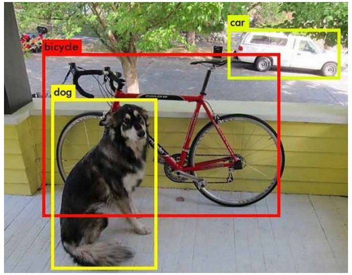
Real DL examples, from colleagues and us

Computer vision intro

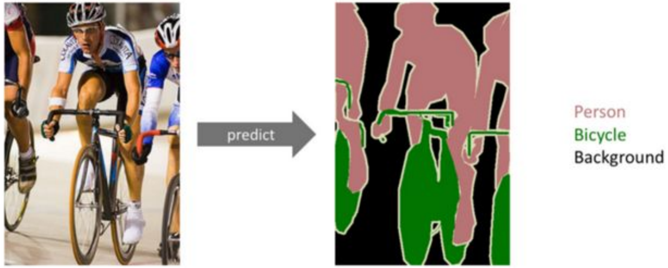
1) Red Blood Cells classification
2) Micro-bead tracking
3) Bacteria semantic segmentation

Computer Vision - possible tasks

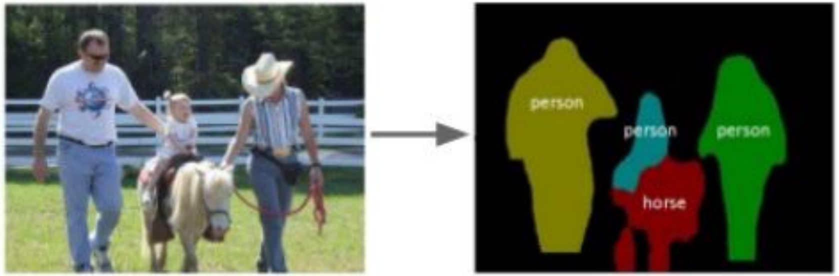
classification
“cat”



classification + localization
class + bounding box



Semantic segmentation
each pixel : class

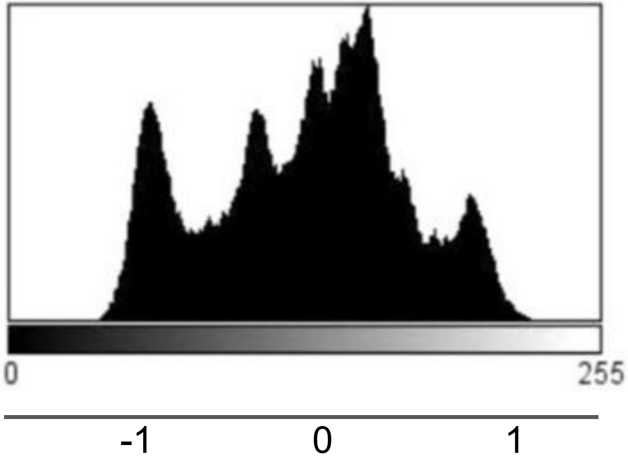


Instance segmentation
each pixel:
class “person” + instance #1
class “person” + instance #2 ...

Important points

1. Your images should be normalized:

    - Average intensity equals to 0
    - Standard deviation equals to 1



Original

Transformed

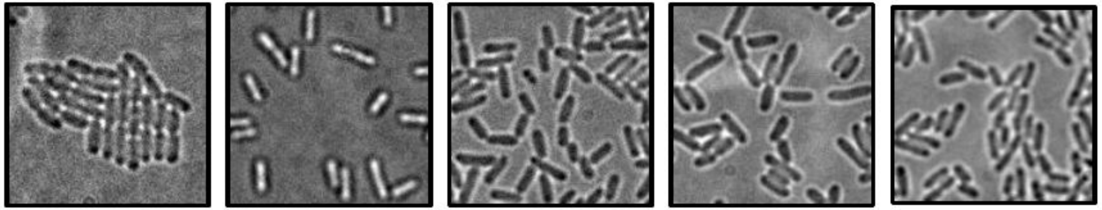
```
image = image - mean(image)
image = image/std(image)
```

Important points

1. Your images should be normalized:

    - Average intensity equals to 0
    - Standard deviation equals to 1

2. Your set of images should be **<u>as representative and diverse as possible</u>**



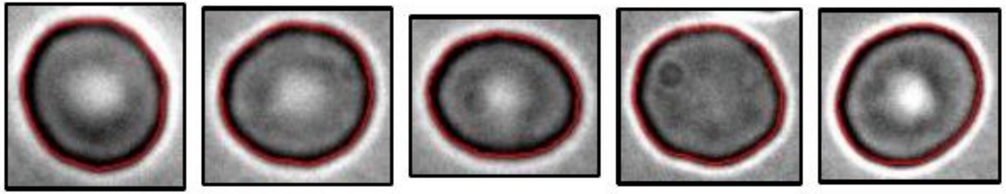
<u>Variation in :</u>

- Density of cells
- Different focal planes
- Light intensity
- etc.

Good generalization!

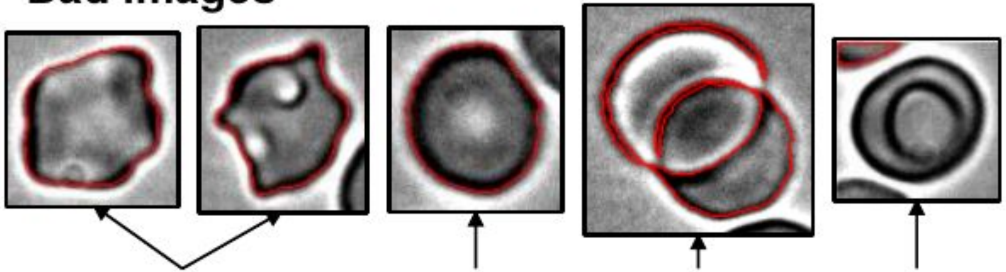
1) Red Blood Cells classification

Good images



In-focus image of RBC with a properly defined contour (red)

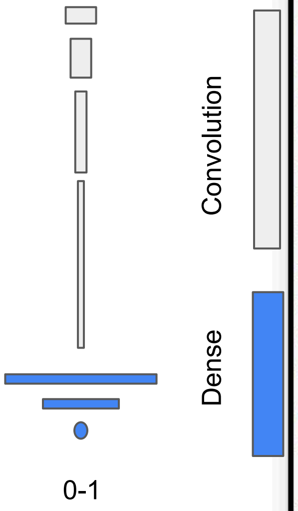
Bad images



Poor quality images that need to be discarded from the analysis

1) Red Blood Cells classification - ConvNet classifier

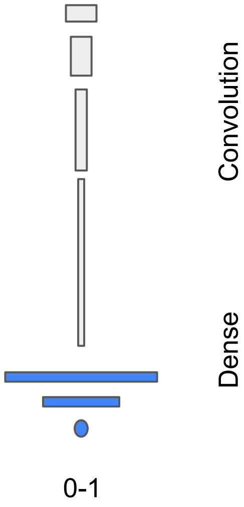
<u>Type of data :</u> png RGB images resize to 85x85 and **normalized**
<u>Size of the training set :</u> 956 good images / 2000 bad images
<u>Size of the validation/testing sets :</u> 319 good images / 667 bad images

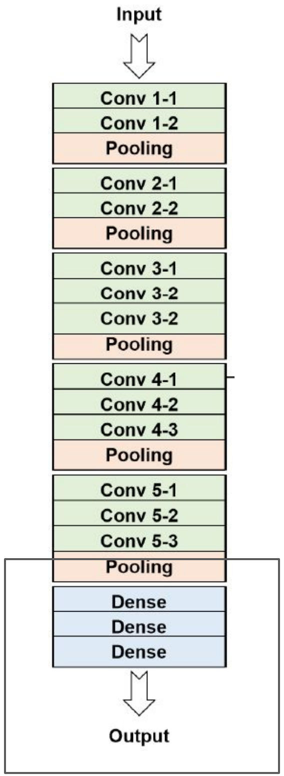



```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(85,85,3)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64,(3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128,(3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128,(3,3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer = optimizers.RMSprop(lr=1e-4),
            loss='binary_crossentropy',
            metrics=['accuracy'])

model.summary()
```

Number of trainable parameters : <u>1,453,761</u>

1) Red Blood Cells classification - Transfer learning



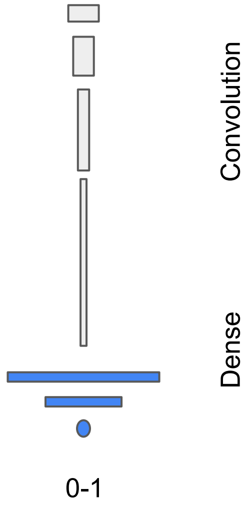


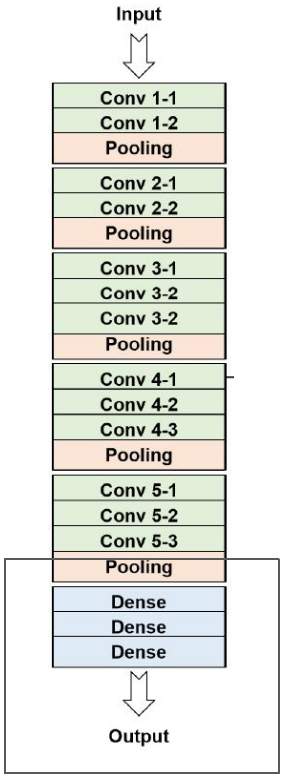
VGG16 already trained on millions of images for many classes

Transfer learning:
change or reset only the last "dense classifier", keep the rest fixed, and retrain on your images

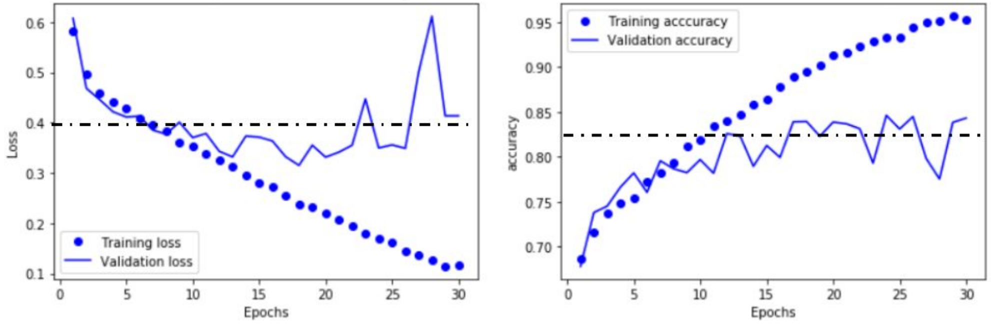
1) Red Blood Cells classification -

Transfer learning

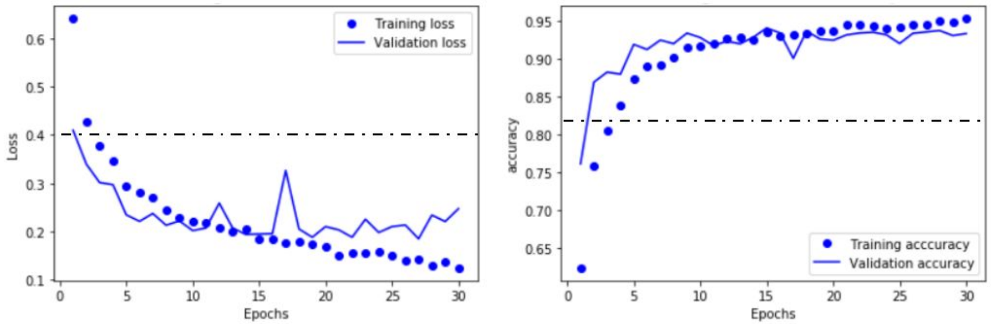




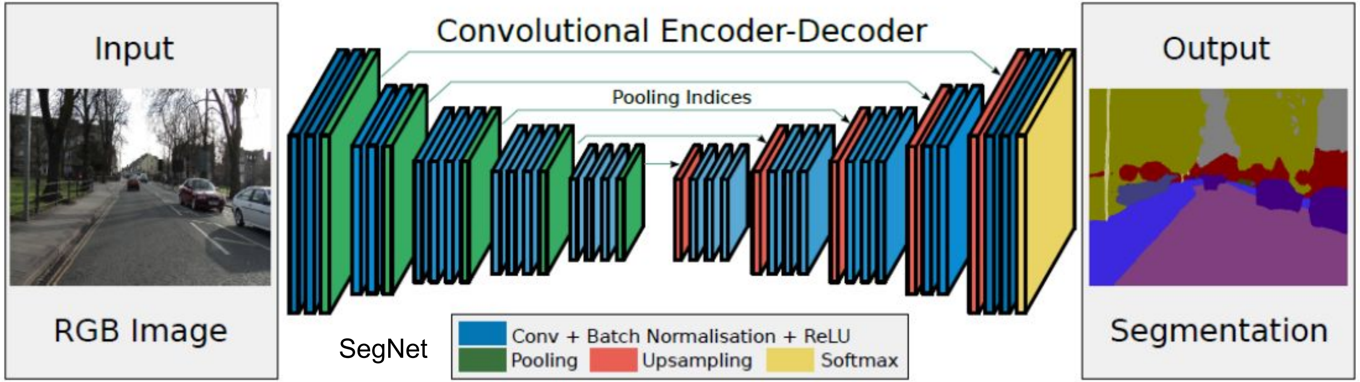
Basic convNet



Fine-tune a pre-trained network



Fully convolutional



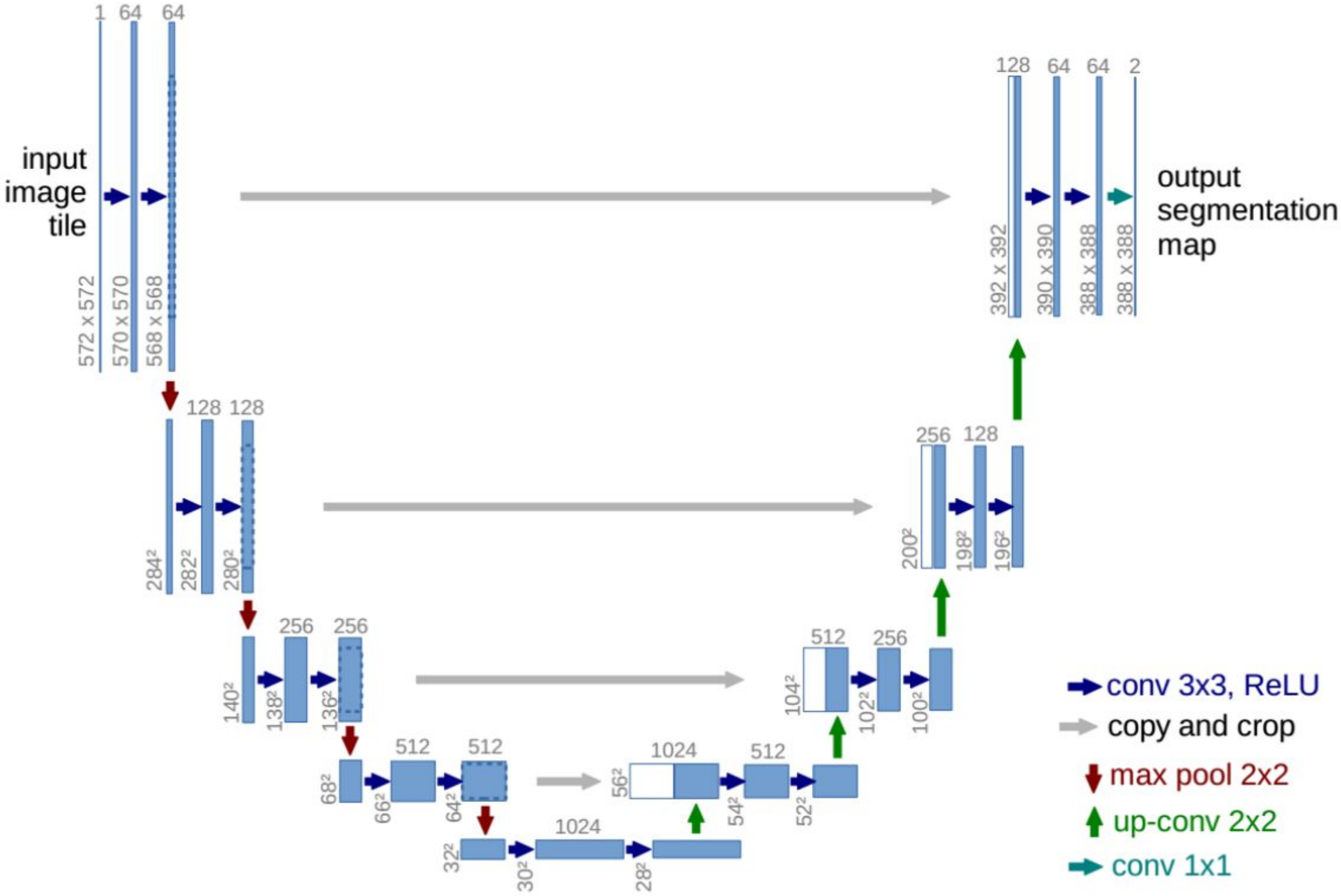
SegNet

This type of neural network architecture has many advantages:

- Simple to implement
- **Faster**
- **Less heavy** on memory since no fully connected layers
- **No restriction** on the size of the input images
- The **output image is already segmented**

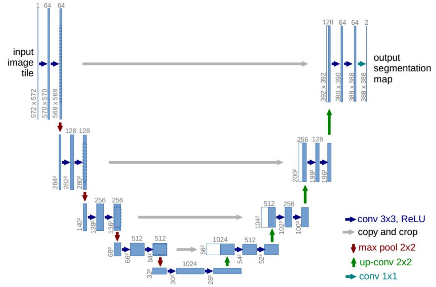
Badrinarayanan et al. IEEE 2016 – Noh et al. ArXiv 2015 – Ronnerberger et al. ArXiv 2015

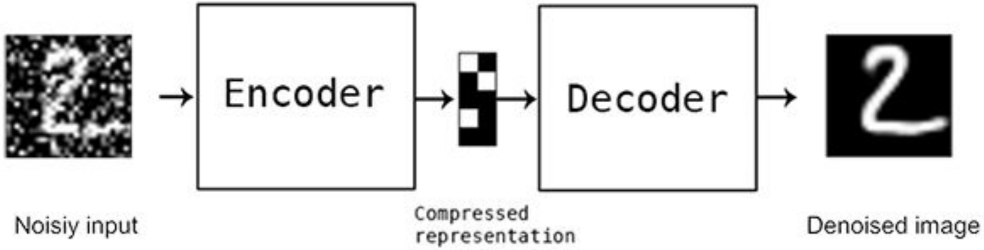
UNET

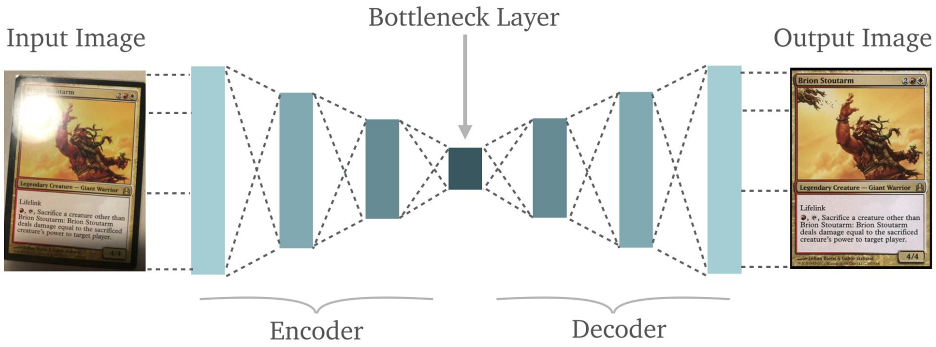


Badrinarayanan et al. IEEE 2016
Noh et al. ArXiv 2015
Ronnerberger et al. ArXiv 2015

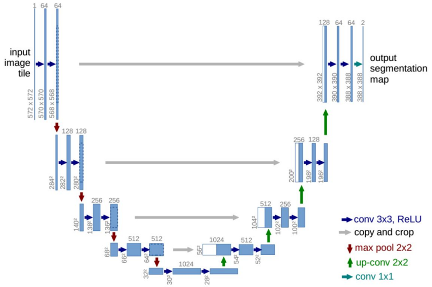
Fully conv. application - denoising

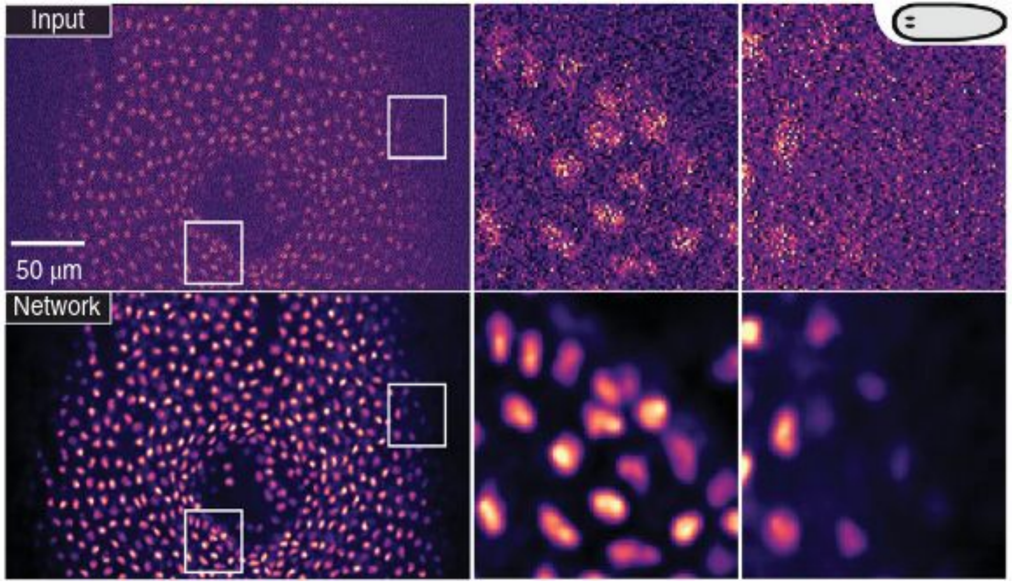






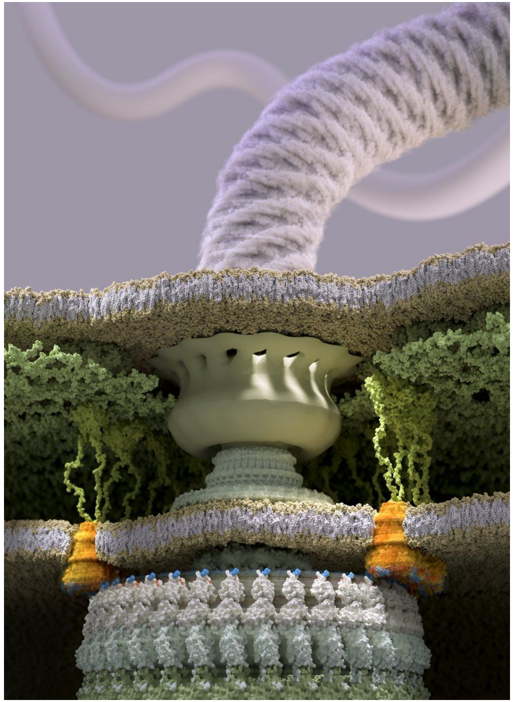
Fully conv. application - denoising

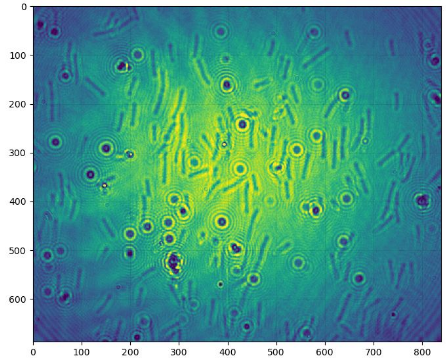


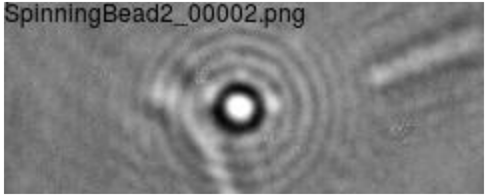


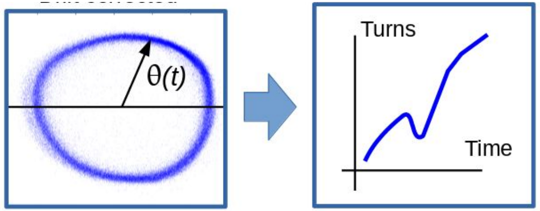
https://github.com/CSBDeep/CSBDeep - Weigert et al. 2017. Content-aware image restoration: pushing the limits of fluorescence microscopy

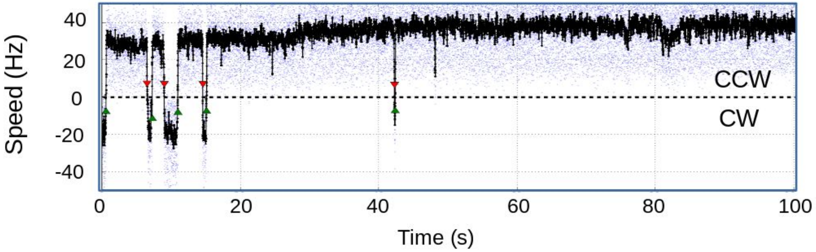
2) Tracking of microbeads



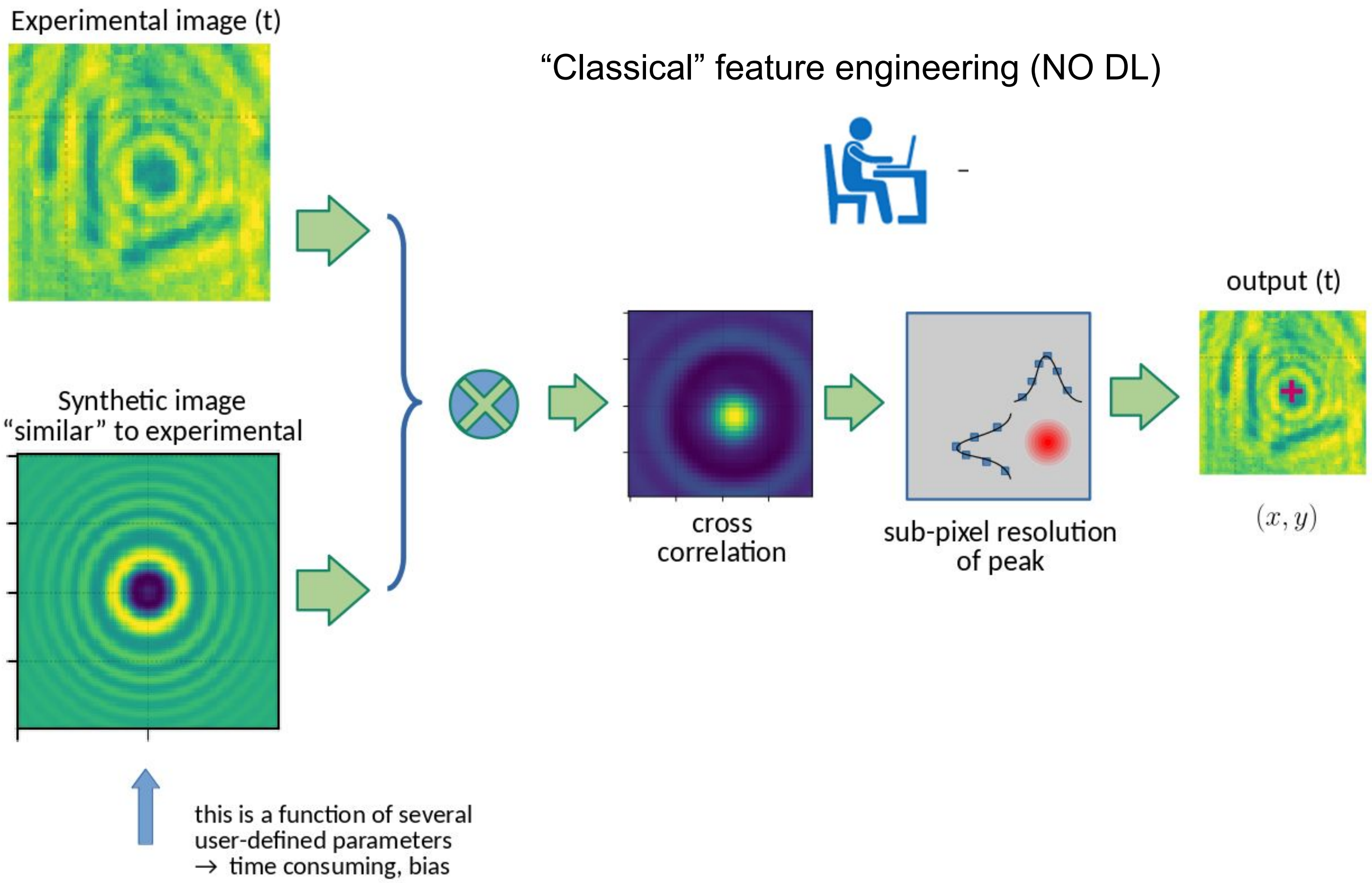




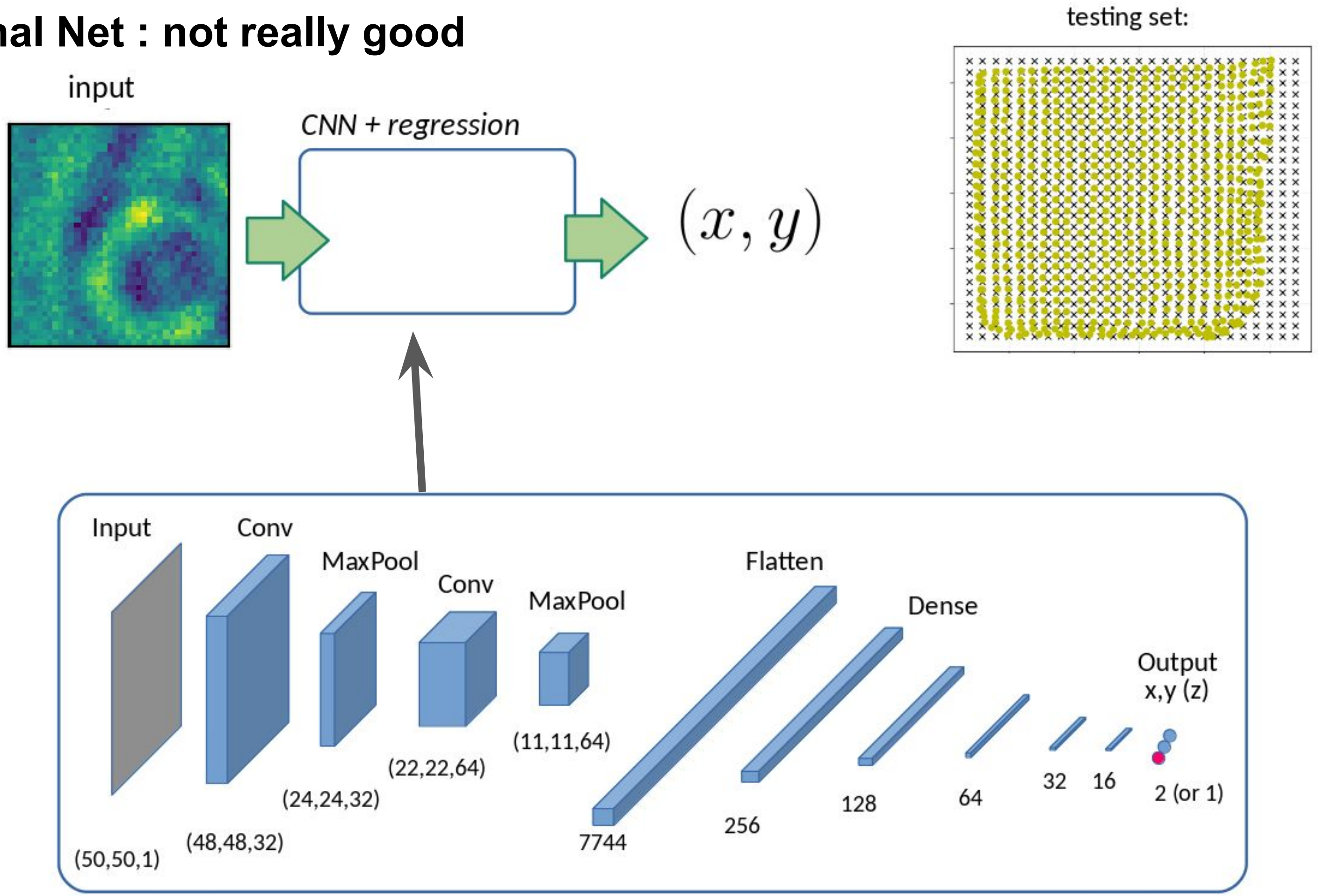




2) Tracking of microbeads

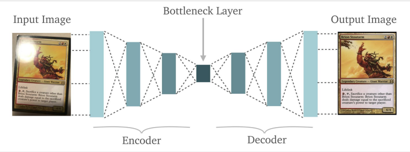


2) Tracking of microbeads
Convolutional Net : not really good



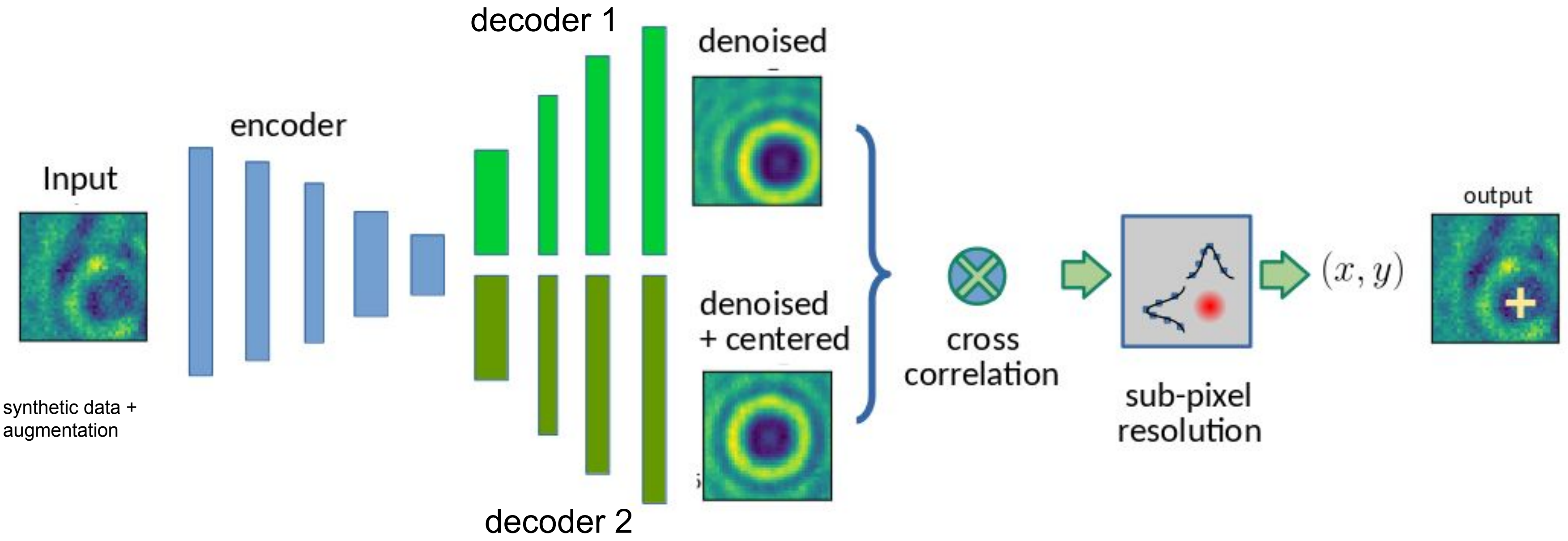
2) Tracking of microbeads - Hybrid Unet + “classical”

Deep learning

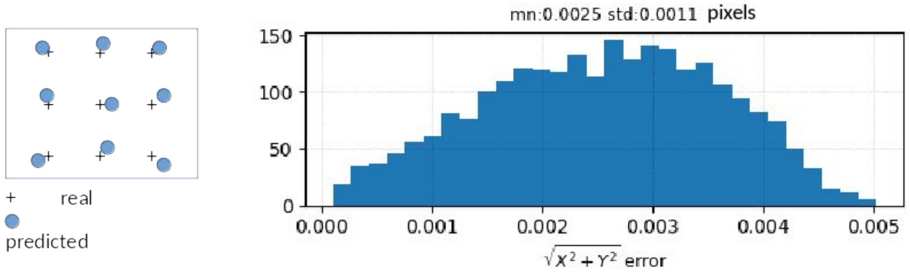


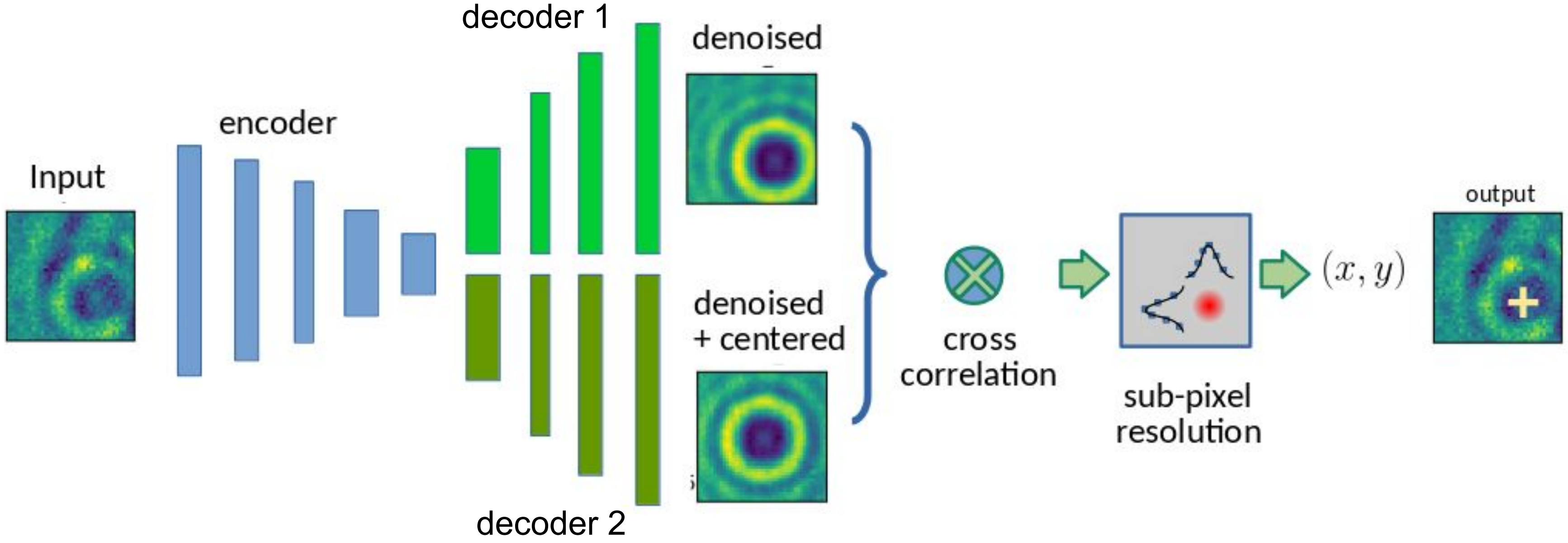
+

“Classical ML”

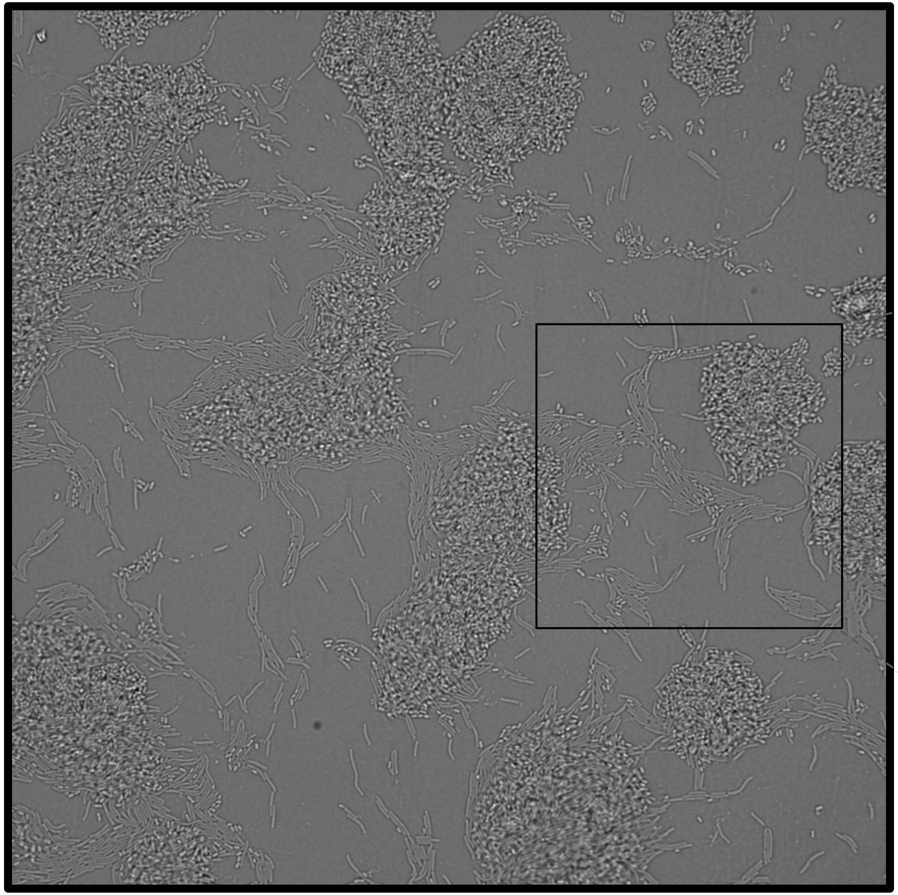


2) Tracking of microbeads - Hybrid Unet + “classical”

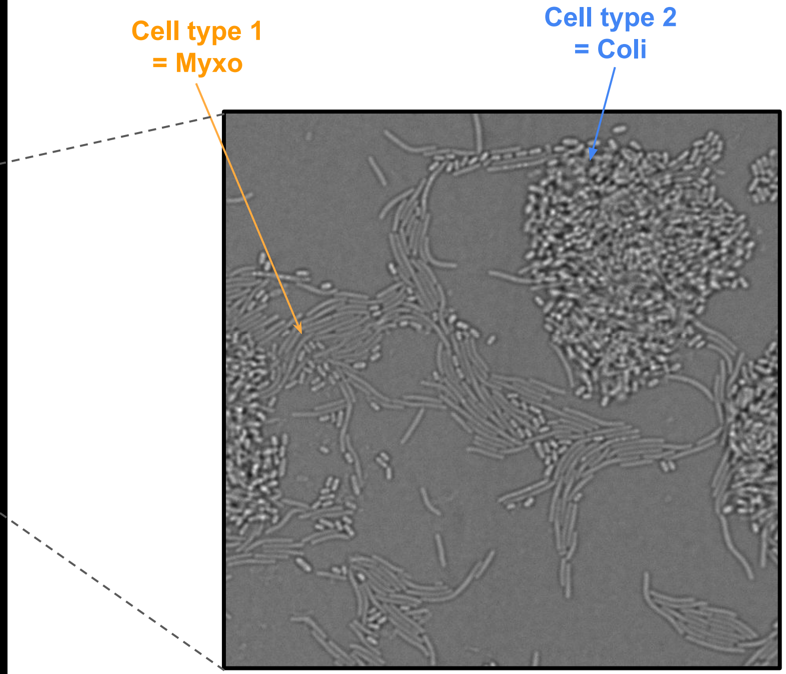




3) Semantic segmentation of bacteria images

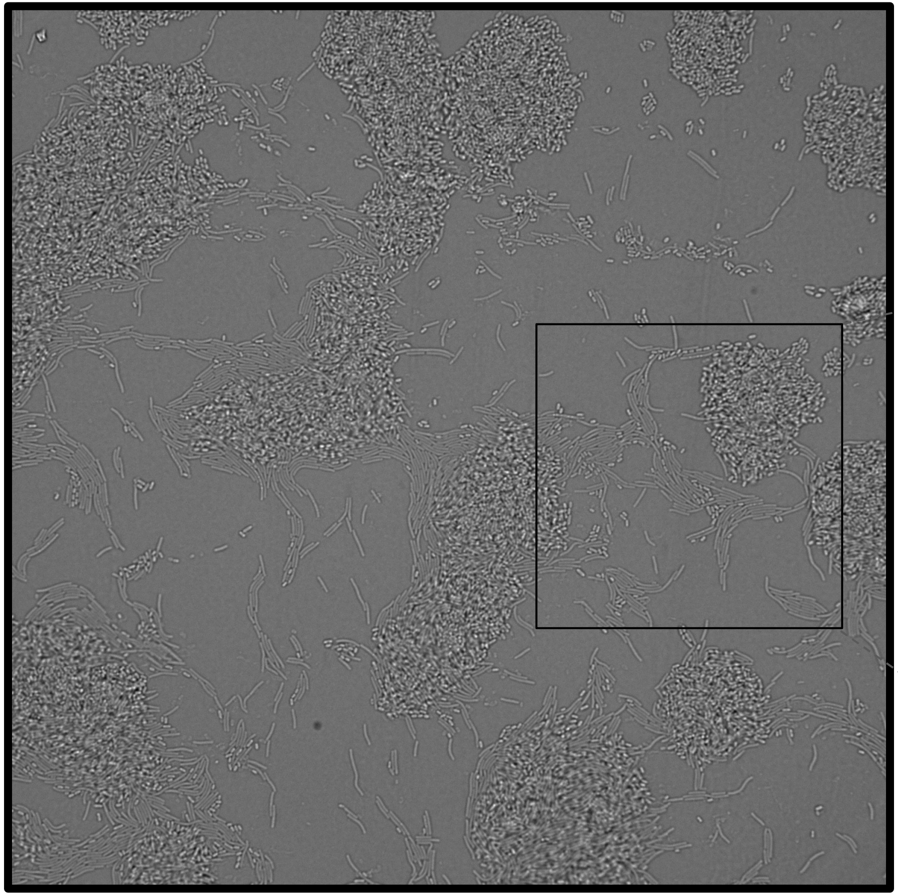


Brightfield images of bacteria (2048x2048 pixels). Two types of cells :

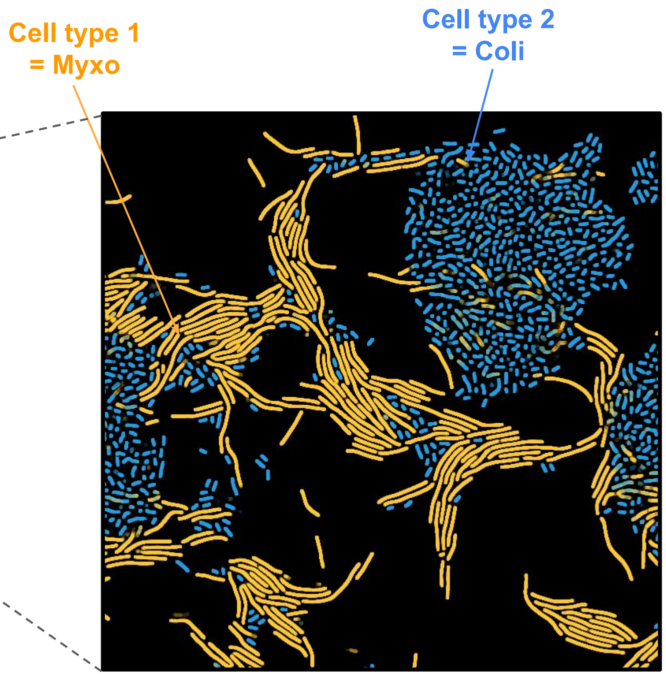


Images credit : Sara Rombouts

3) Semantic segmentation of bacteria images



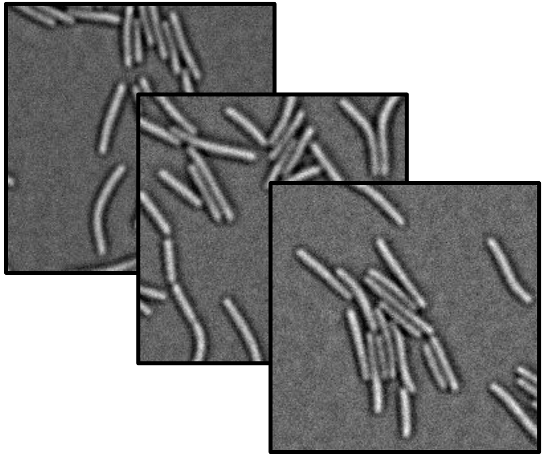
Brightfield images of bacteria (2048x2048 pixels). Two types of cells :



Images credit : Sara Rombouts

3) Semantic segmentation - How to start?

- First training set : <u>start **SIMPLE**</u>



200x200 pixels

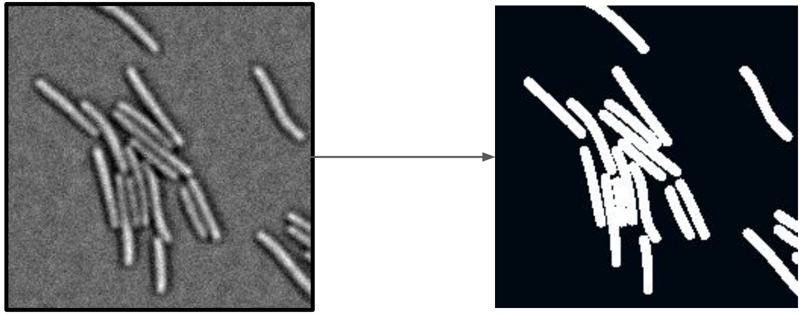
- Small images BUT large enough to have entire cells
- Use **pre-existing tools to create the labeled images** (Ilastik, LabKit, Imaris, etc.)

3) Semantic segmentation - How to start?

- First training set : <u>start **SIMPLE**</u>

- Choose carefully your classes

  <u>2 classes</u> : background (0) & cells (1)



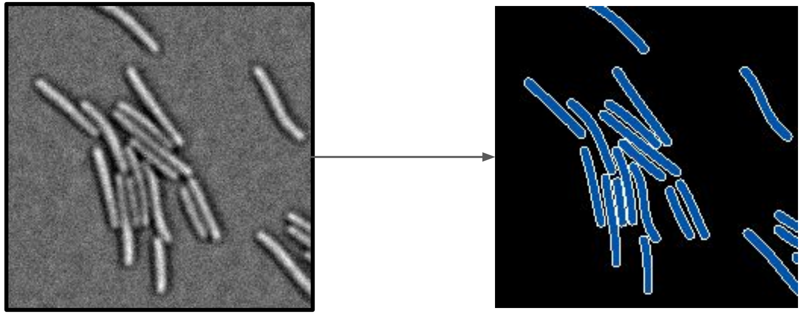
Fused cells

3) Semantic segmentation - How to start?

- First training set : start **SIMPLE**
- Choose carefully your classes

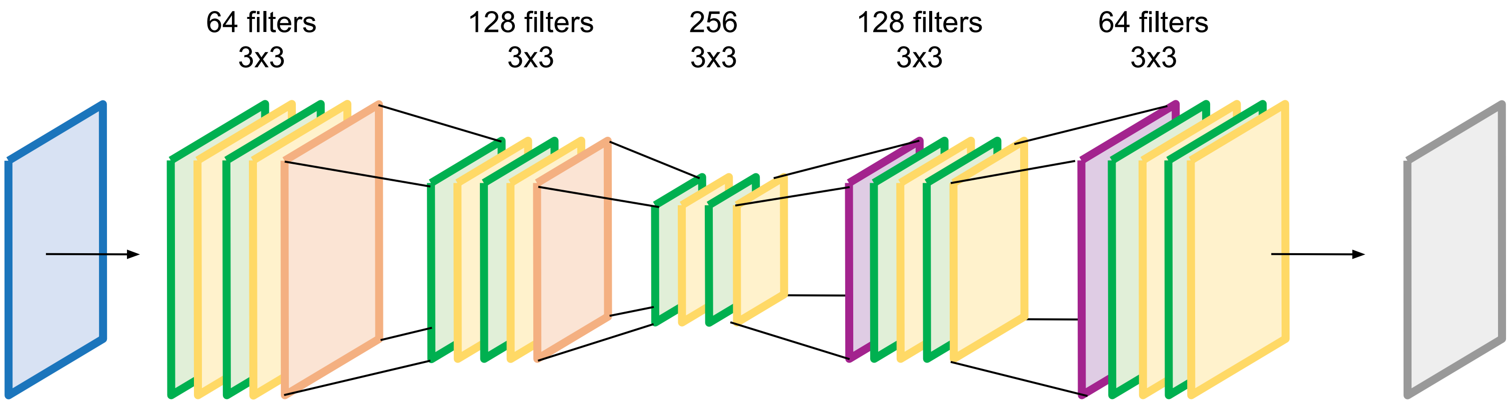
  2 classes : background (0) & cells (1)

  3 classes : background (0) & cell contour (1) & cell body (2)



3) Semantic segmentation - How to start?

- First training set : <u>start **SIMPLE**</u>
- Choose carefully your classes
- Use a SOLID BASELINE for the model : Unet



3) Semantic segmentation - Evaluation of the model

- After the training, metrics are available to evaluate the performance of your network (Accuracy, IoU, etc.).

  Unet training :
    - mean accuracy : 72.9%
    - mean IoU : 61.8%

3) Semantic segmentation - Evaluation of the model

- After the training, metrics are available to evaluate the performance of your network (Accuracy, IoU, etc.).

Unet training :

- mean accuracy : 72.9%
- mean IoU : 61.8%

Confusion matrix :

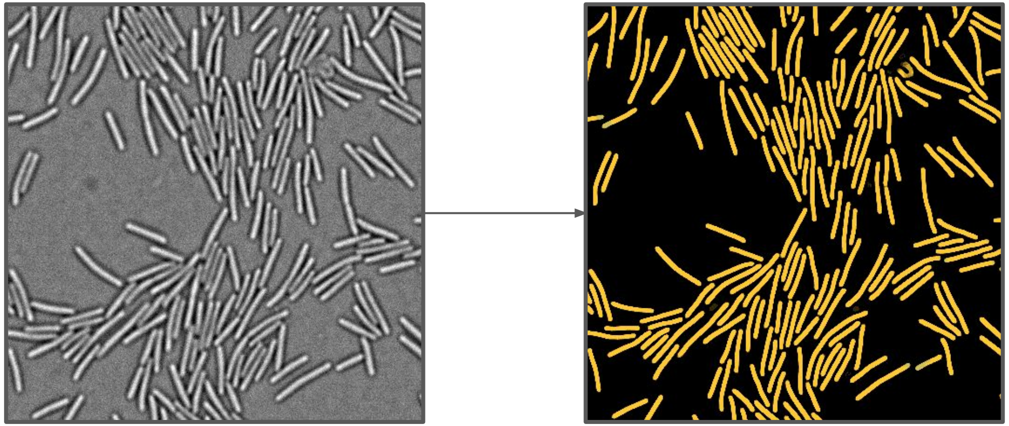
	Background	**Cell body**	**Cell contour**
Background	94.9%	0.96%	4.14%
Cell body	2%	93.4%	4.6%
Cell contour	15.8%	24.1%	60.1%

3) Semantic segmentation - Evaluation of the model

- After the training, metrics are available to evaluate the performance of your network (Accuracy, IoU, etc.).

Unet training :

  - mean accuracy : 72.9%
  - mean IoU : 61.8%



3) Semantic segmentation - Optimization of the model

- After the training, metrics are available to evaluate the performance of your network (Accuracy, IoU, etc.).

- Fine-tune the parameters of your network :
  - size of the filters
  - number of layers
  - optimizer
  - batch size
  - etc.

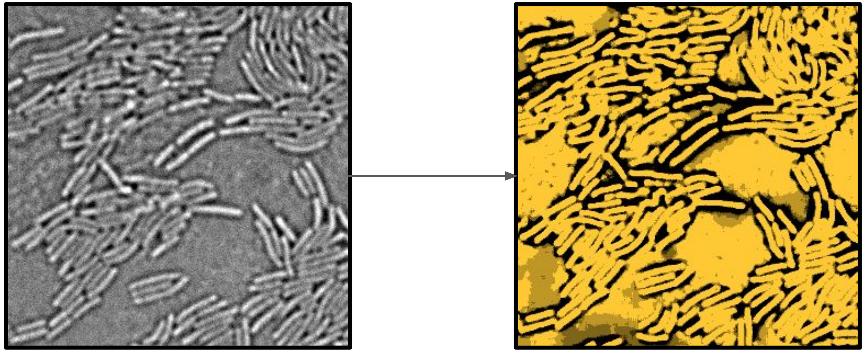
⇨ **"Quick" and easy gain** of a few percent of accuracy ✓

3) Semantic segmentation - Optimization of the model

- After the training, metrics are available to evaluate the performance of your network (Accuracy, IoU, etc.).

- Fine-tune the parameters of your network

- Image augmentation :
    - horizontal and vertical flip
    - **add synthetic/real noise**
    - add out of focus images to the data



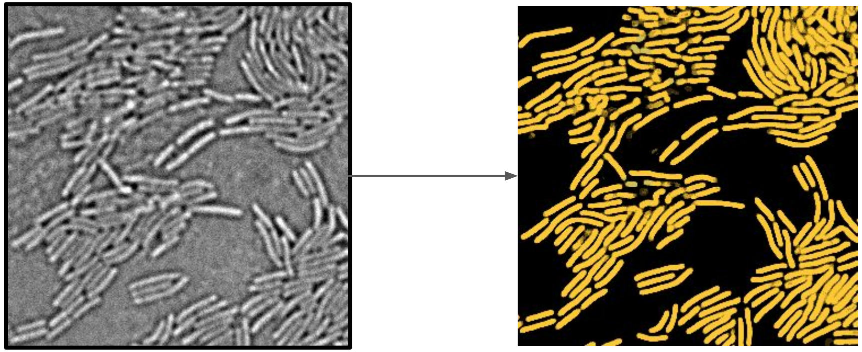
3) Semantic segmentation - Optimization of the model

- After the training, metrics are available to evaluate the performance of your network (Accuracy, IoU, etc.).

- Fine-tune the parameters of your network

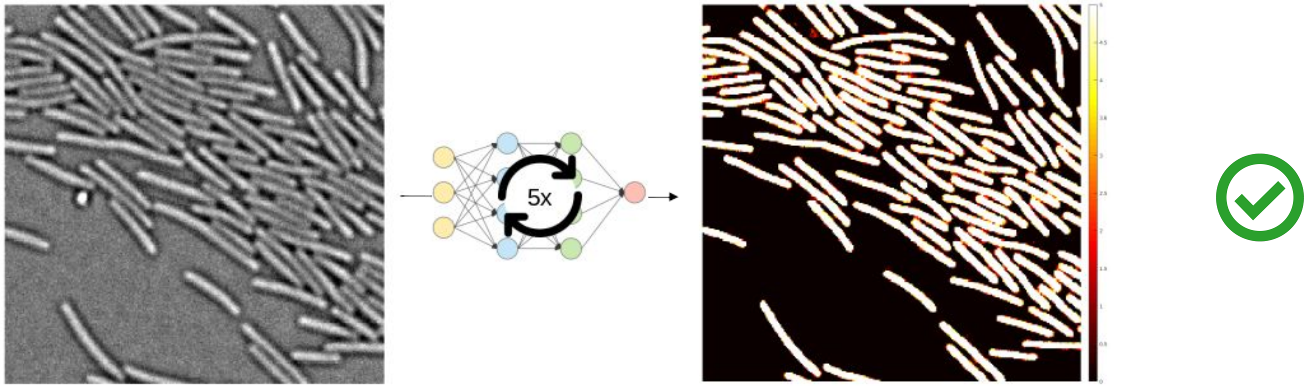
- Image augmentation :
  - horizontal and vertical flip
  - **add synthetic/real noise**
  - add out of focus images to the data

Helped improved the generalization of the network

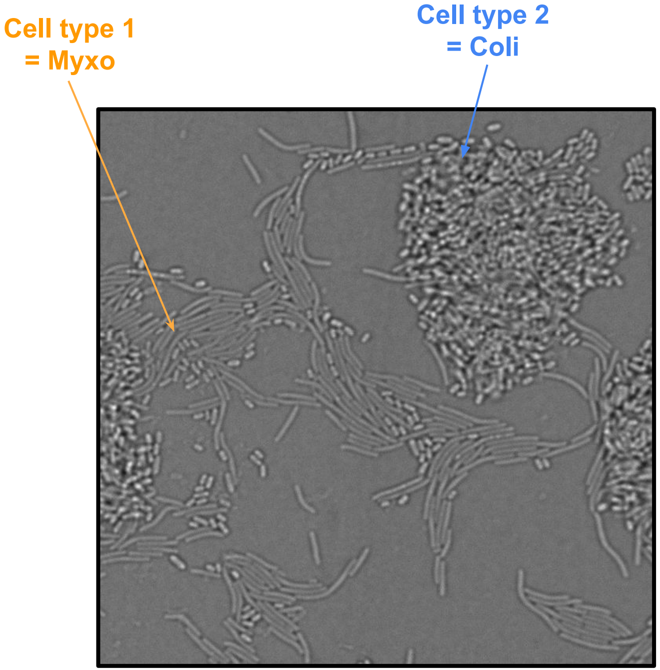


3) Semantic segmentation - Optimization of the model

- After the training, metrics are available to evaluate the performance of your network (Accuracy, IoU, etc.).
- Fine-tune the parameters of your network
- Image augmentation
- Combine multiple training together :



3) Semantic segmentation - Add useful information



Unet with 5 classes :

- background (0)
- Myxo body (1) and contour (2)
- Coli body (3) and contour (4)

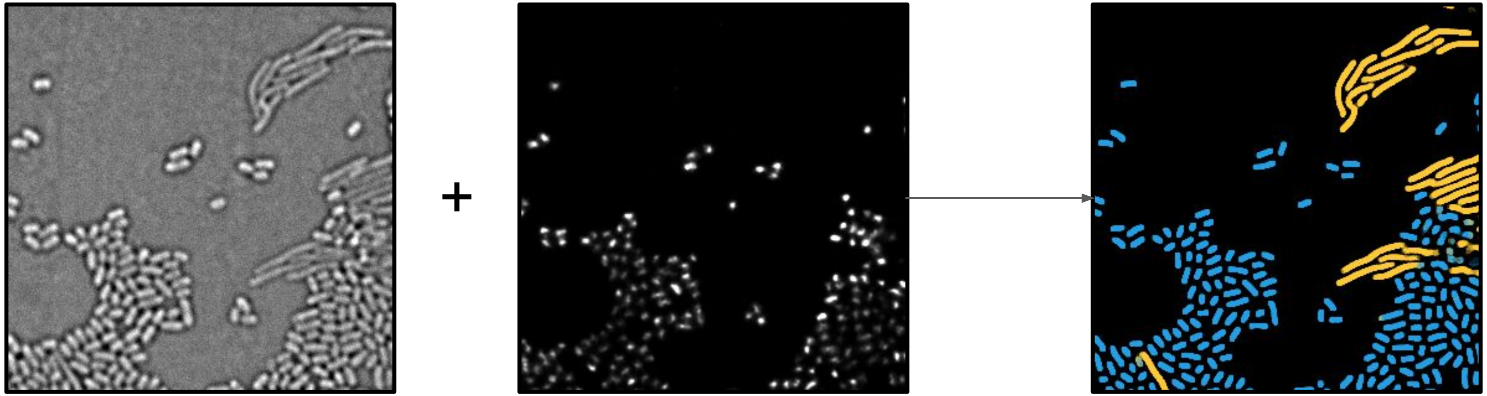
3) Semantic segmentation - Add useful information

- Create two separate networks :
  - one for cell segmentation
  - one for cell differentiation

⇒ Improved the differentiation but not enough

- Add more images → too much work
- Add fluorescence images

Coli fluorescence



3) Semantic segmentation - Take home messages

I. Always start with **simple data and a solid baseline**

II. Setup your **own evaluation strategy** :

  - build a test set representative of your problem
  - choose the right performance indicators for your problem

III. **Know your data** :

  - your training set should be as representative as possible of your data
  - double/triple check for errors