

# Convolutional neuronal networks

Johannes Müller

With Material from

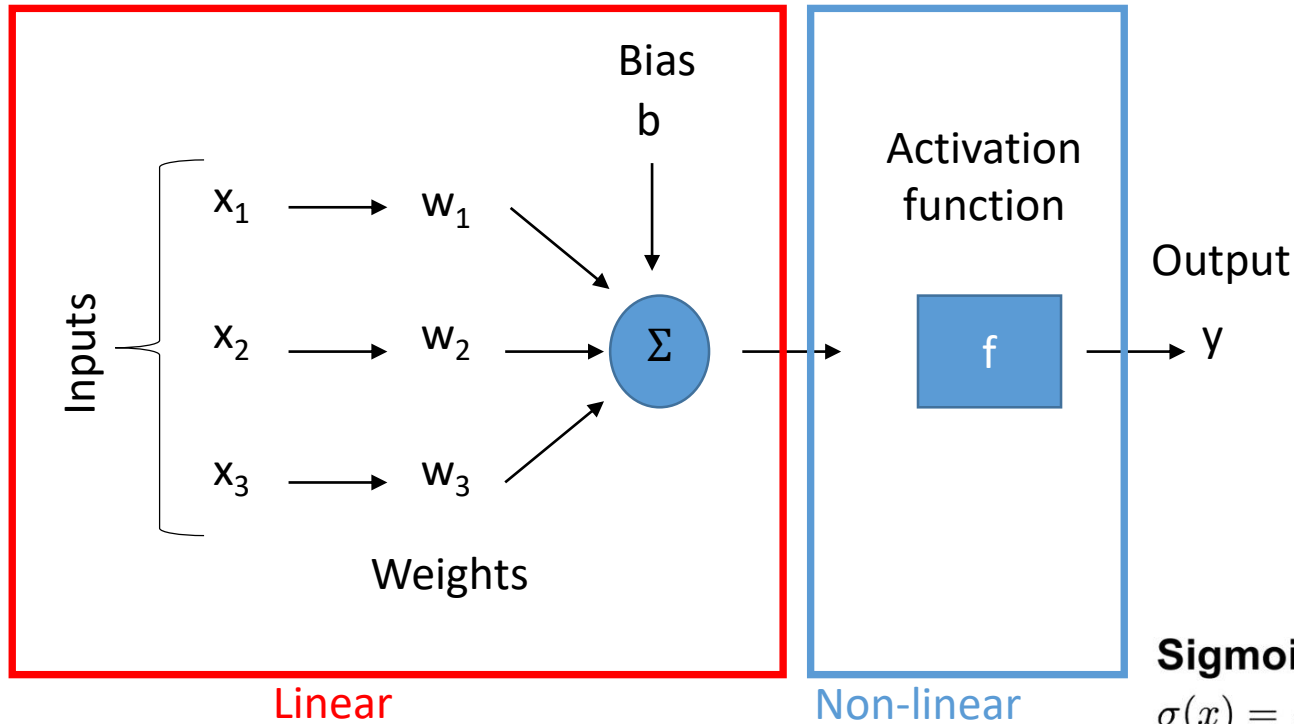
Robert Haase, PoL

Alex Krull, MPI CBG

Martin Weigert, EPFL Lausanne

Uwe Schmidt, MPI CBG

Ignacio Arganda-Carreras, Universidad del Pais Vasco



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

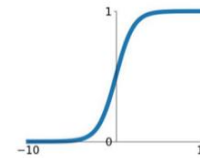
Pixel intensities

Pixel coordinates

Filter kernel entries

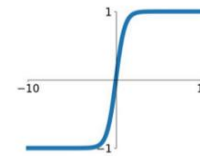
**Sigmoid**

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



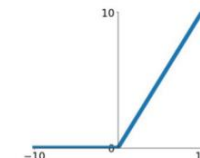
**tanh**

$$\tanh(x)$$



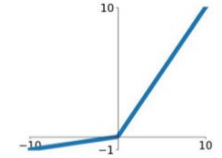
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

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

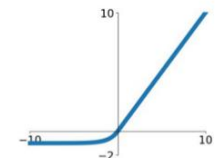


**Maxout**

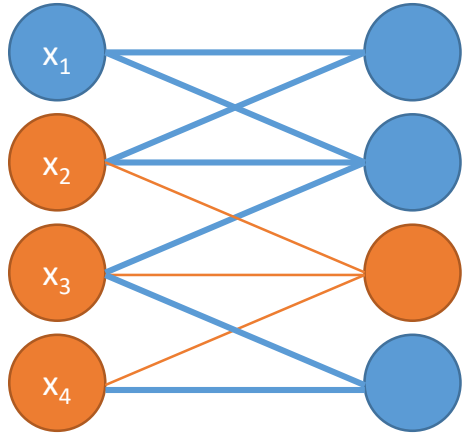
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

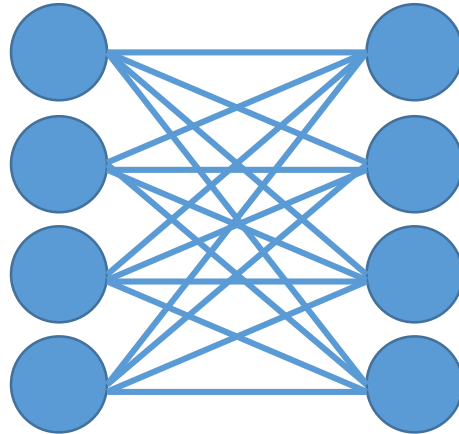
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



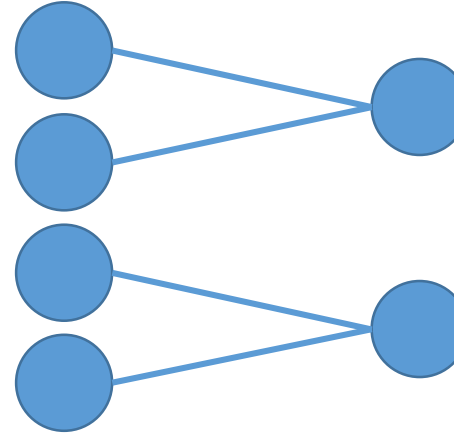
- Layers



Convolutional layer



Fully connected layer

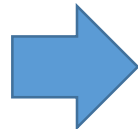


Pooling layer  
("Max pool", "Average pool") Pooling maximal values

**Previously:**

Defined filter kernels

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

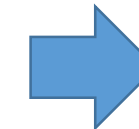


**Now:**

Undefined filter kernels

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

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



15	13
11	9

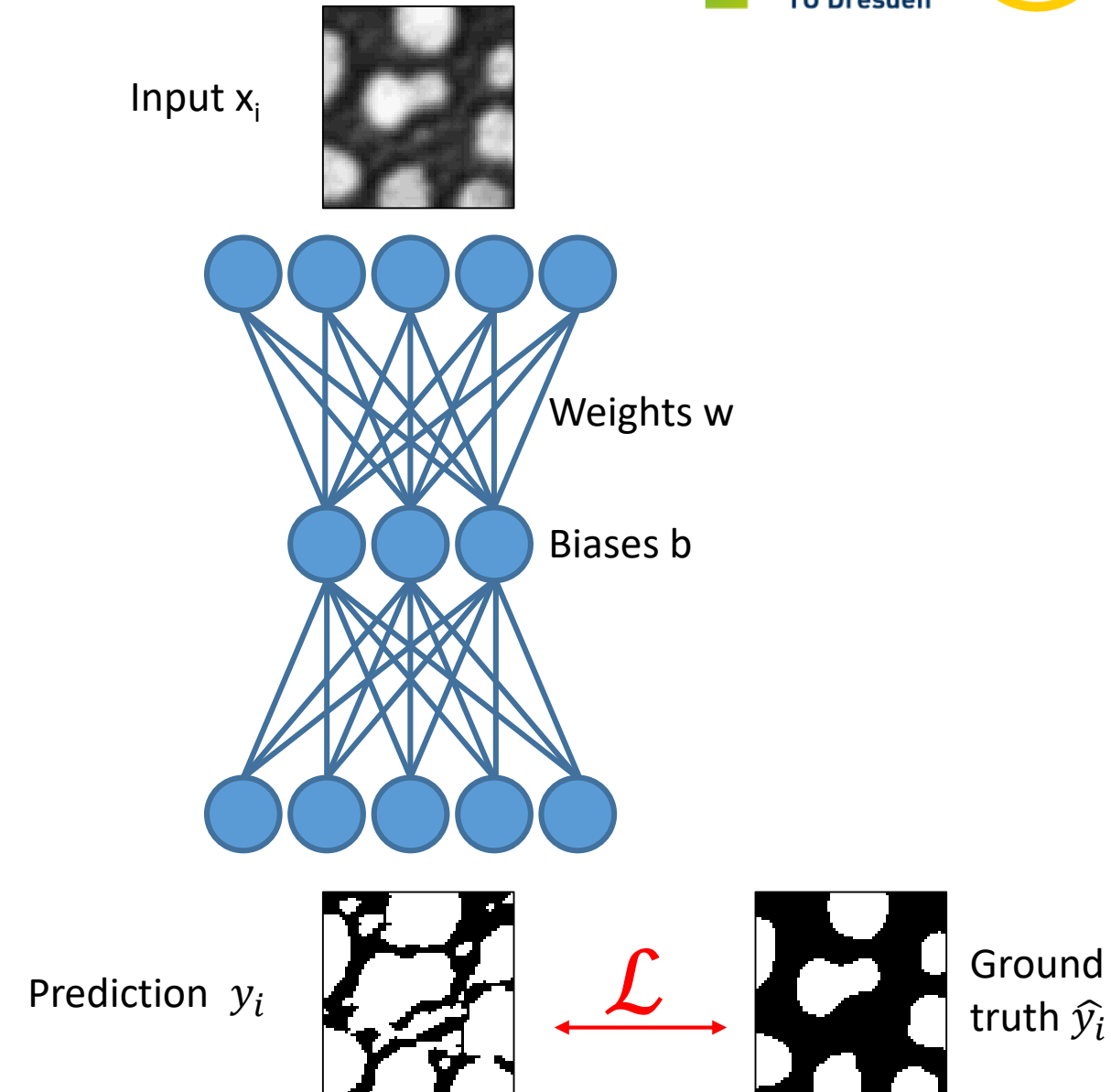
Averaging values

8.5	6.0
6.3	5.8

- 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

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

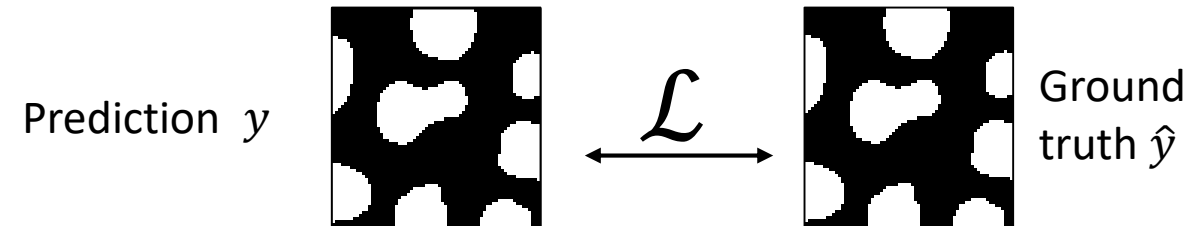
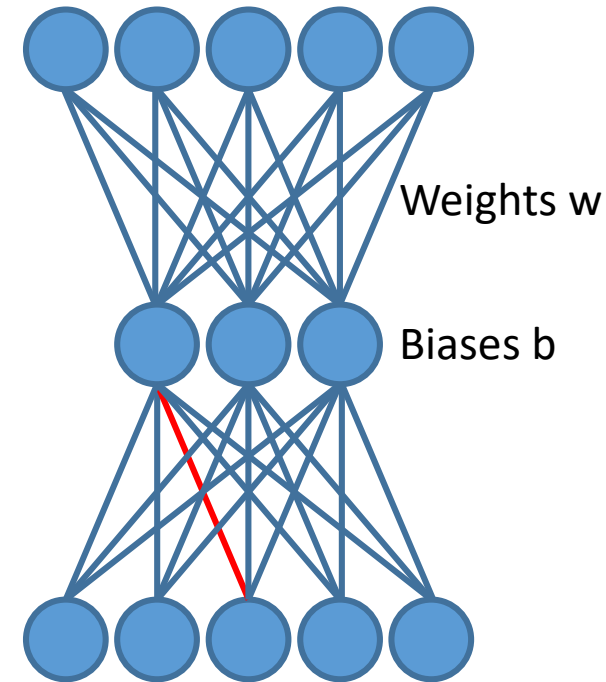
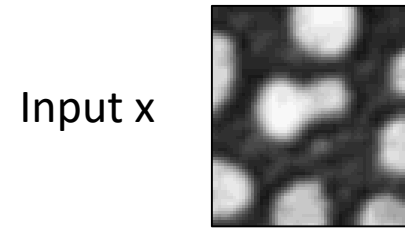
as the prediction depends on inputs  $x$  weights  $w$  and bias  $b$

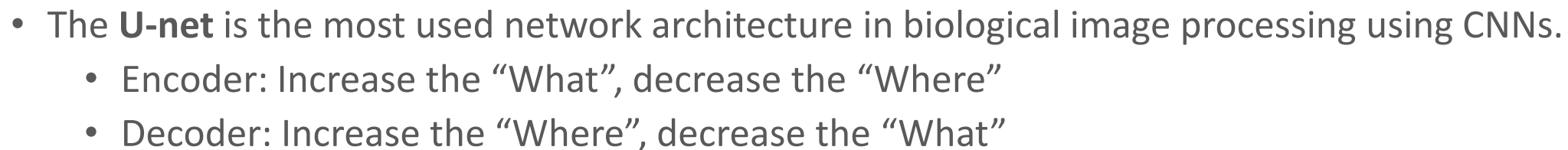
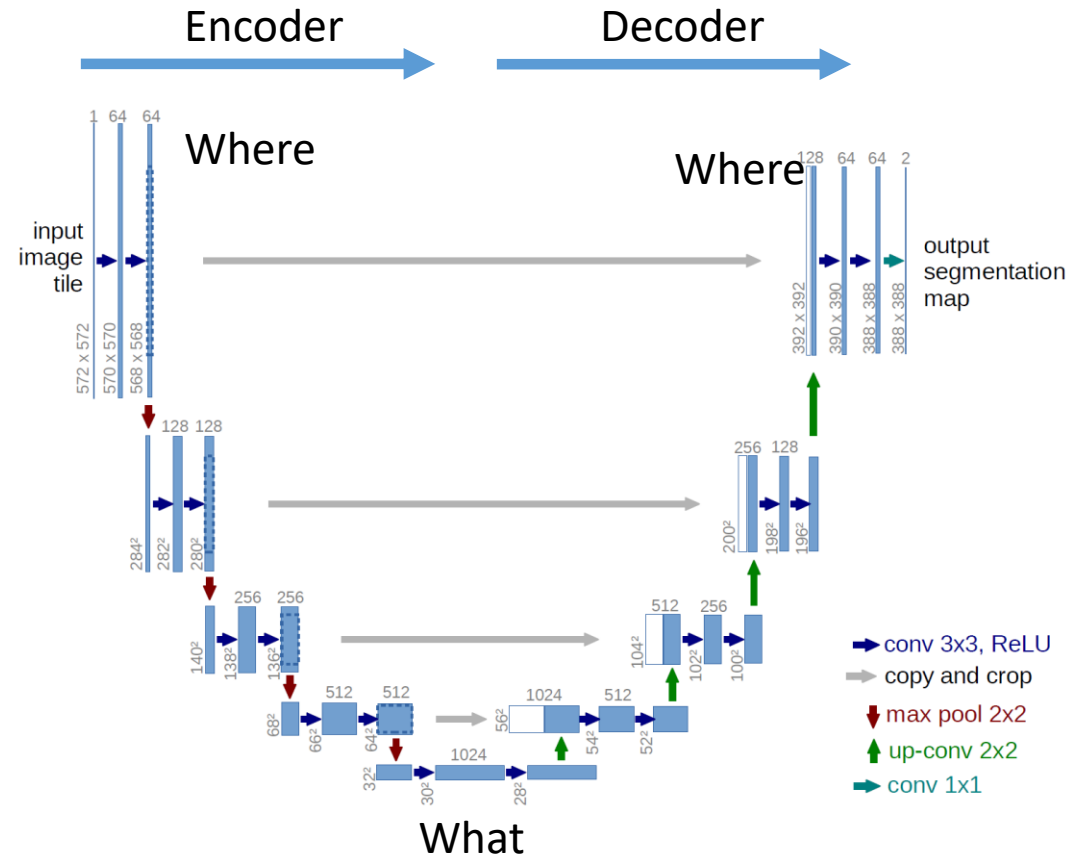
$$\mathcal{L}(\hat{y}, x, w) = \frac{1}{M} \sum_{i=1}^M (\hat{y}_i - (w^T x_i + b))^2$$

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

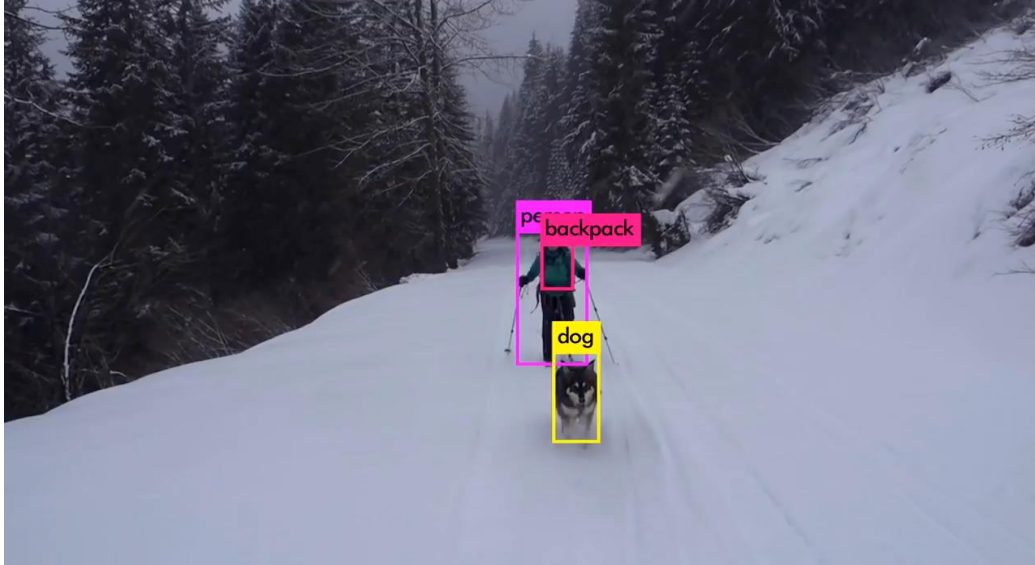
Repeat this  $n$  times, each time update weights  $w$







YOLO: You only look once



<https://github.com/ultralytics/yolov5>

## Deep learning can predict microsatellite instability directly from histology in gastrointestinal cancer

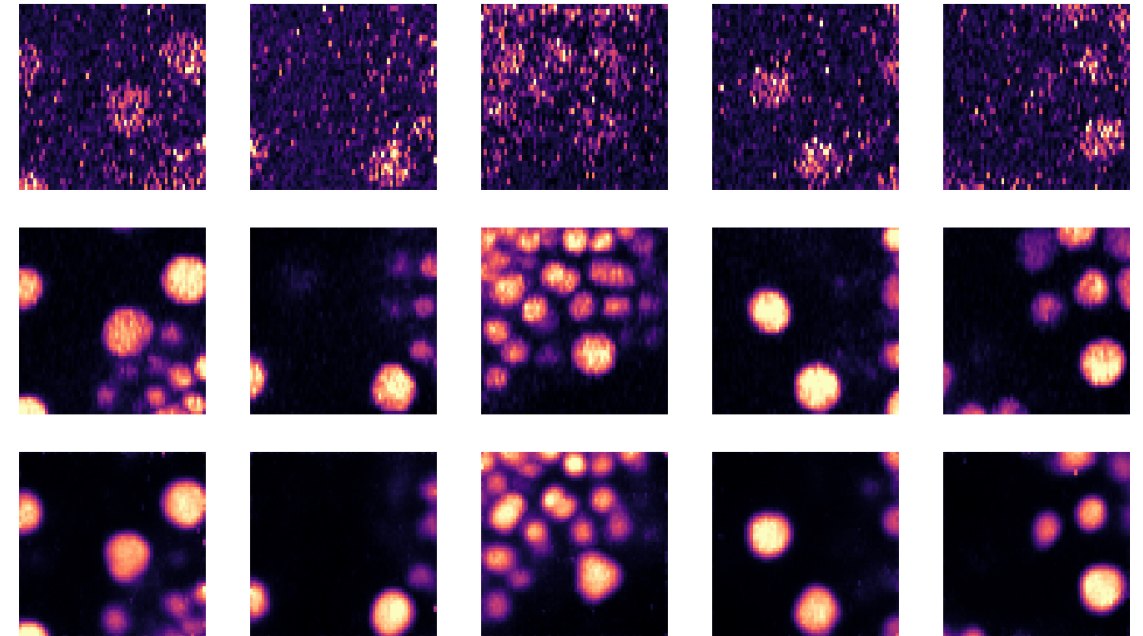
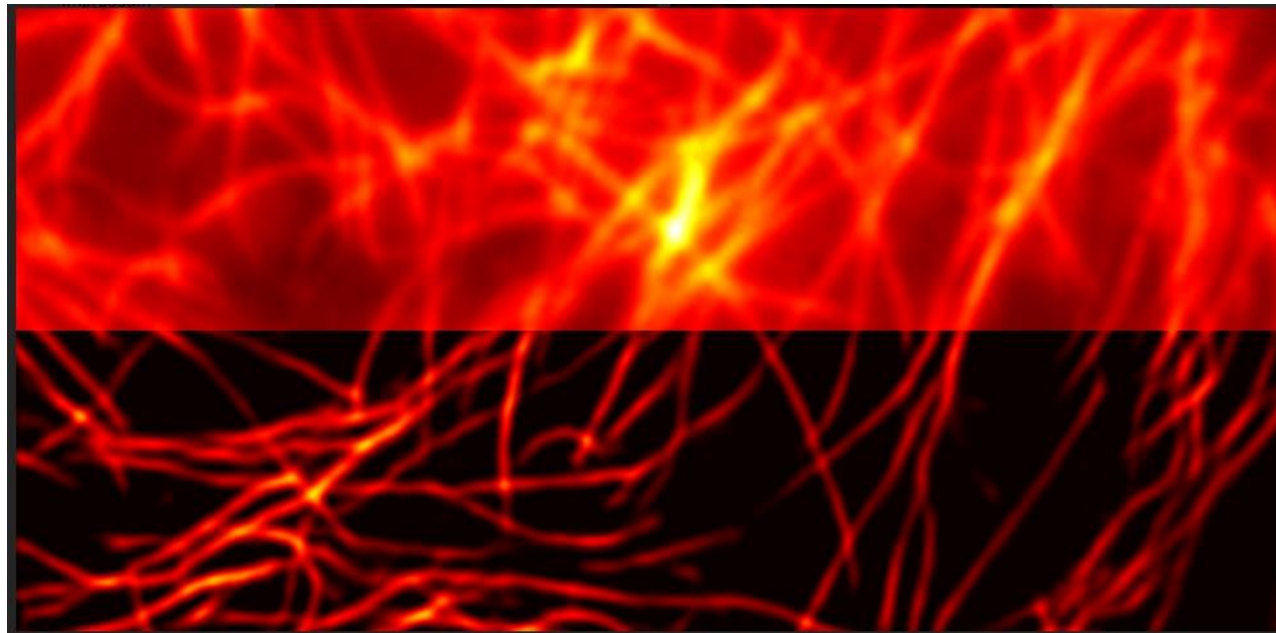
[Jakob Nikolas Kather](#) , [Alexander T. Pearson](#), [Niels Halama](#), [Dirk Jäger](#), [Jeremias Krause](#), [Sven H. Loosen](#), [Alexander Marx](#), [Peter Boor](#), [Frank Tacke](#), [Ulf Peter Neumann](#), [Heike I. Grabsch](#), [Takaki Yoshikawa](#), [Hermann Brenner](#), [Jenny Chang-Claude](#), [Michael Hoffmeister](#), [Christian Trautwein](#) & [Tom Luedde](#) 

*Nature Medicine* **25**, 1054–1056 (2019) | [Cite this article](#)

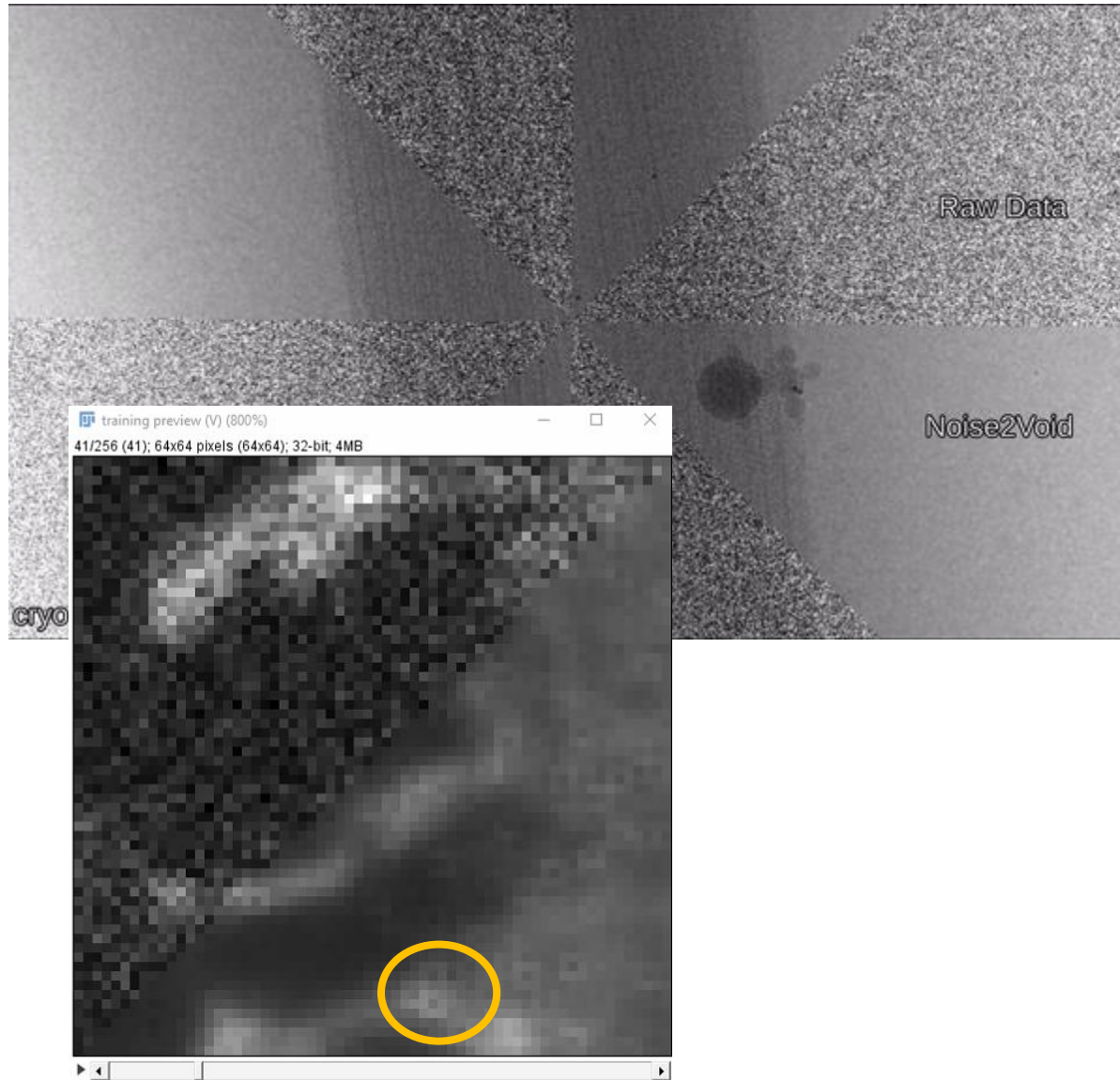
**32k** Accesses | **408** Citations | **263** Altmetric | [Metrics](#)

- **CARE:** content-aware restoration
- Image acquisition of pairs of images: A high-quality and a low-quality image.
- Problem: Shot noise, Biology moves!
- Trained model only applicable to image data of the same conditions (biological system, microscope, etc)

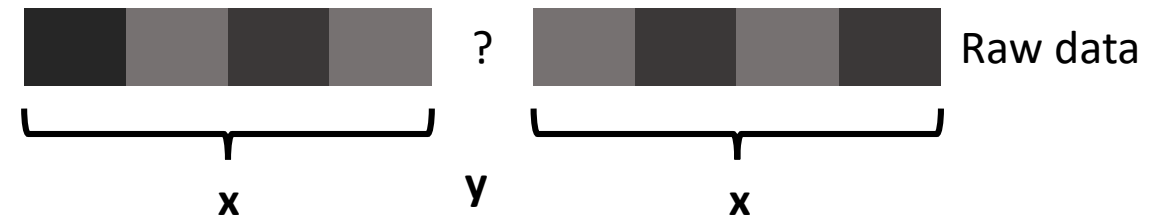
5 example validation patches  
top row: input (source), middle row: target (ground truth), bottom row: predicted from source







Raw data = signal + noise



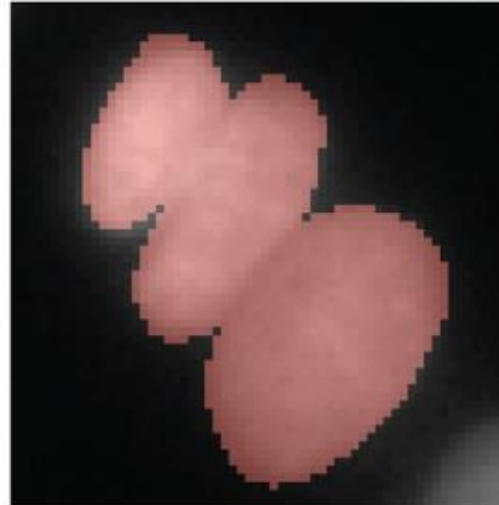
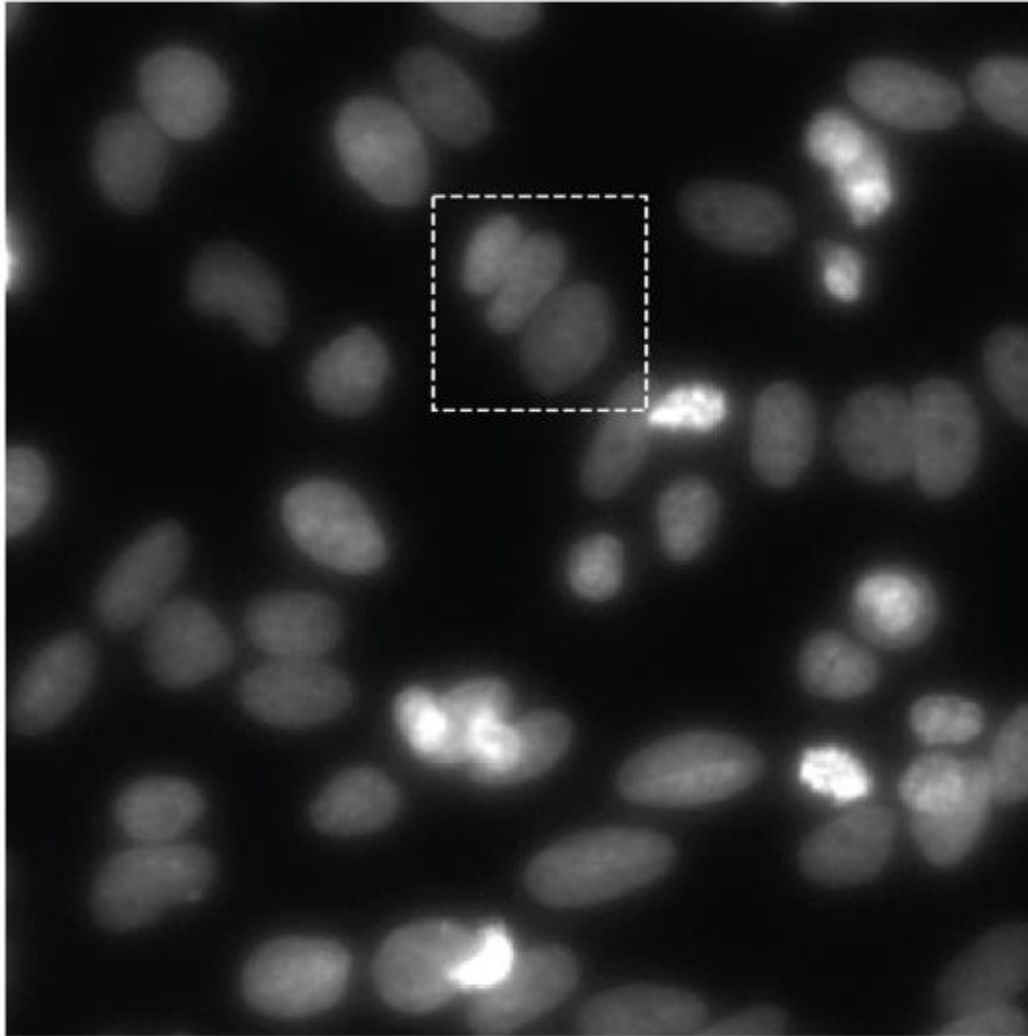
## Strategy:

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

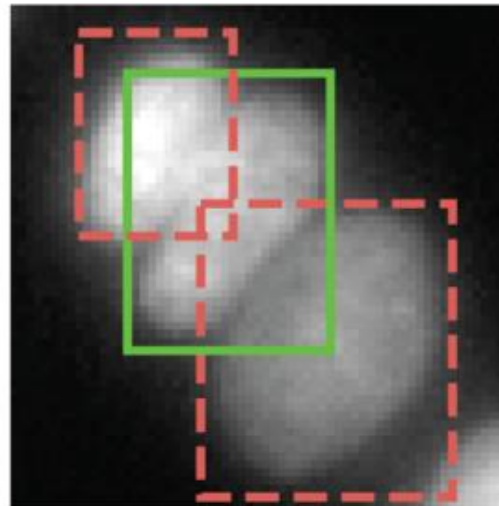
<https://github.com/juglab/n2v>

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

Noisy images + Crowded cells = Common source of segmentation errors



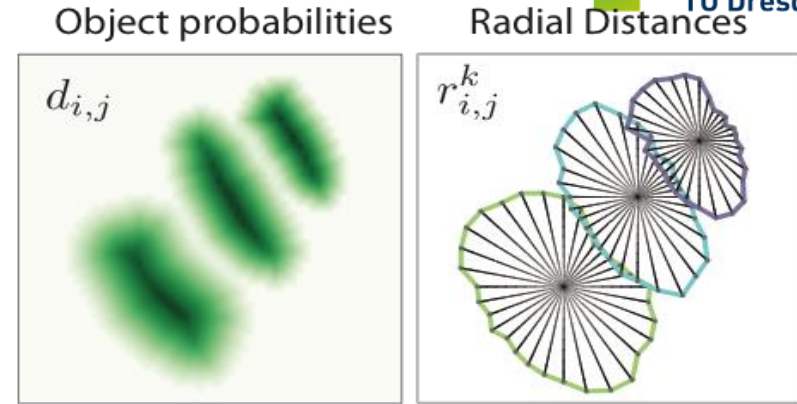
Dense Segmentation  
(e.g. U-Net)



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

## Strategy:

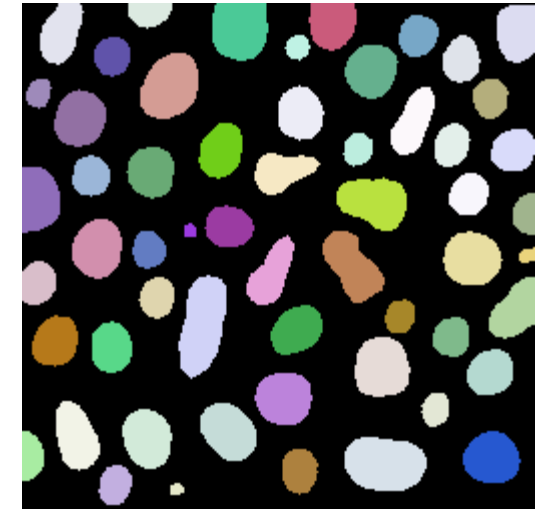
- Add additional information to prediction
- Member pixels of objects (nuclei) can be reached via a straight line from the center



NMS

$r_{i,j}^k$

$d_{i,j}$



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

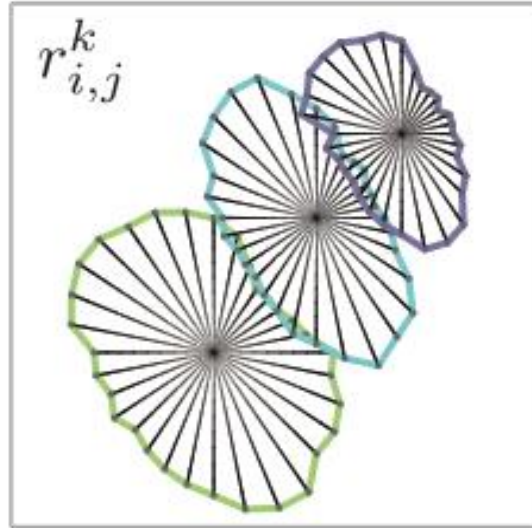
Polygon Selection  
(Non-Maximum Suppression NMS)

Schmid et al., MICCAI (2018)

Object probabilities



Radial Distances



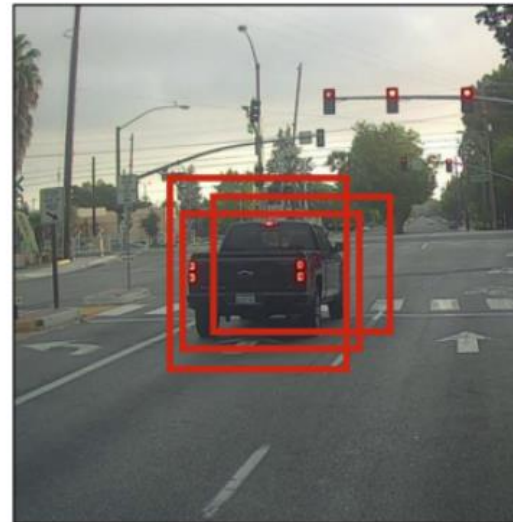
## Problem:

- Multiple candidate points for nucleus center
- Overlapping instance predictions

## 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

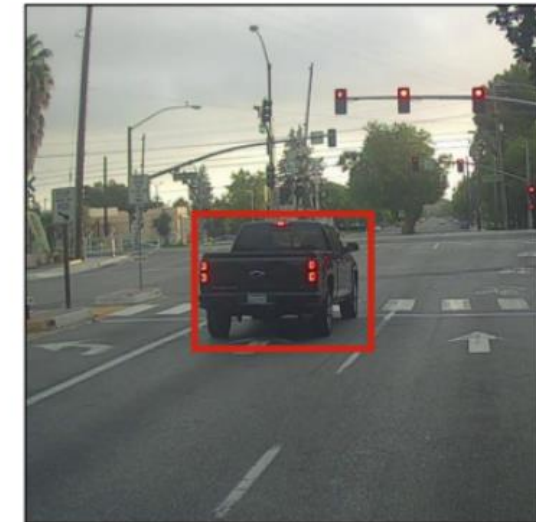
Before non-max suppression



Non-Max  
Suppression



After non-max suppression

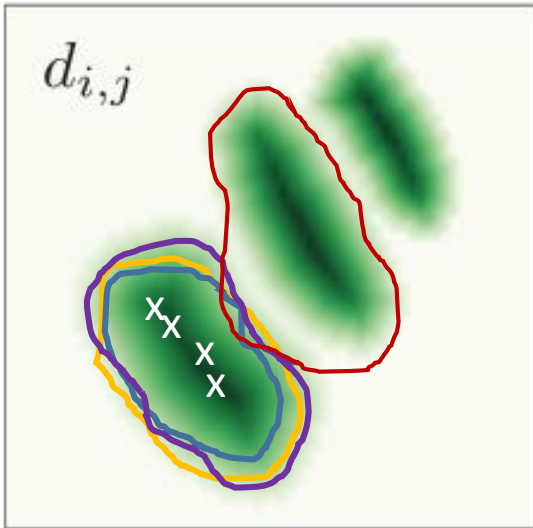


<https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c>

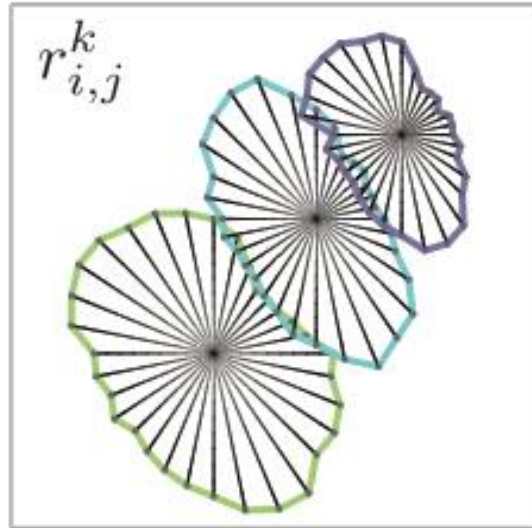
Schmid et al., MICCAI (2018)



Object probabilities











Radial Distances

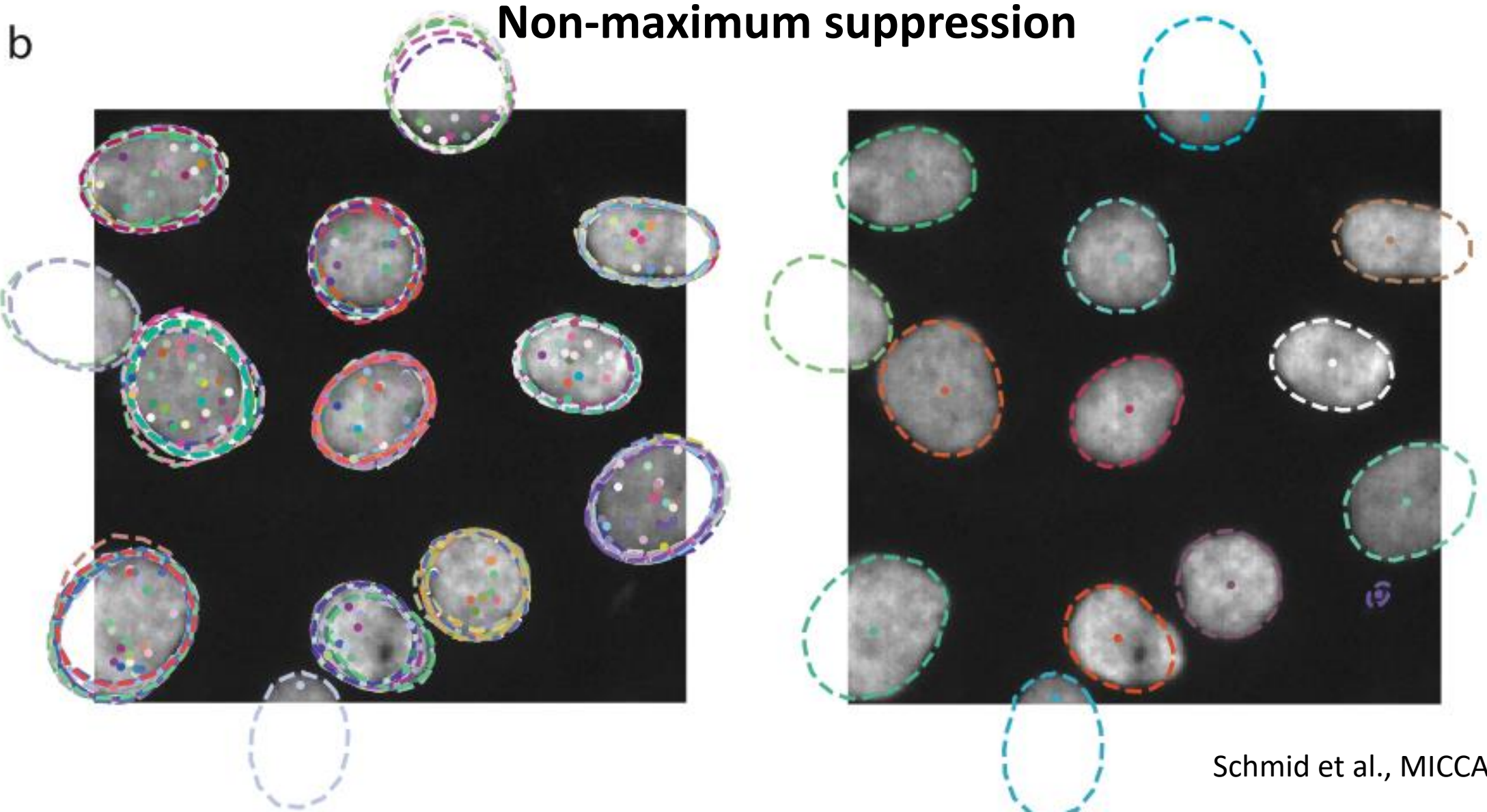


## Non-maximum-suppression (NMS):

- 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!

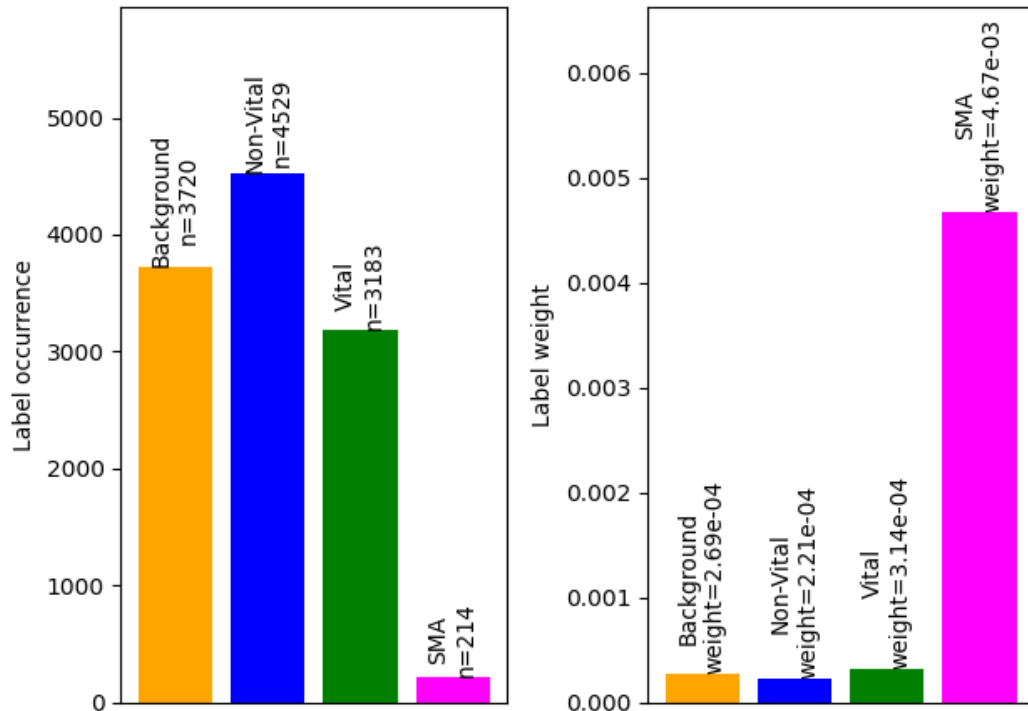
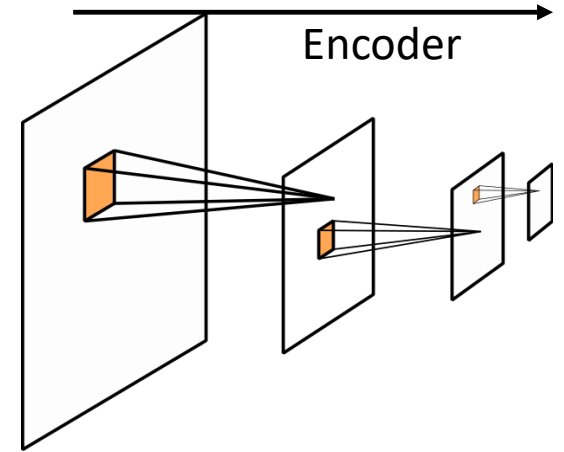




Many prediction frameworks use UNets – similar weak points

## Receptive field:

- Neurons in deeper layers can only “see” parts of the raw image
- Objects must be smaller than receptive field to be detectable



## Unbalanced training data:

- Heterogeneous occurrence of labels in training data
- Rare events will not be caught because they don't harm accuracy much
- Weighted data sampling

### 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/>

Popular frameworks: <https://www.tensorflow.org/> , <https://www.pytorch.org/>

PyTorch

TensorFlow

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

**Memory:** GPU memory limits < RAM memory → Images are tiled and passed through the network in *batches*

**Batch-processing:**

Batch dimension

Channel dimension

→ For 2D multichannel image:  $\text{data.shape} = [B, C, Y, X]$

→ Batch normalization: Data in batch is normalized from  $[-\sigma, \sigma] \rightarrow [-1, 1]$

Higher batch size, smaller tiles → Better generalization of image statistics

Lower batch size, bigger tiles → Potentially larger receptive field

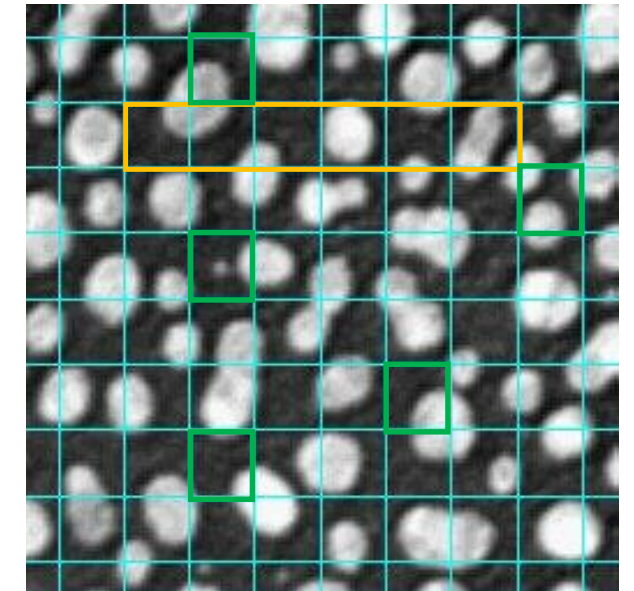
**Train/test/validation split:**

→ During training: Measure performance (“*loss*”) of network, **update weights**

→ During testing: Measure performance of updated network, **keep weights**

→ Validation: Measure performance in **unseen** validation dataset

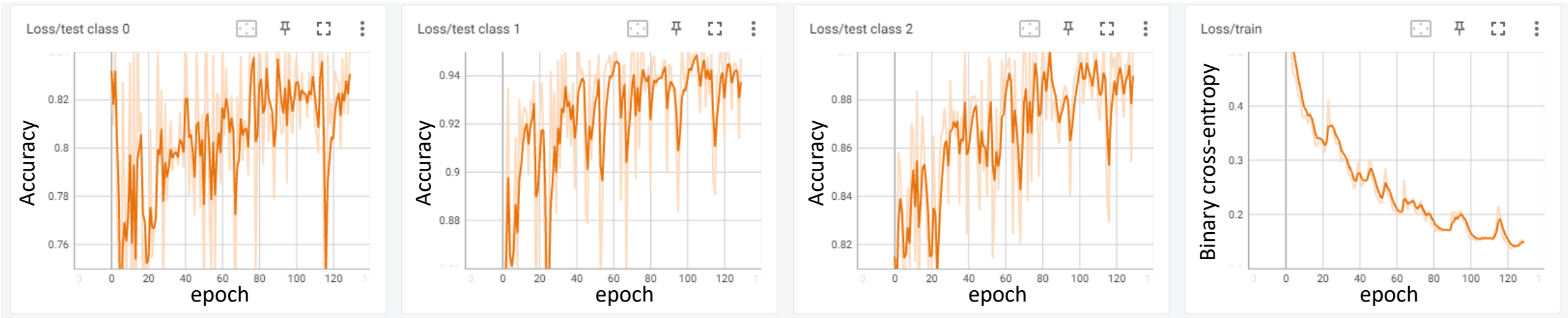
1 epoch



Tiles in batch

Tiles in batch

**Training loss:** Binary cross-entropy (Measure for information in image A contained in image B)

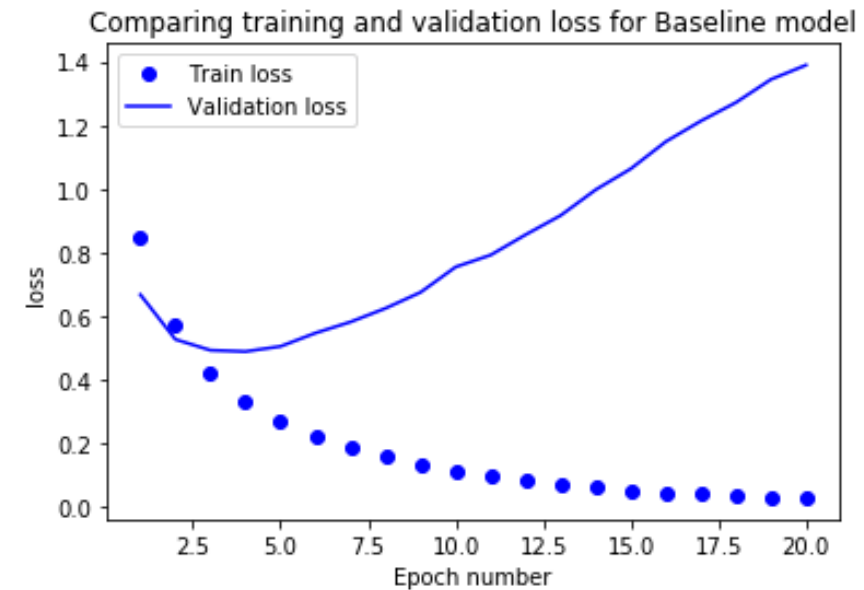


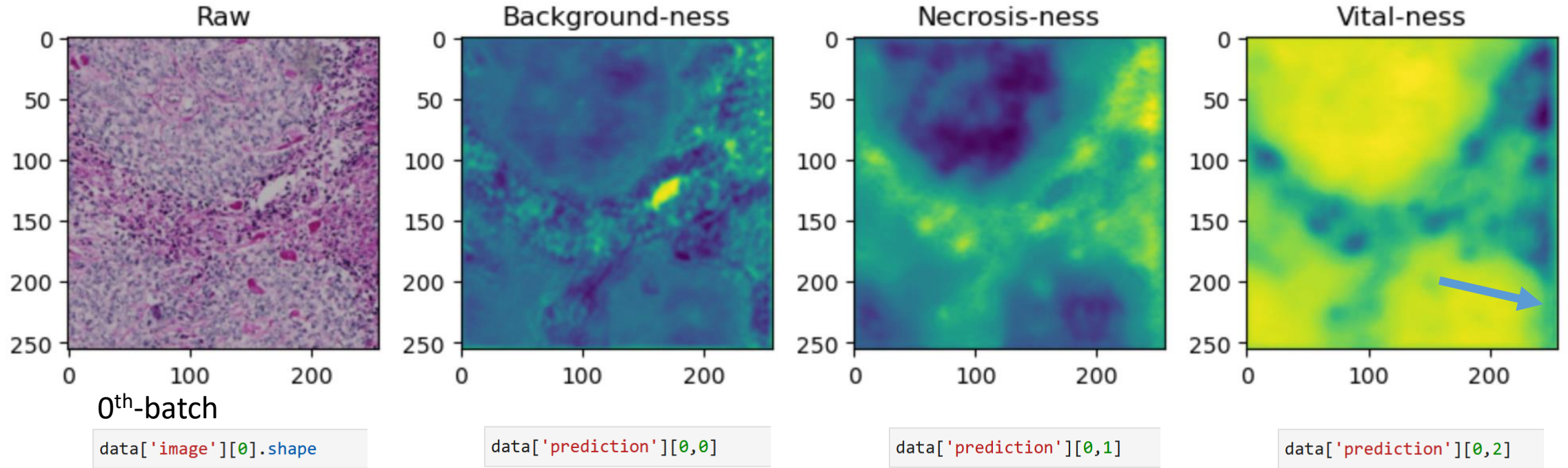
**Test loss:** Label-wise accuracy

→ We use different metrics in training/testing!

**Overfitting:**

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





`torch.Size([3, 256, 256])`

## Stride:

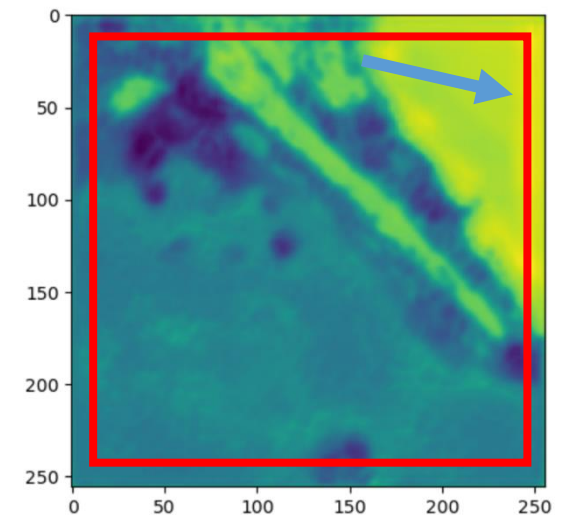
- Network can not predict accurately at image edges
- We throw away a X-pixel wide margin at the image edge

What we do here is....

Semantic segmentation

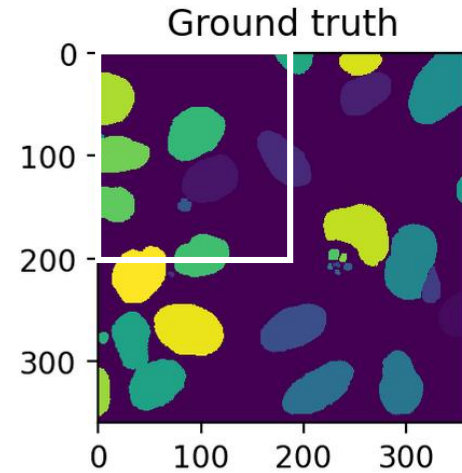
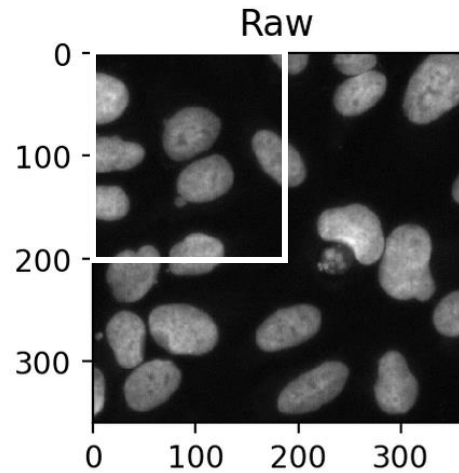
Instance segmentation

Class regression

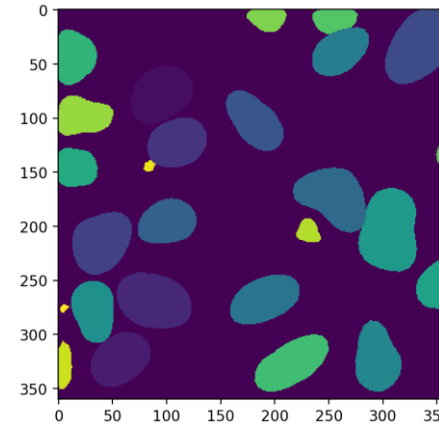




# Example



Prediction

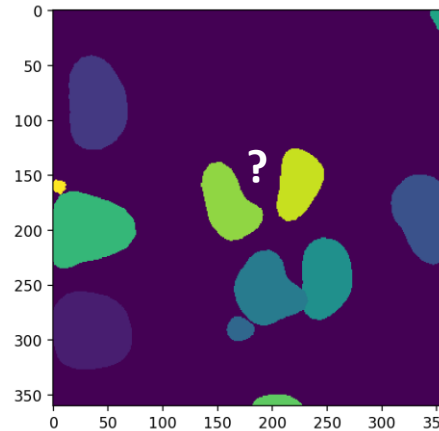
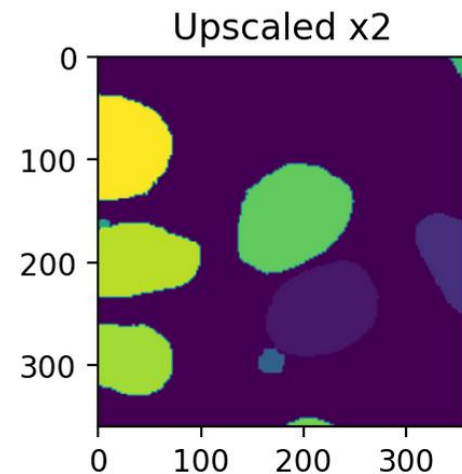
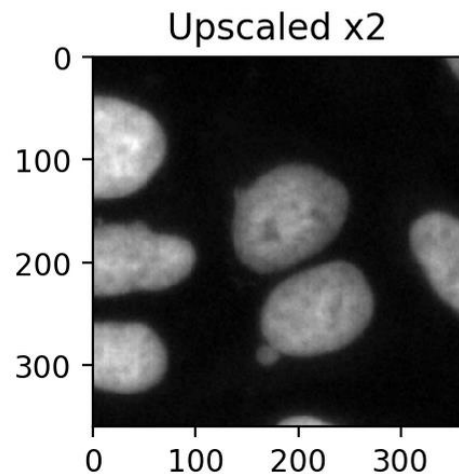


What happened here?

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 > better model
- Deep learning often performs fantastic – *but you don't know why*