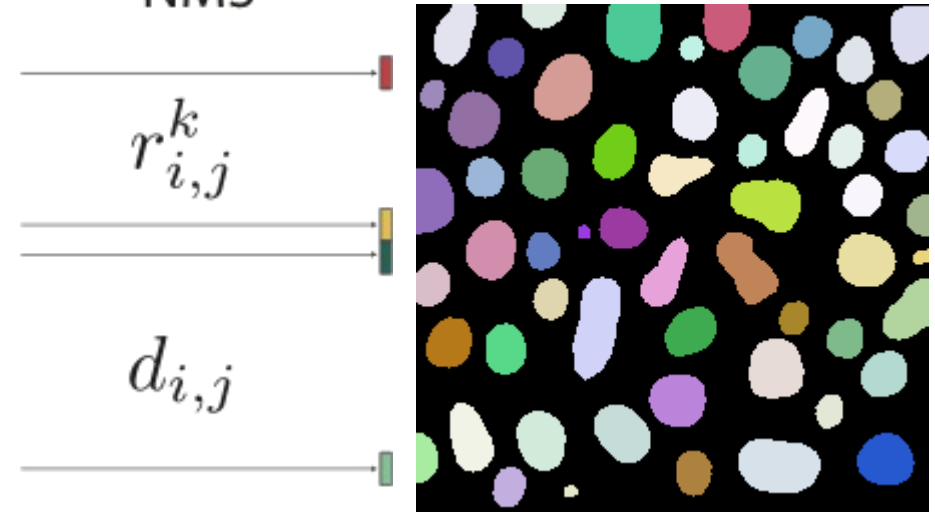
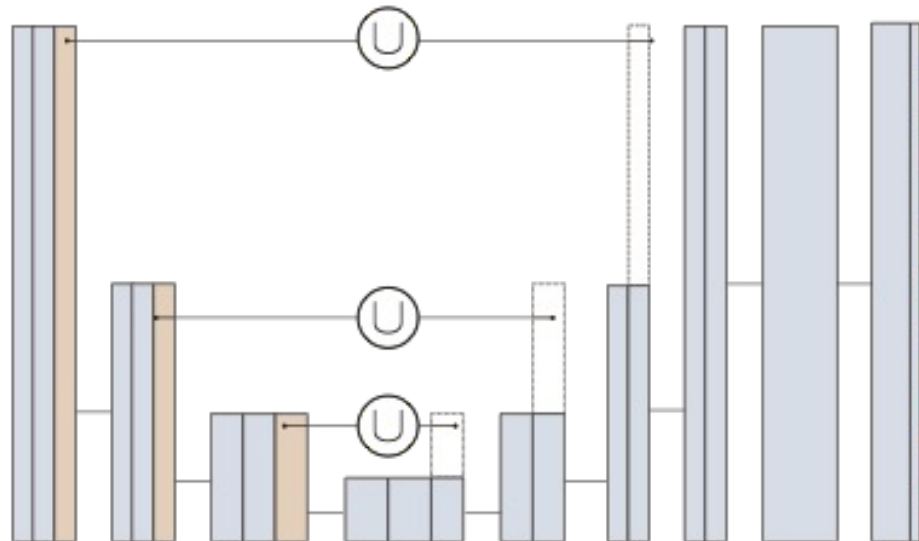
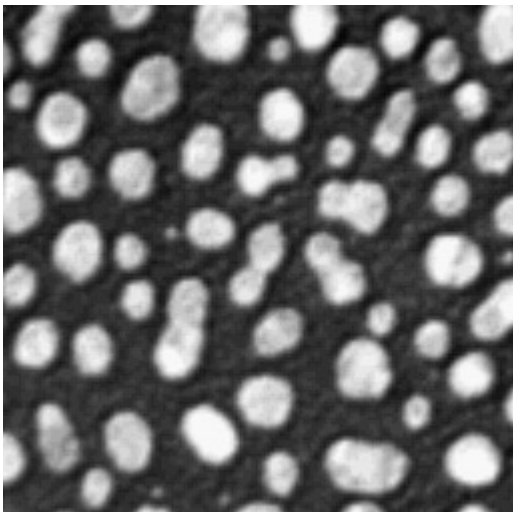
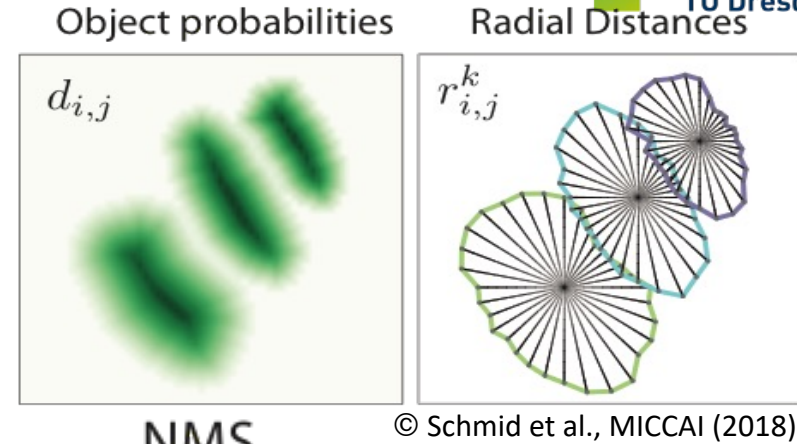
- Network is learning things „by heart“
- Hint at this happening: Updated weights from training fail to perform well in test



- The **U-net** is the most used network architecture in biological image processing using CNNs.
  - Encoder: Increase the “What”, decrease the “Where”
  - Decoder: Use the “What”, to identify the “Where”

## Strategy:

- Add additional information to prediction
- Member pixels of objects (nuclei) can be reached via a straight line from the center

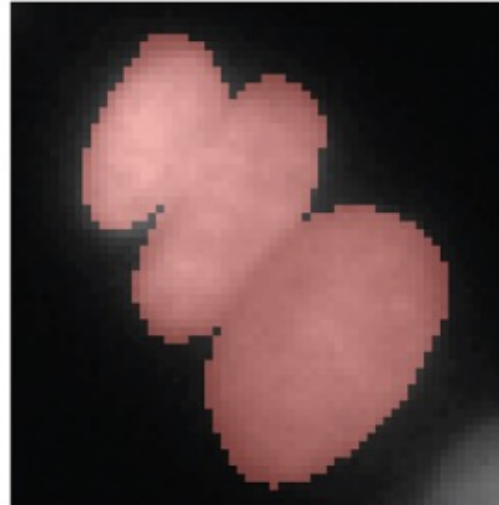
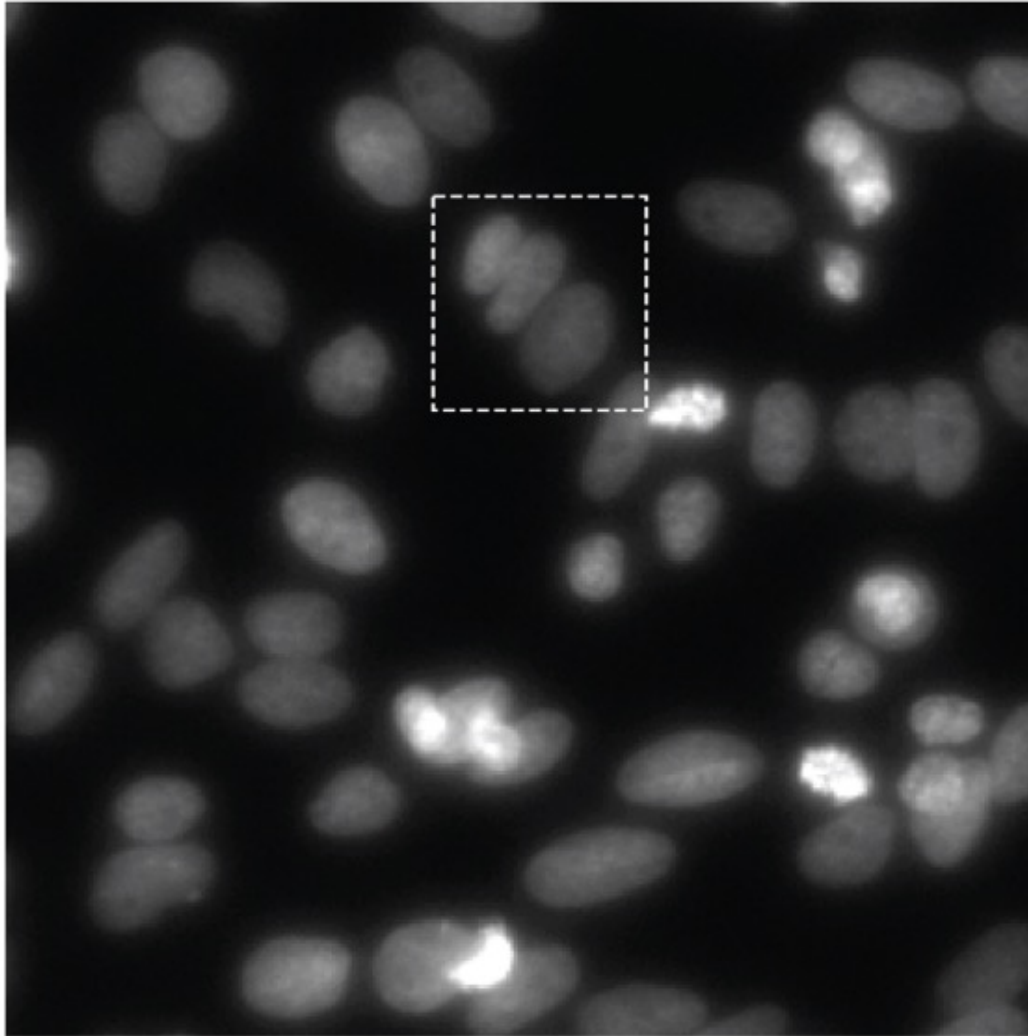


Dense Polygon Prediction  
(e.g. U-Net, ResNet)

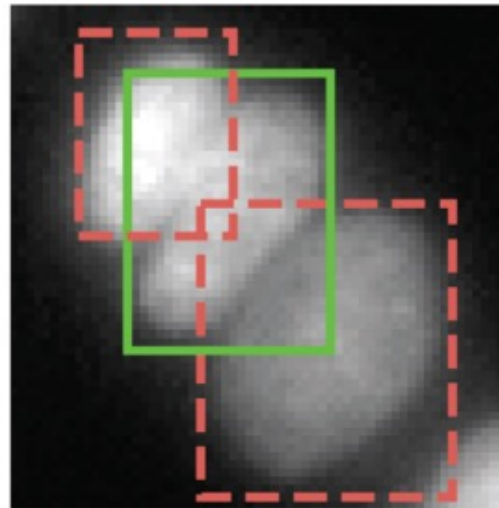
Polygon Selection  
(Non-Maximum Suppression NMS)



Noisy images + Crowded cells = Common source of segmentation errors



Dense Segmentation  
(e.g. U-Net)

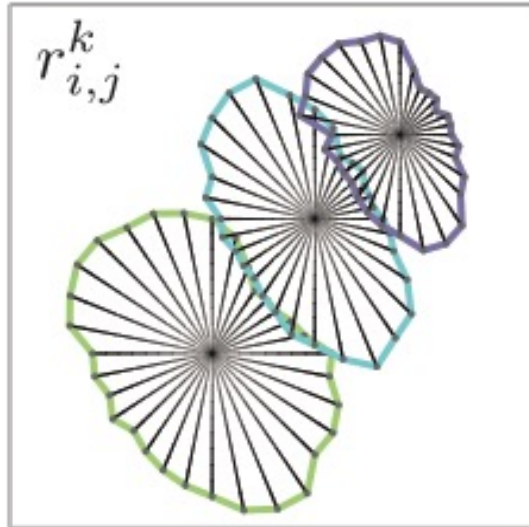


Bounding box based methods  
(e.g. Mask-RCNN)

Object probabilities



Radial Distances



© Schmid et al., MICCAI (2018)

## Problem:

- Multiple candidate points for nucleus center
- Overlapping instance predictions

Before NMS



After NMS

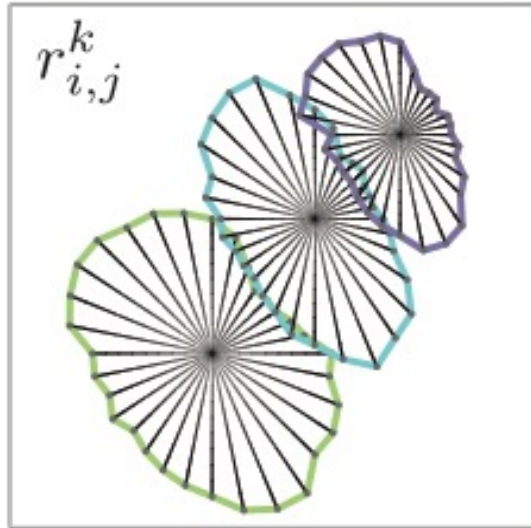
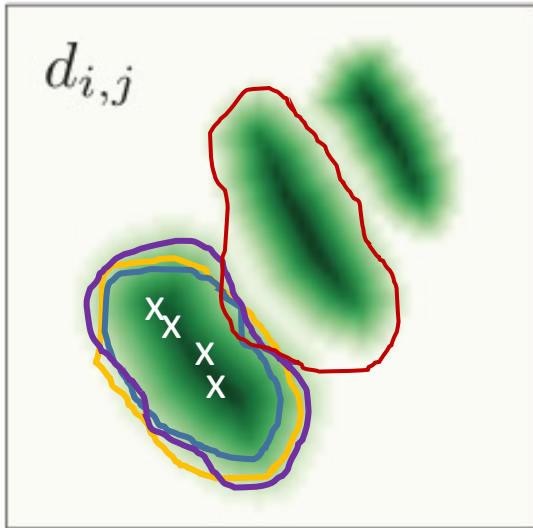


## Non-maximum-suppression (NMS):

- Intersection over Union (IoU) threshold  $\tau$  determines „conservativeness“:
  - High  $\tau$ : Objects tend to be considered as separate objects
  - Low  $\tau$ : Objects tend to be considered as the same objects

Object probabilities

Radial Distances











© Schmid et al., MICCAI (2018)

## Non-maximum-suppression (NMS):

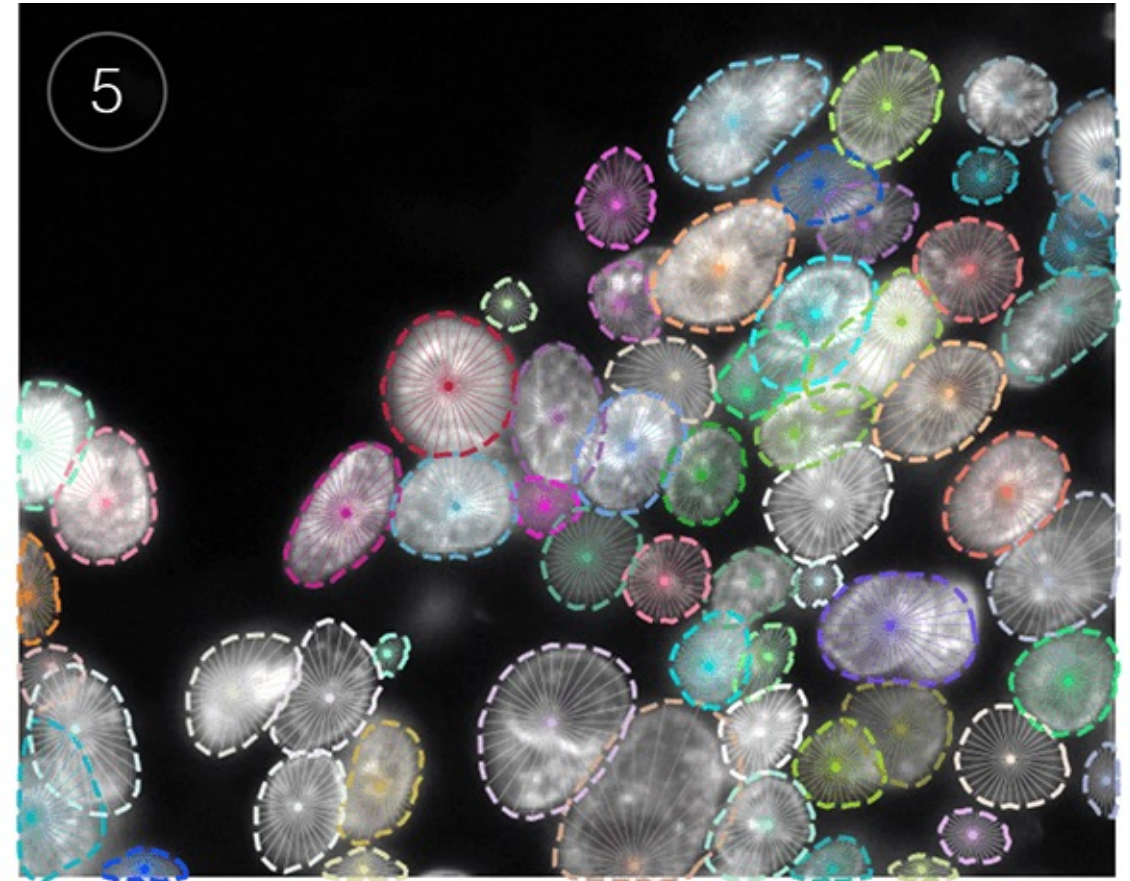
- Object probabilities: Probability that pixel belongs to class “nucleus”
- Multiple maxima lead to multiple possible polygons for the same nucleus

## Algorithm:

- Select polygon with highest object probability inside: 
- Look at other polygons: Is the overlap of  with  larger than threshold  $\tau$ ?
  - Yes:  and  are actually the same object, drop 
  - No:  and  are separate nuclei
- Setting  $\tau$  very high leads to many false positives!



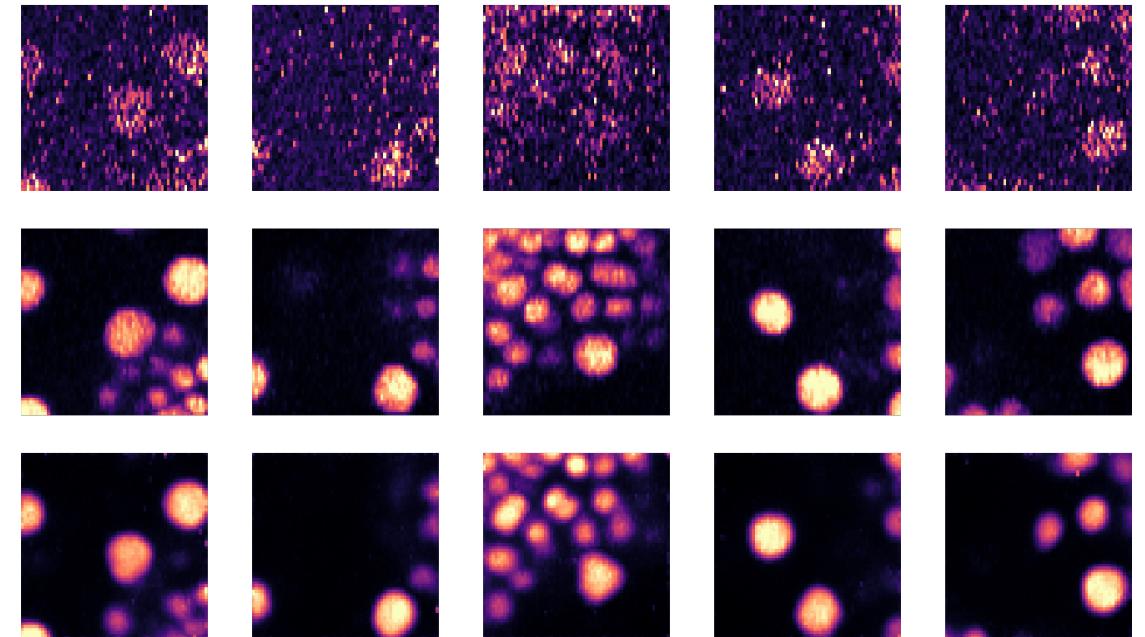
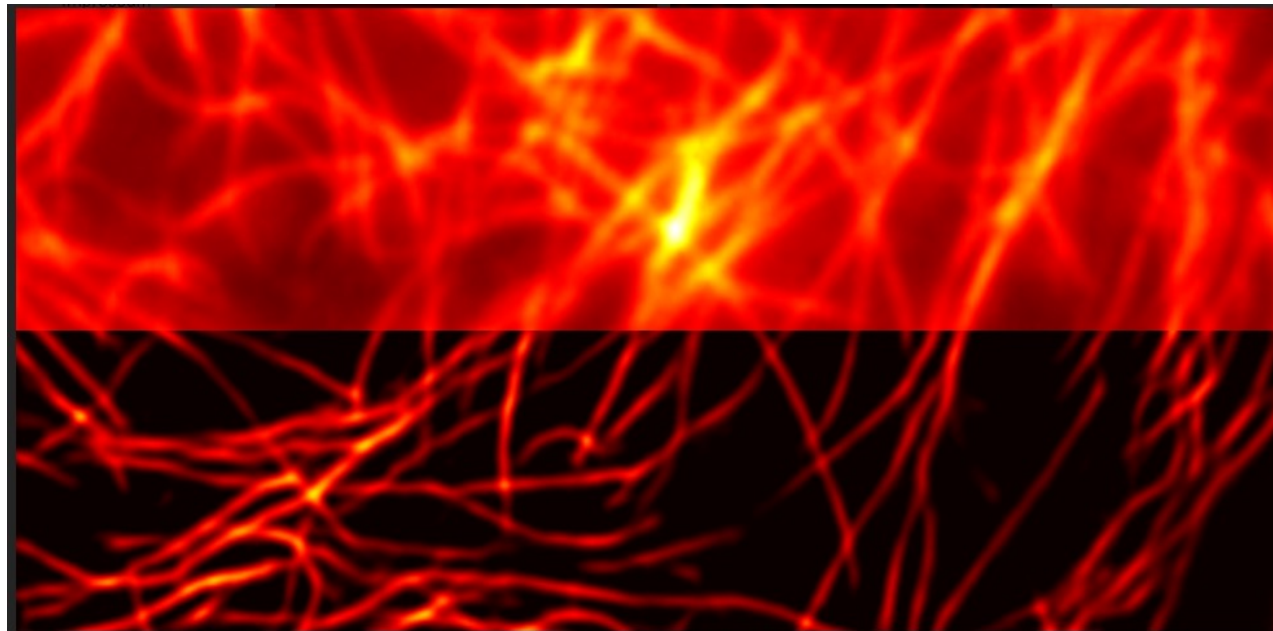
## Non-maximum suppression



© Schmid et al., MICCAI (2018)

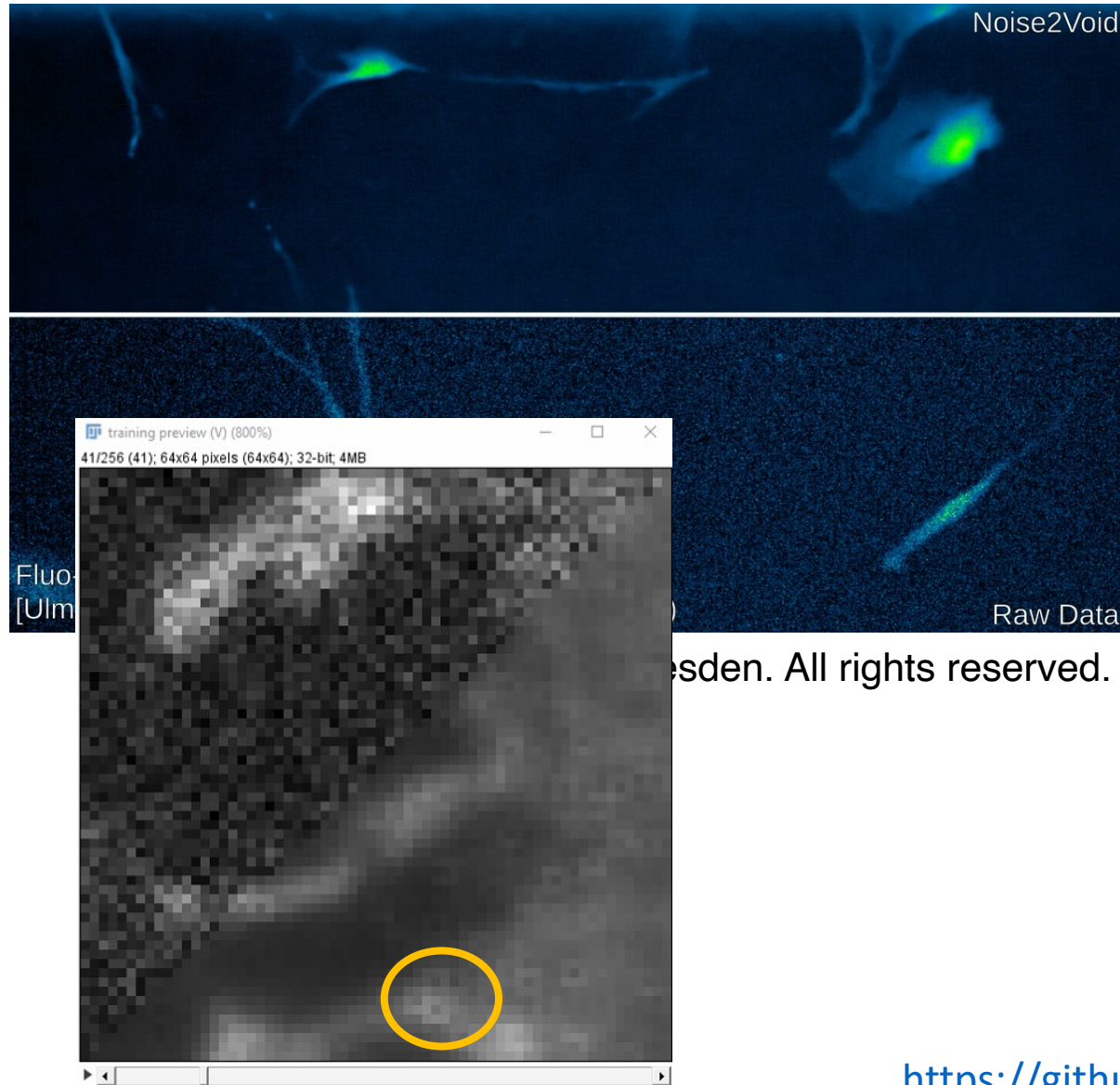
- **CARE**: content-aware restoration
- Image acquisition of pairs of images: A high-quality and a low-quality image.
- Caveats:
  - Reconstructs shot noise present in high quality training images
  - Trained model only applicable to image data of the same conditions (biological sample, microscope, etc)

5 example validation patches  
top row: input (source), middle row: target (ground truth), bottom row: predicted from source

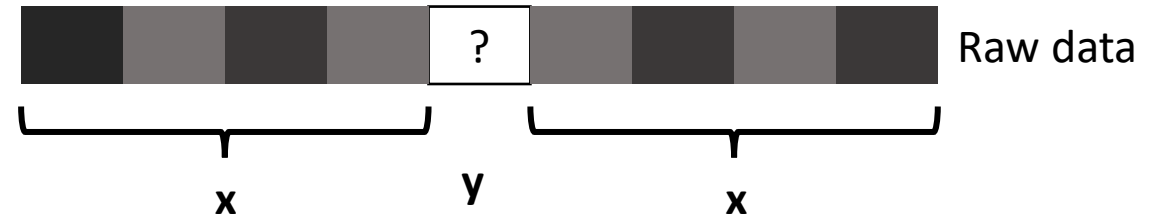


© MPI-CBG, Dresden. All rights reserved.





Raw data = signal + noise



## Strategy:

- Try to predict intensity of pixel  $y$  from surrounding pixels  $x$
- CNN fails to predict noise component → N2V can only reproduce signal from the surroundings of  $y$

## Beware:

- Only **random** noise can be removed, otherwise artifacts occur

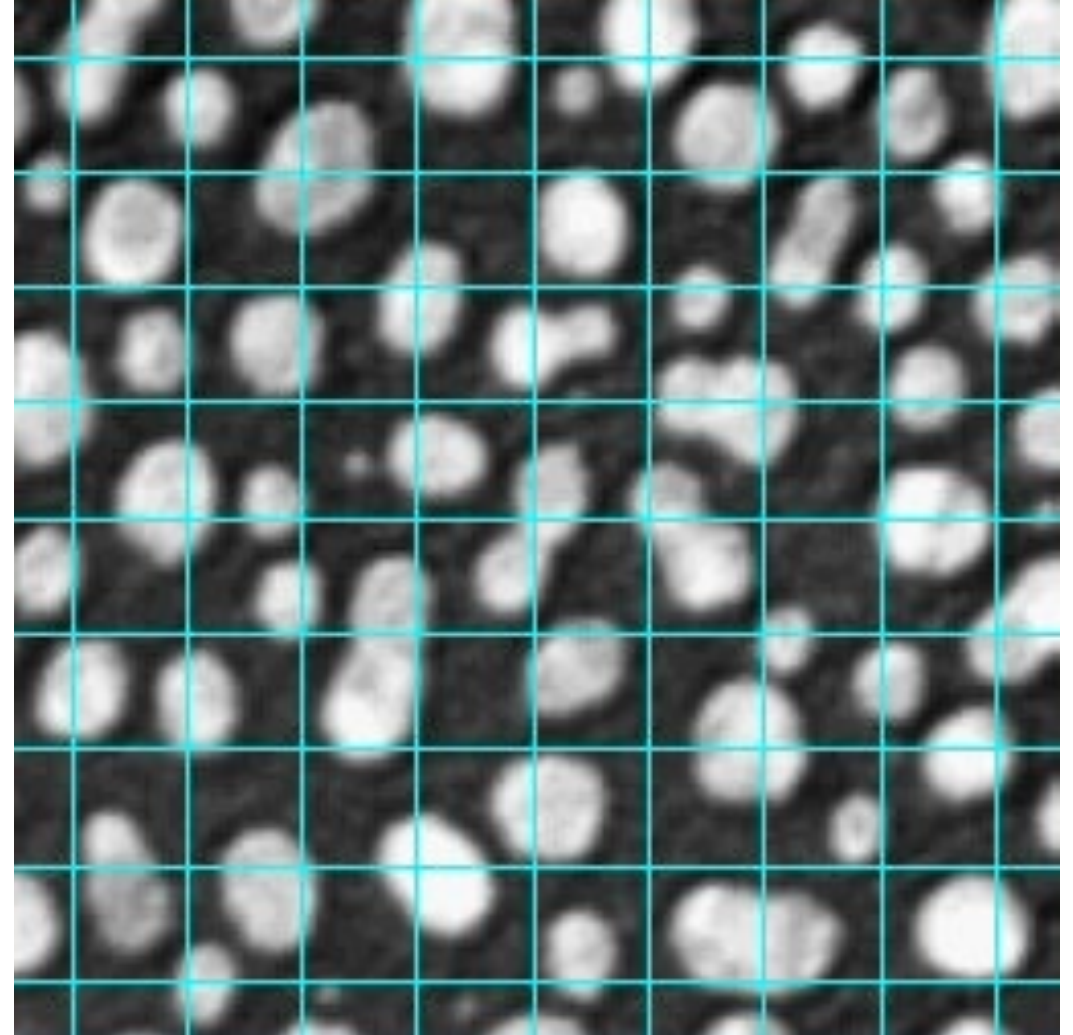
<https://github.com/juglab/n2v>

<https://forum.image.sc/t/n2v-artefacts-in-training-data/70686>

- Images are tiled
- limited “receptive field of the network”

## **Receptive field:**

- Objects must be smaller than receptive field to be detectable



## Unbalanced training data:

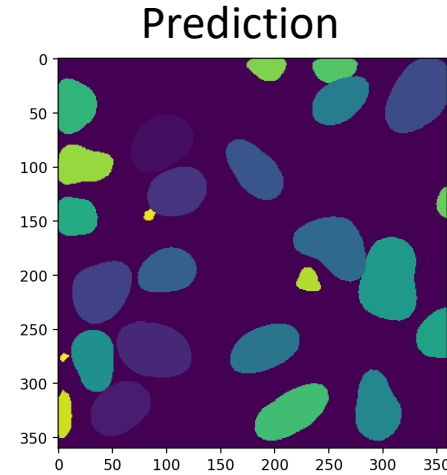
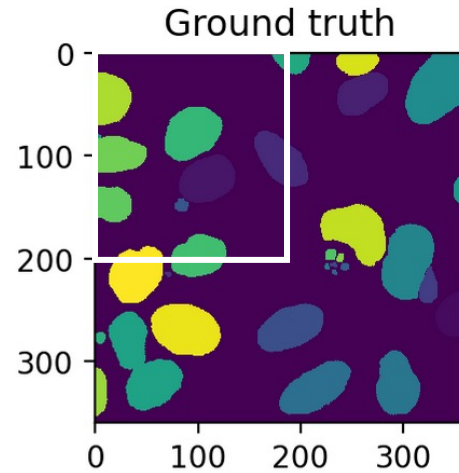
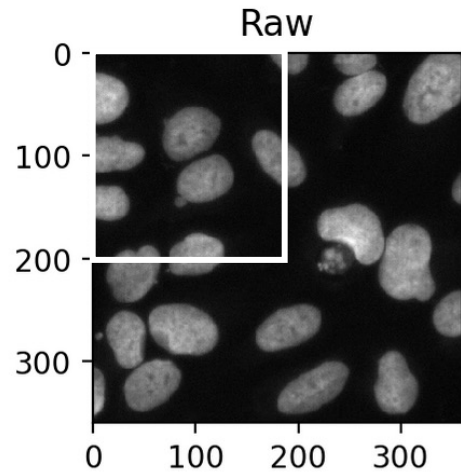
- Some labels appear more often in training data than others
- Rare events will not be learned because missing them doesn't harm accuracy much
- Weighted data sampling
- **Biased results!!**

### **Is the iPhone racist? Chinese users claim iPhoneX face recognition can't tell them apart**

APPLE has come under fire following numerous complaints from Chinese users who claim the iPhone X face recognition can't tell them apart.

<https://www.news.com.au/technology/gadgets/mobile-phones/is-the-iphone-racist-chinese-users-claim-iphonex-face-recognition-cant-tell-them-apart/news-story/13814540e8c82ad466aca687e12af64c>

# When the input data does not fit to the training data

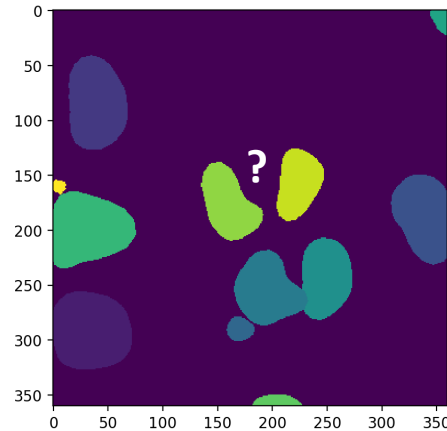
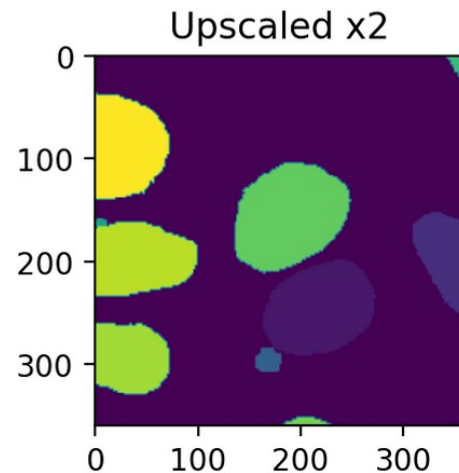
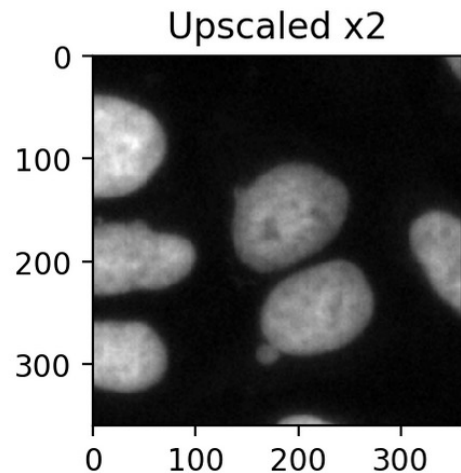


What happened here?

Receptive field too small

I used a different resolution than during training

Overfitting



- With great power comes great responsibility: **Validate your models well!**
- Better data more important than better model
- Often performs fantastic – *but you don't know why*
- Generative neural networks (like CARE) can dream up data – to a hammer everything looks like a nail!