

Machine Learning for Pixel and Object Segmentation



Robert Haase

Reusing some material from
the Scikit-Learn community

Overview

- Machine learning for Pixel and Object Classification
 - Random Forest Classifiers
- Python
 - scikit-learn / napari
 - Accelerated pixel and object classification (APOC)

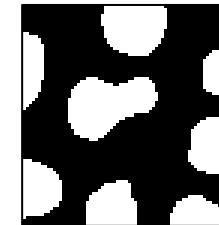
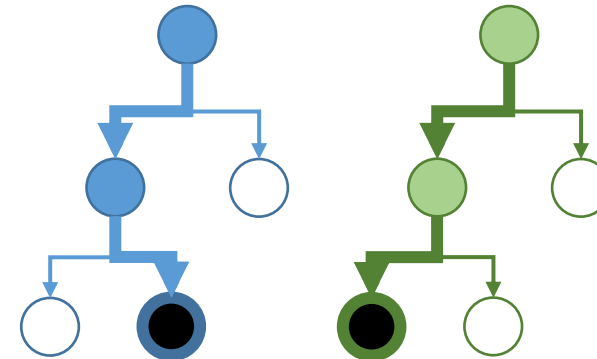
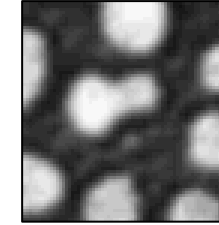
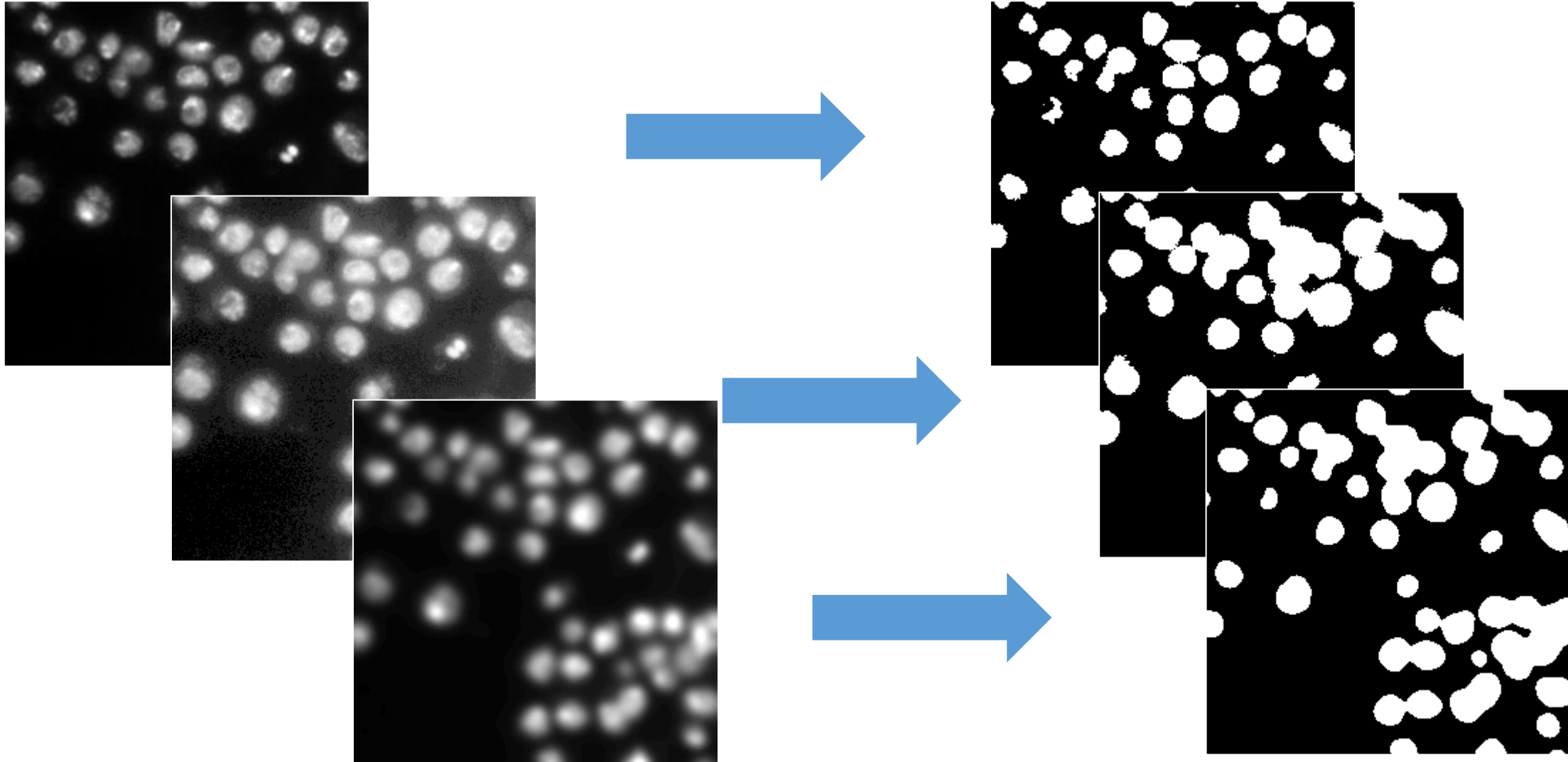
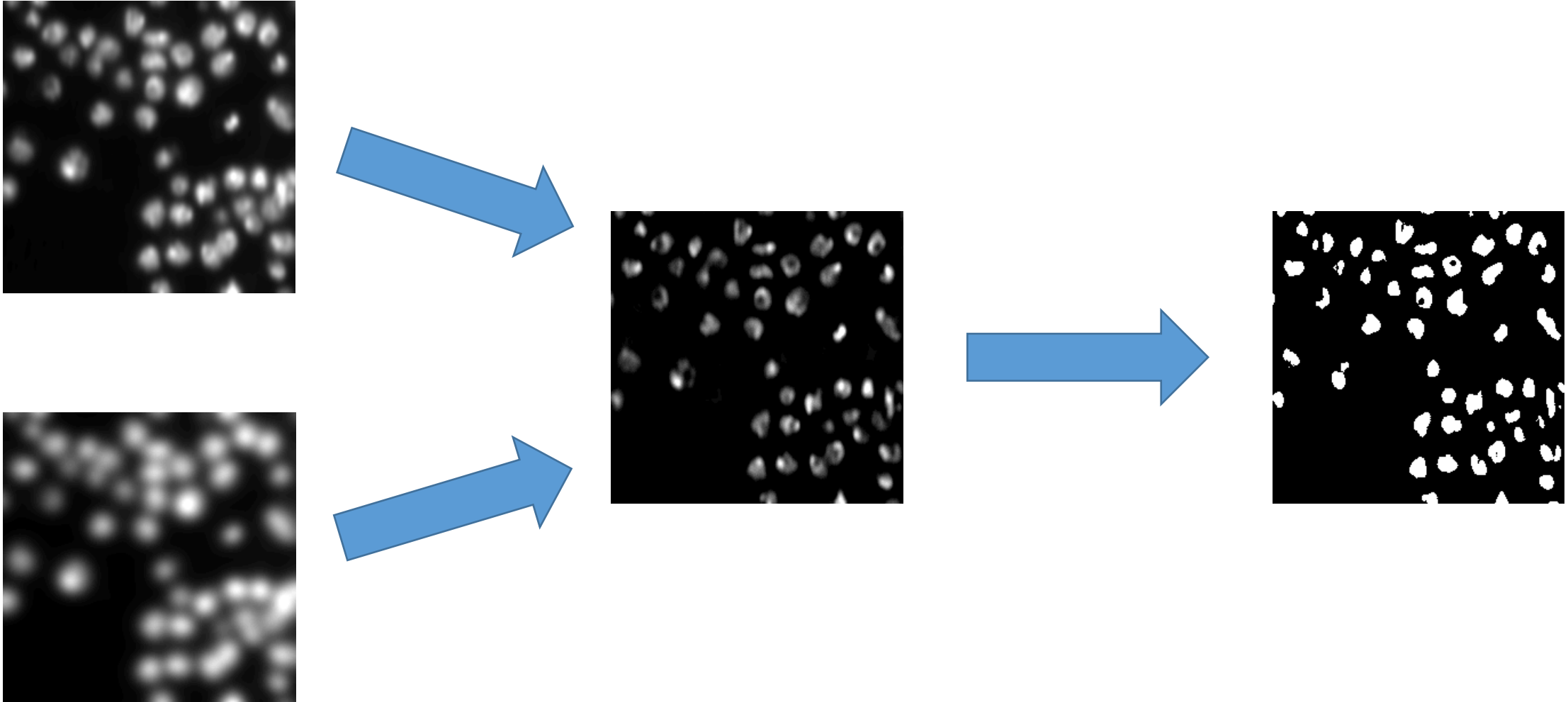


Image segmentation using thresholding

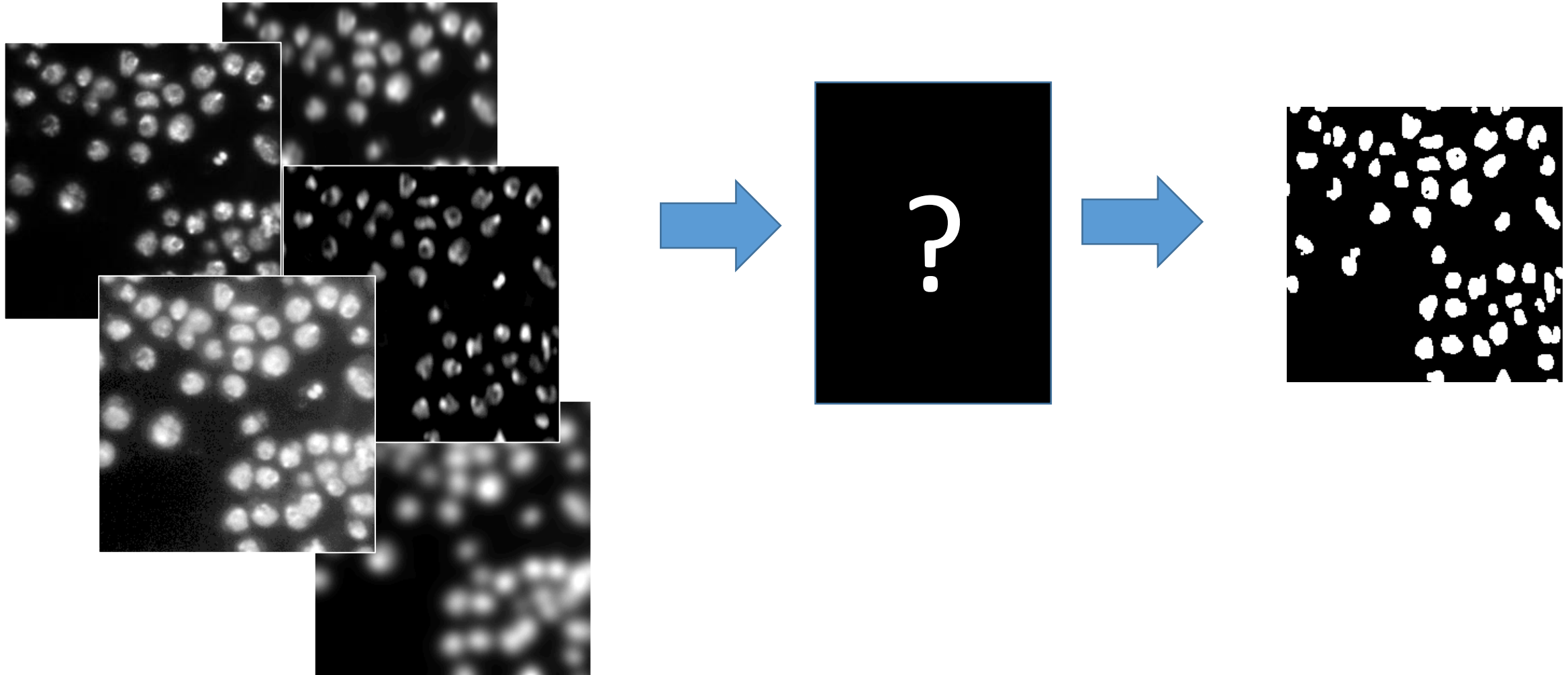
- Recap: Finding the right workflow towards a good segmentation takes time



- Recap: Combining images, e.g. using Difference of Gaussian (DoG)



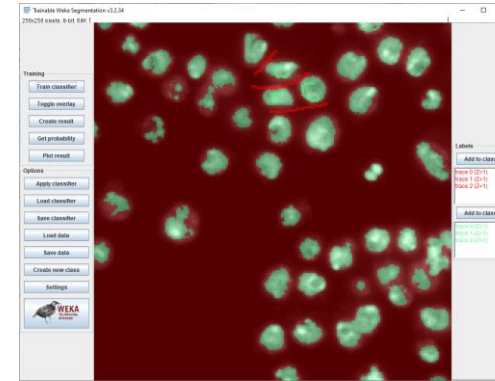
- Might there be a technology for optimization which combination of images can be used to get the best segmentation result?



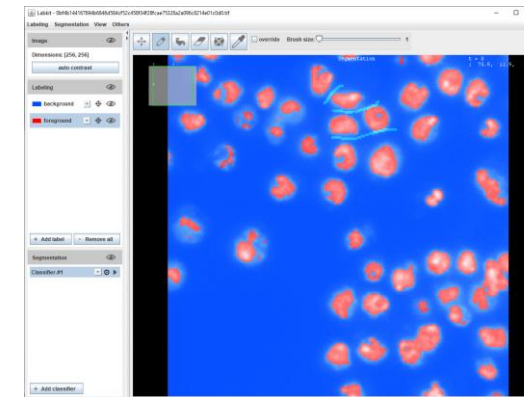
- A research field in computer science
- Finds more and more applications, also in life sciences.

Artificial intelligence

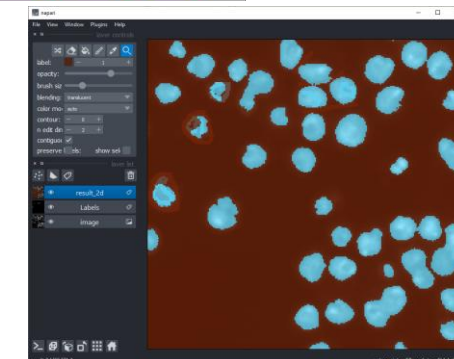
Machine learning



Trainable Weka Segmentation
<https://imagej.net/plugins/tws/>

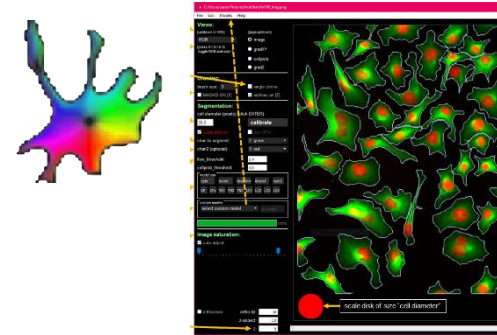
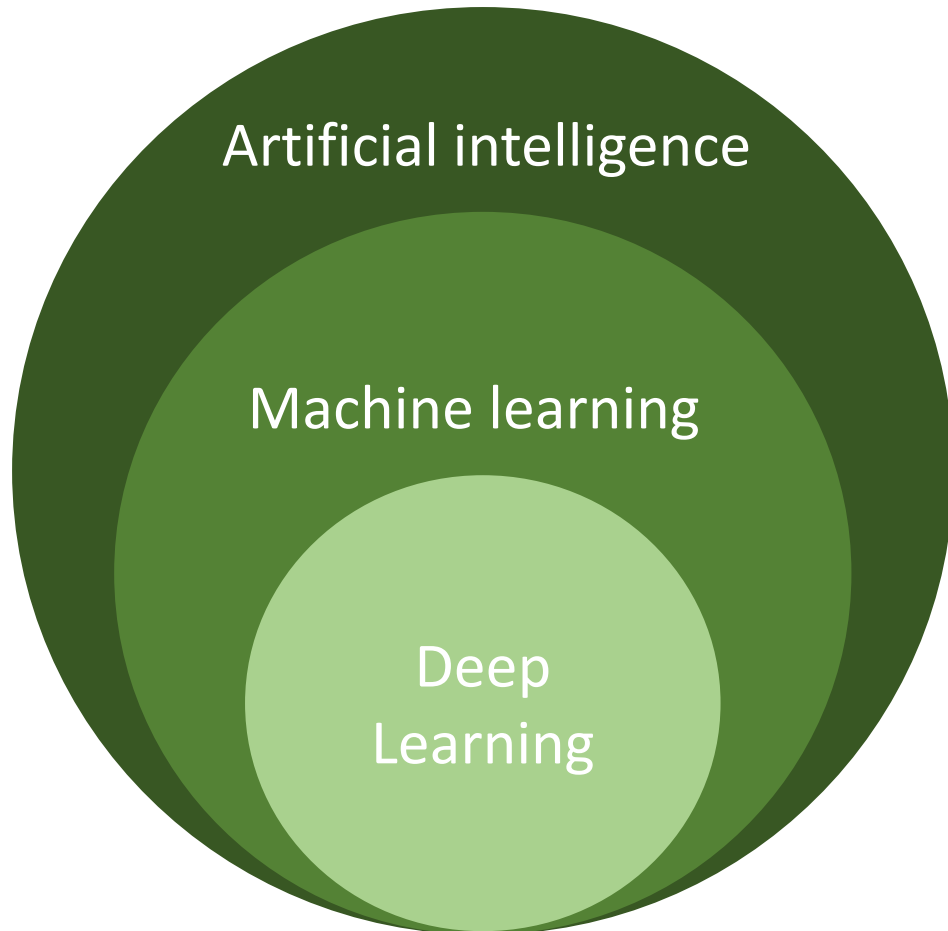


LabKit
<https://imagej.net/plugins/labkit/>

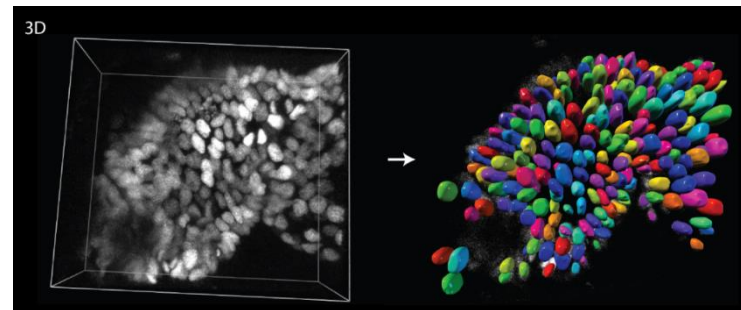


Python /
scikit-learn /
napari /
apoc

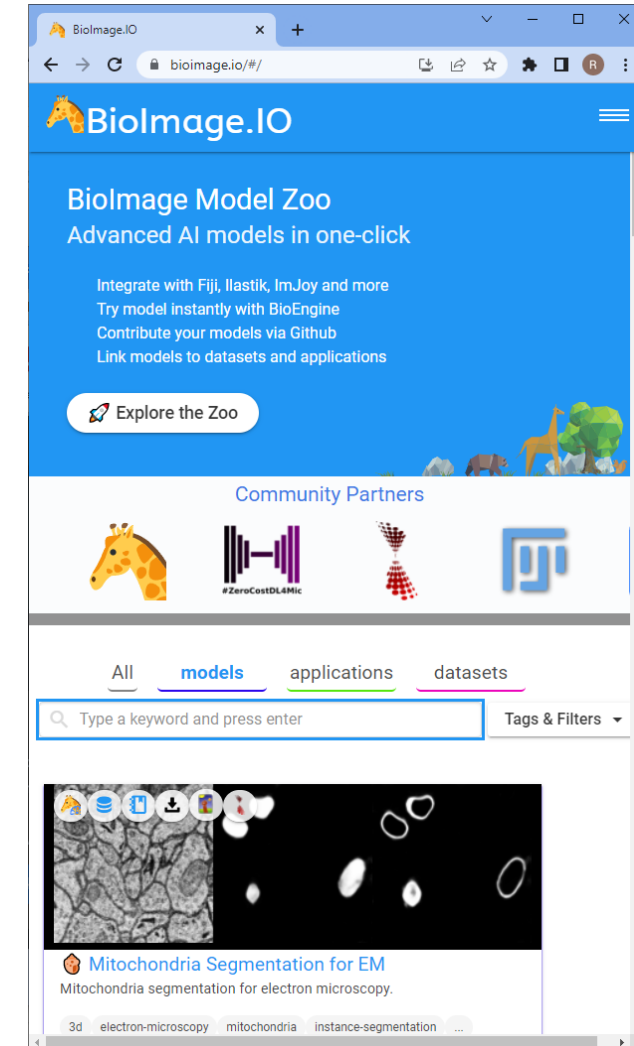
- A research field in computer science
- Finds more and more applications, also in life sciences.



www.cellpose.org/

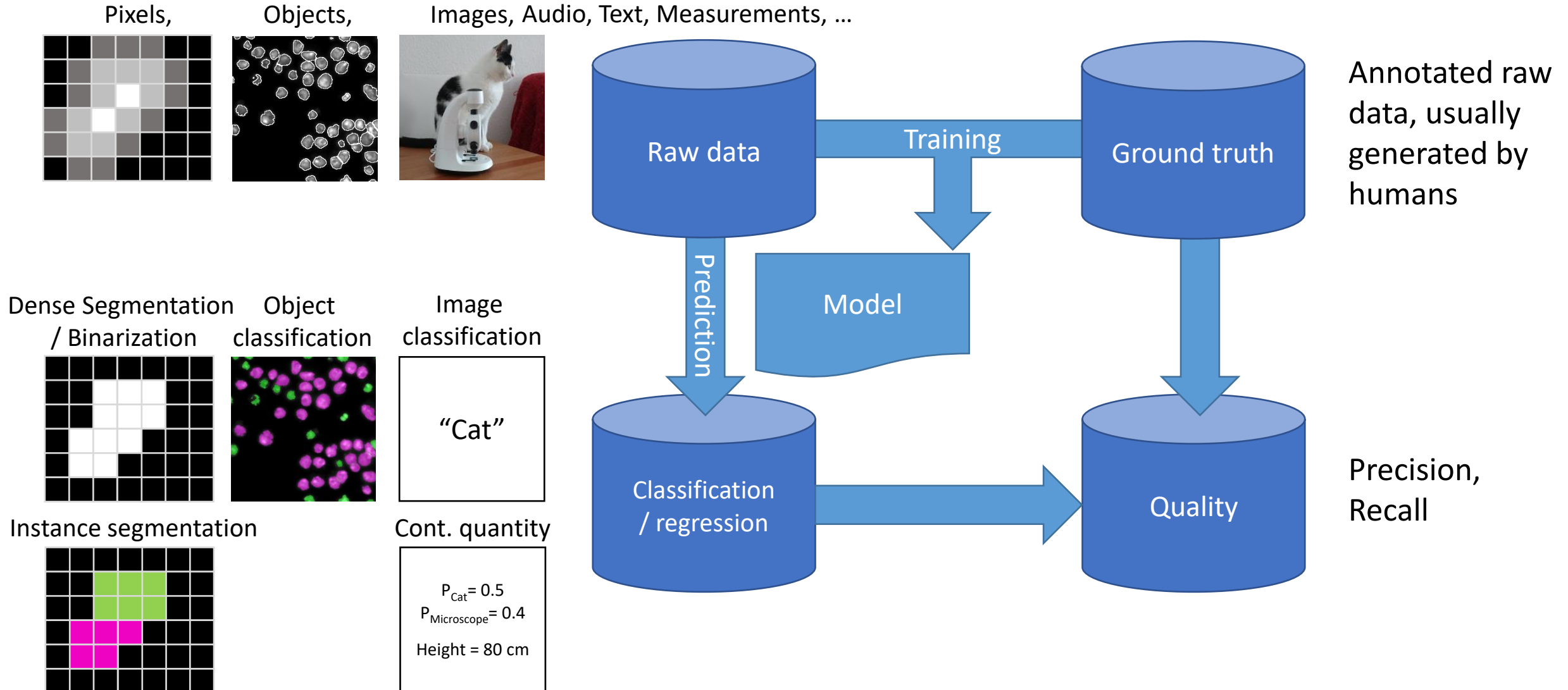


<https://github.com/stardist/stardist>

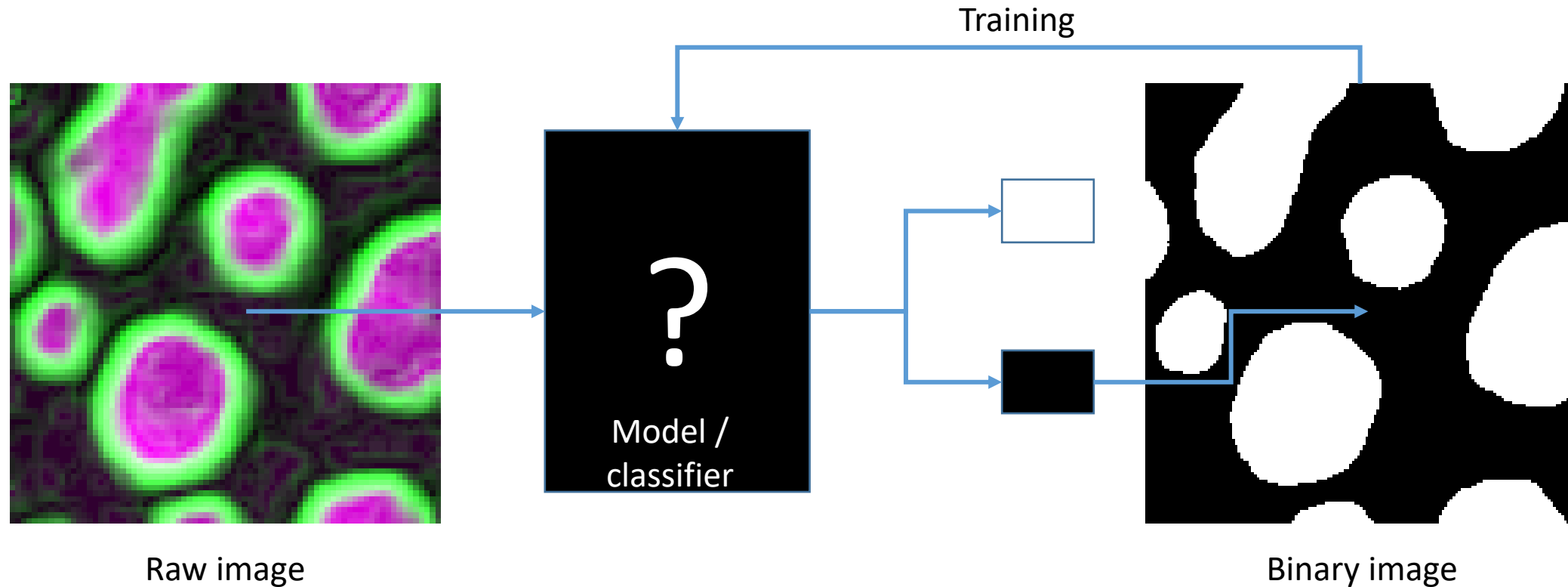


<https://bioimage.io/>

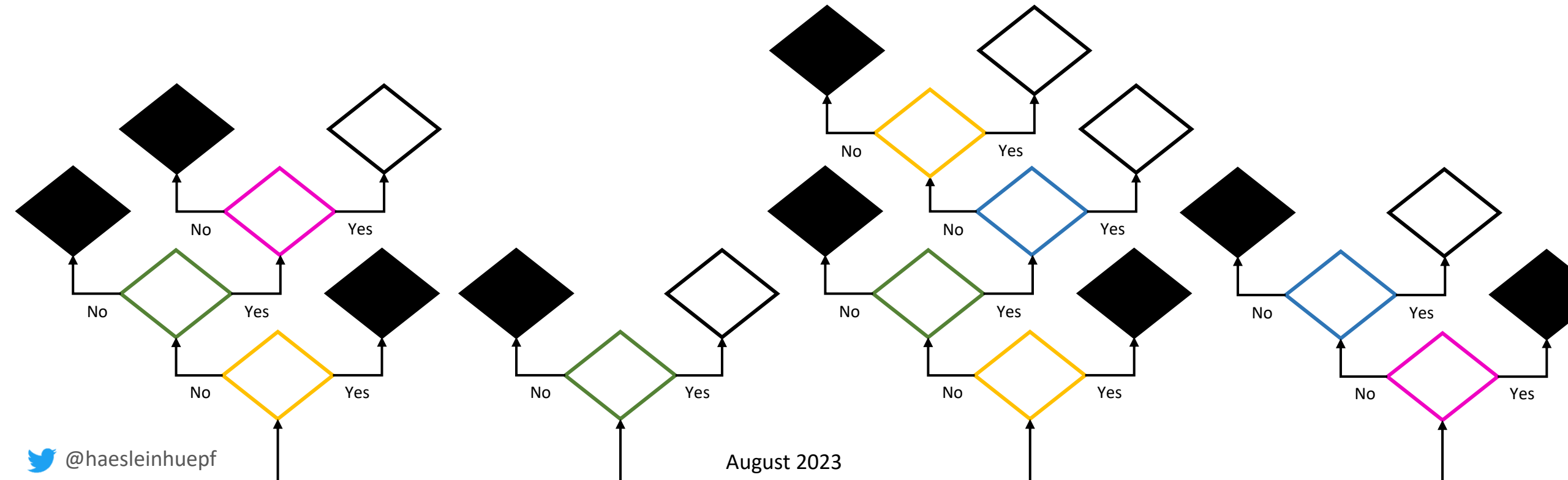
- Automatic construction of predictive models from given data



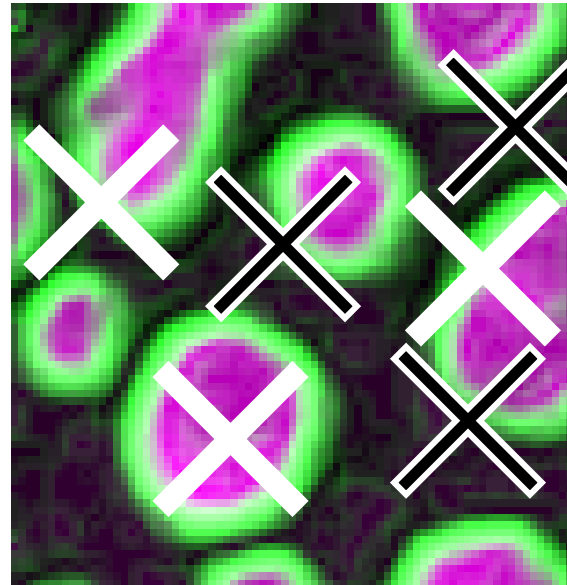
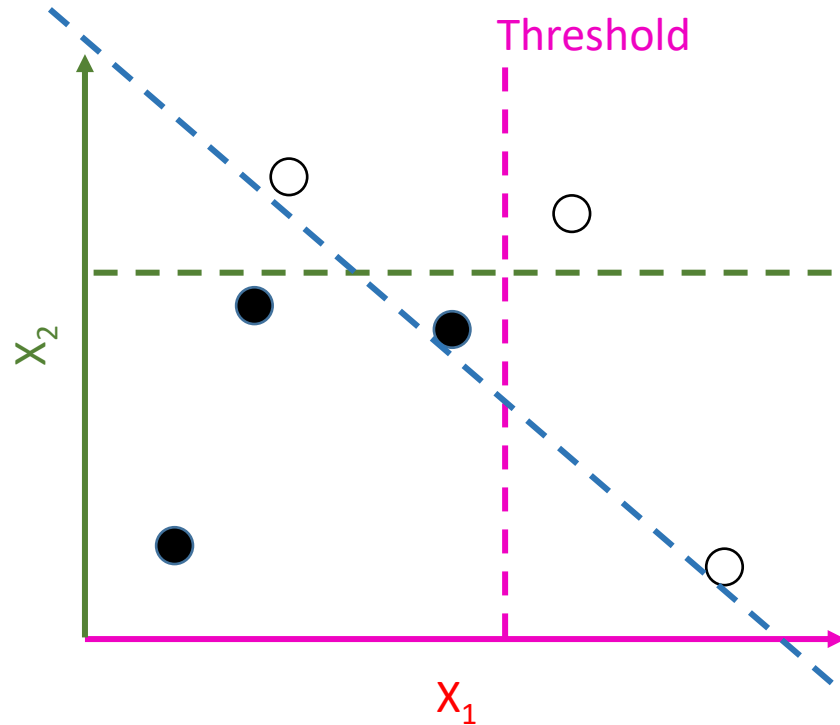
- *Supervised* machine learning: We give the computer some ground truth to learn from
- The computer derives a *model* or a *classifier* which can judge if a pixel should be foreground (white) or background (black)
- Example: Binary classifier



- Decision trees are classifiers, they decide if a pixel should be white or black
- Random decision trees are randomly initialized, afterwards evaluated and selected
- Random forests consist of many random decision trees
- Example: Random forest of binary decision trees



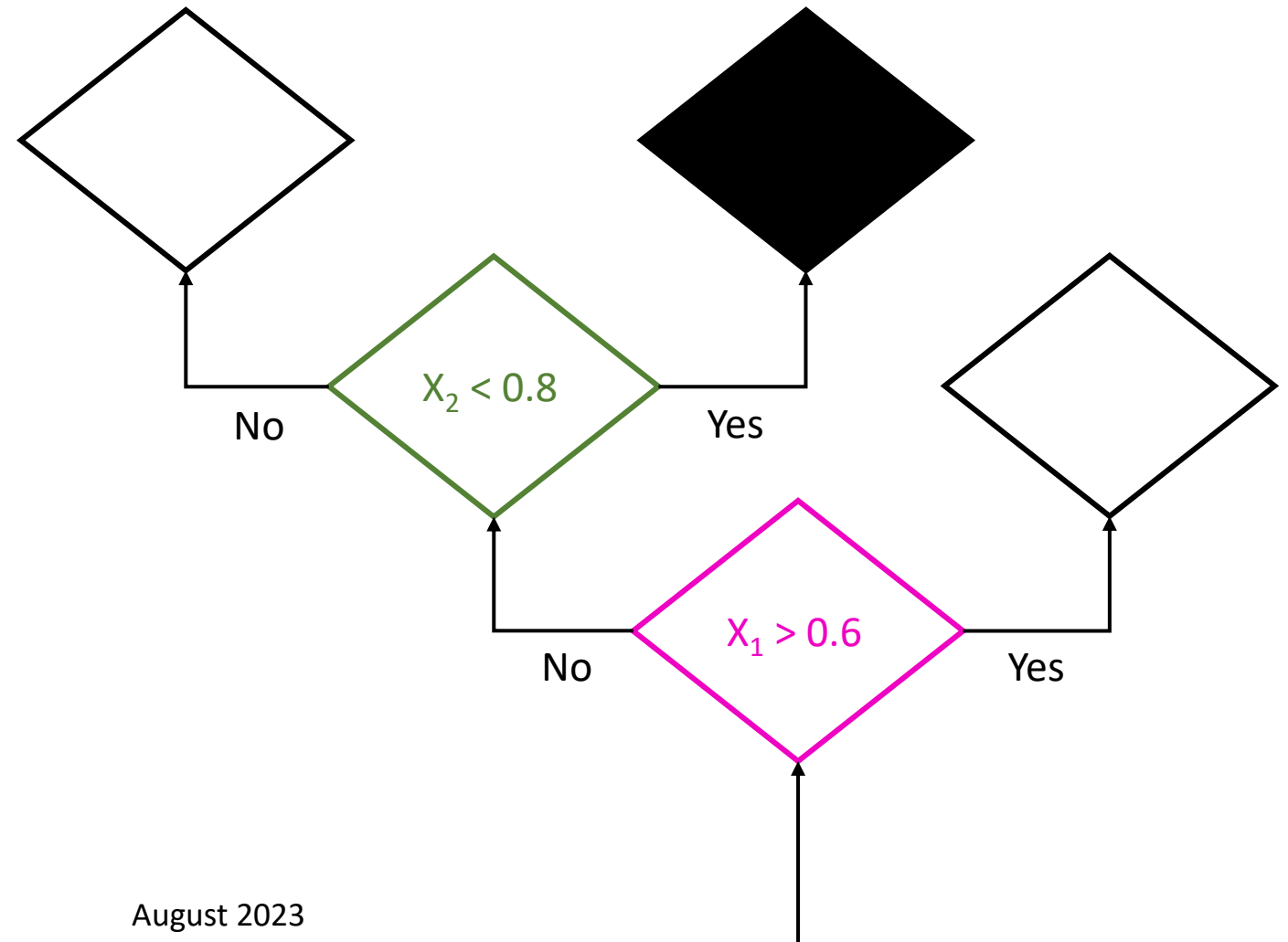
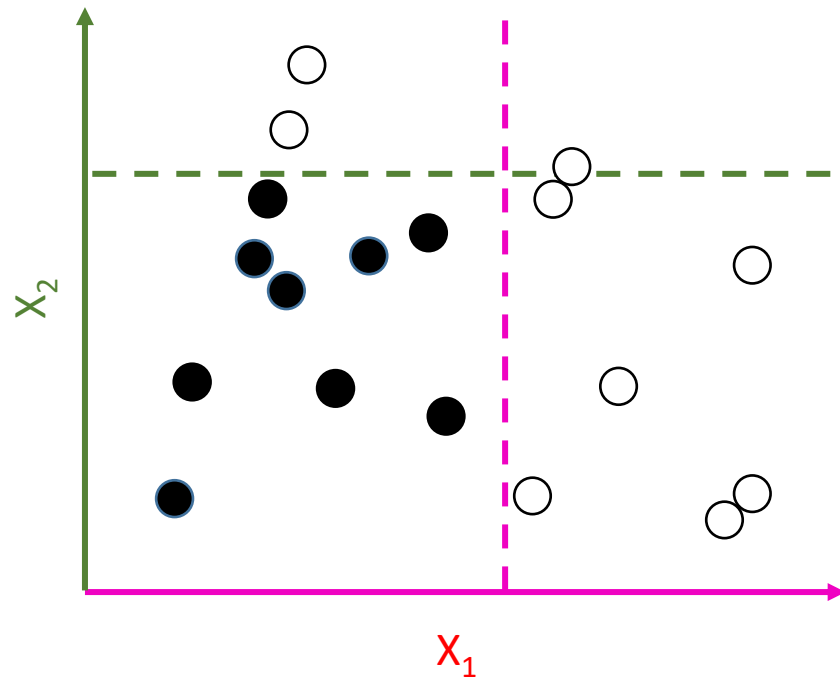
- For efficient processing, we randomly *sample* our data set
 - Individual pixels, their intensity and their classification



Note: You cannot use a single threshold to make the decision correctly

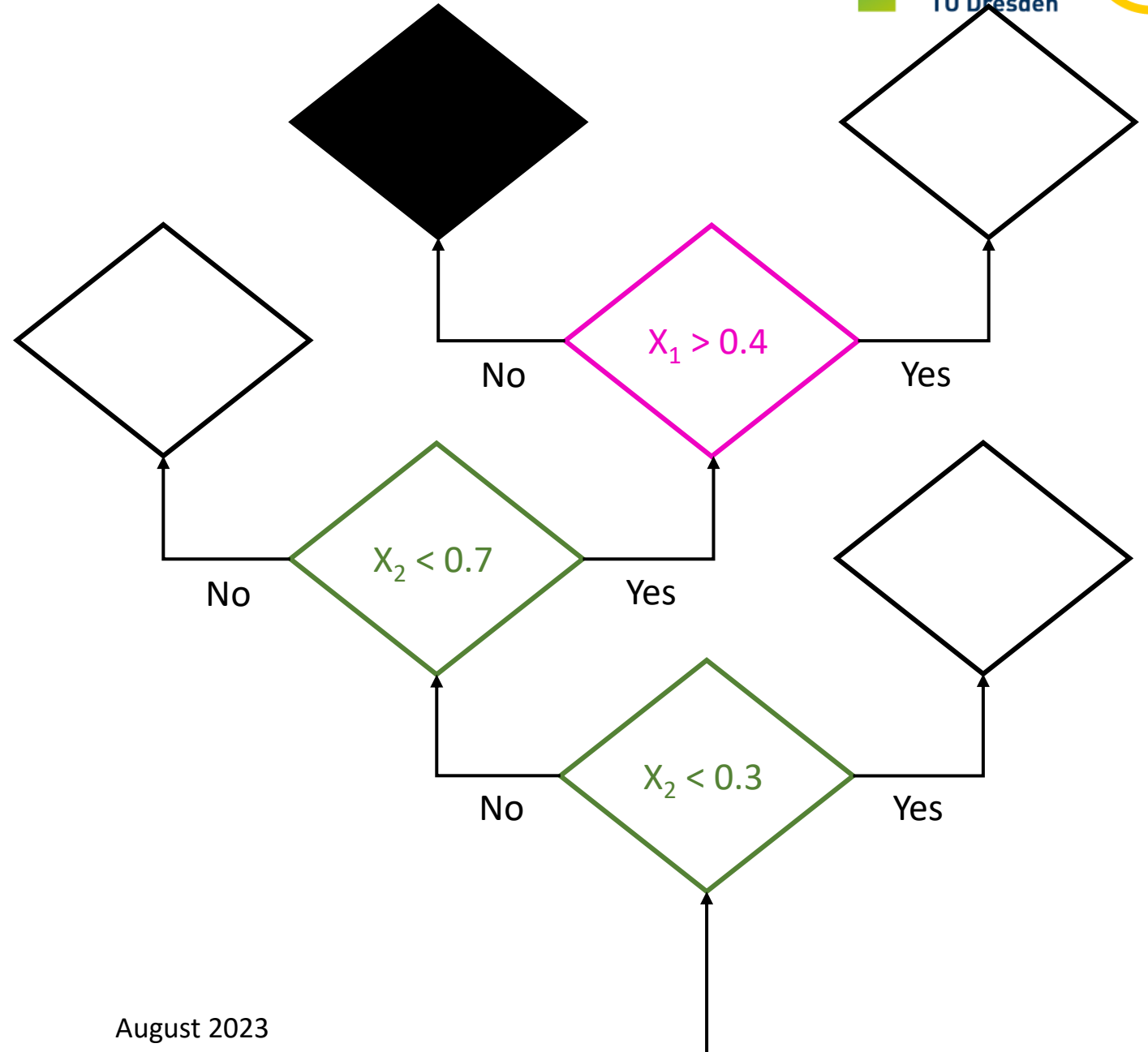
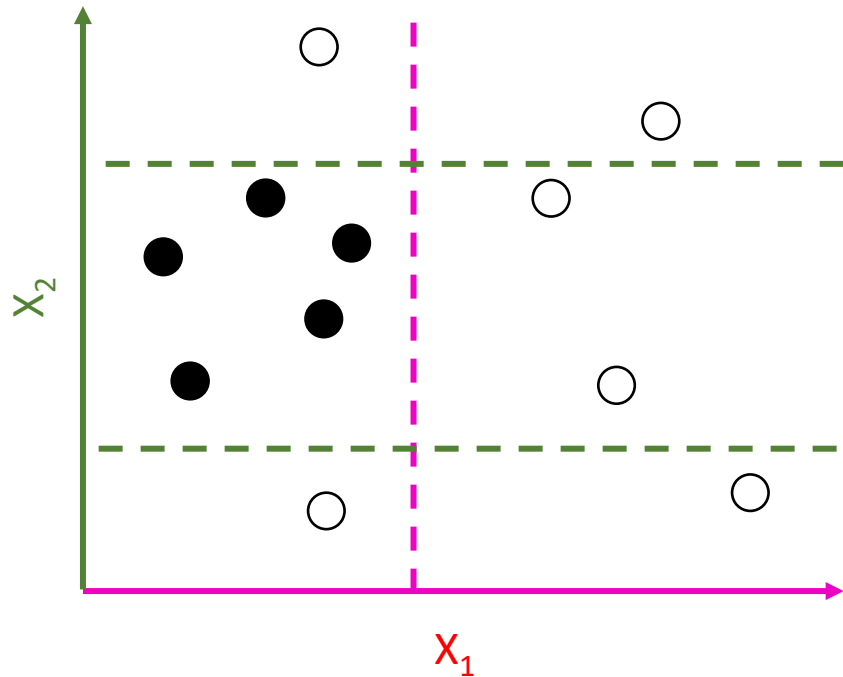
Deriving random decision trees

- Decision trees combine several thresholds on several parameters

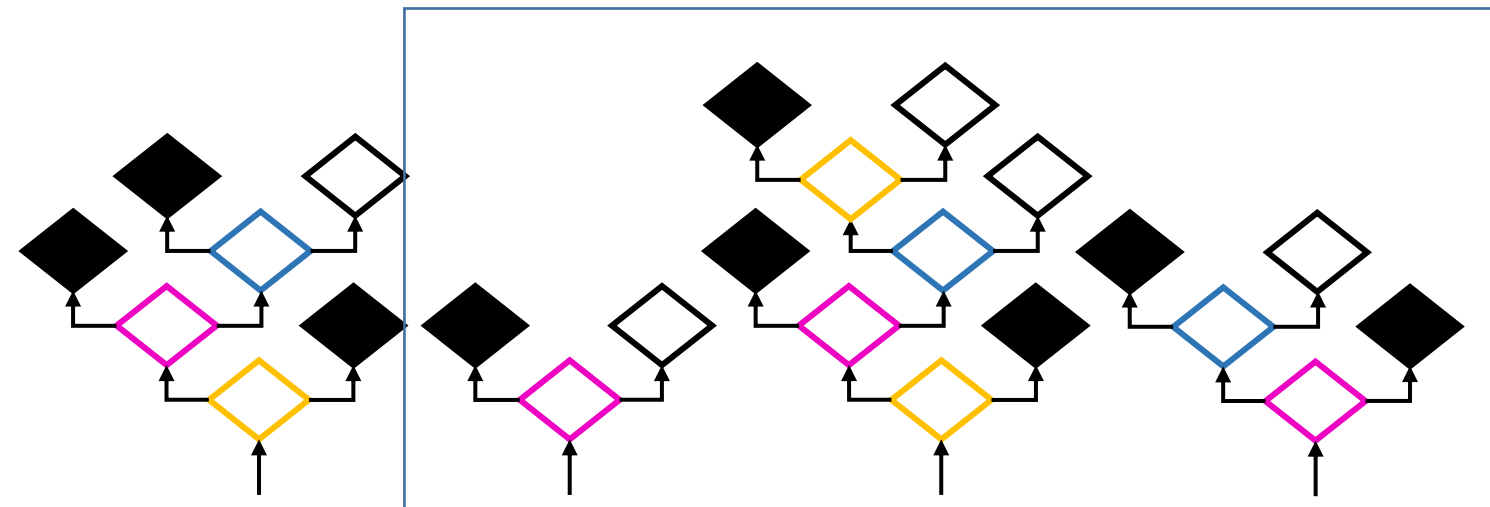
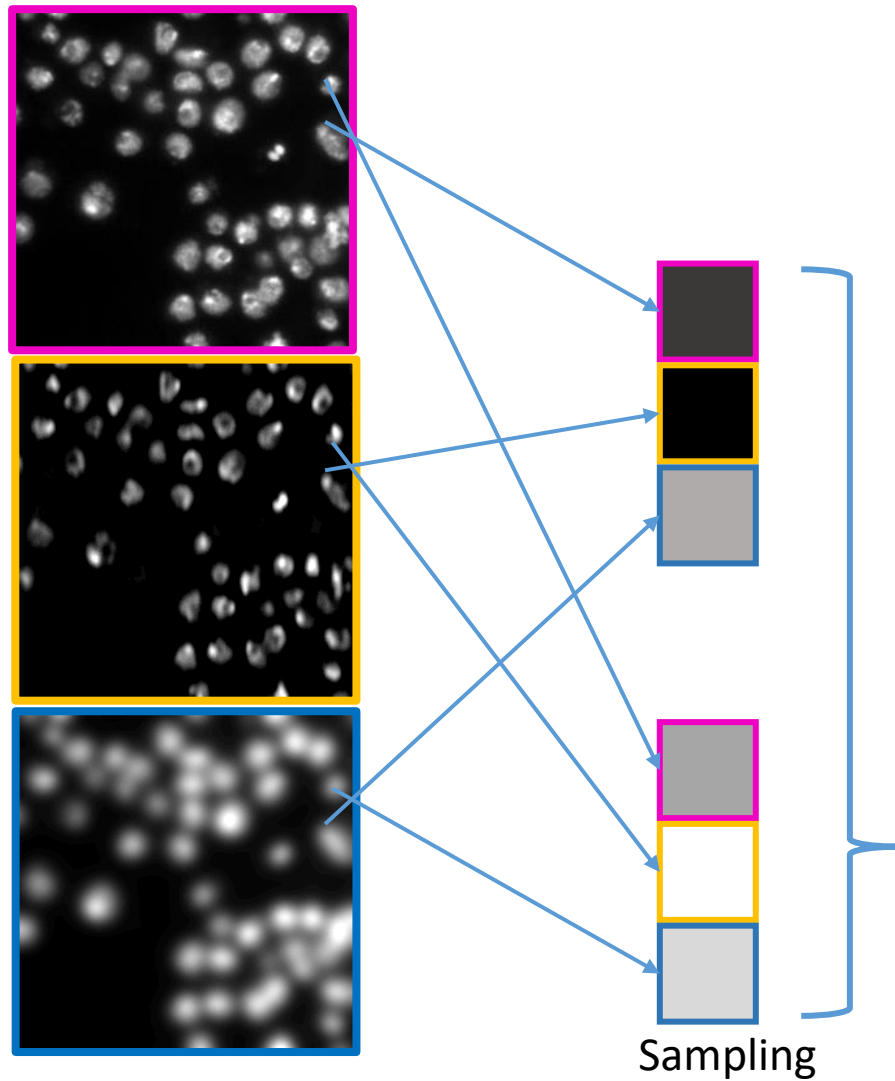


Deriving random decision trees

- Depending on sampling, the decision trees are different

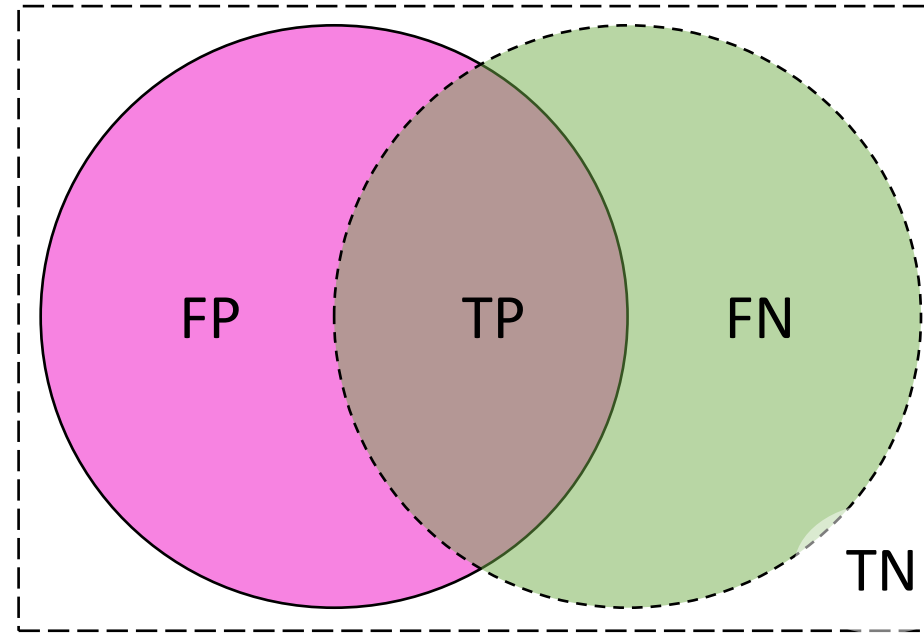




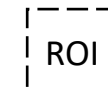




- By comparing performance of individual decision trees, good ones can be selected, bad ones excluded.



Segmentation quality estimation

- In general
 - Define what's positive and what's negative.
 - Compare with a reference to figure out what was true and false
- Welcome to the Theory of Sets



-  Prediction A
-  Reference B (ground truth)
-  Region of interest
-  True-positive
-  False-negative
-  False-positive
-  True-negative

Overlap
(a.k.a. Jaccard index) $\frac{TP}{TP + FN + FP}$

How much do A and B overlap?

Precision $\frac{TP}{TP + FP}$

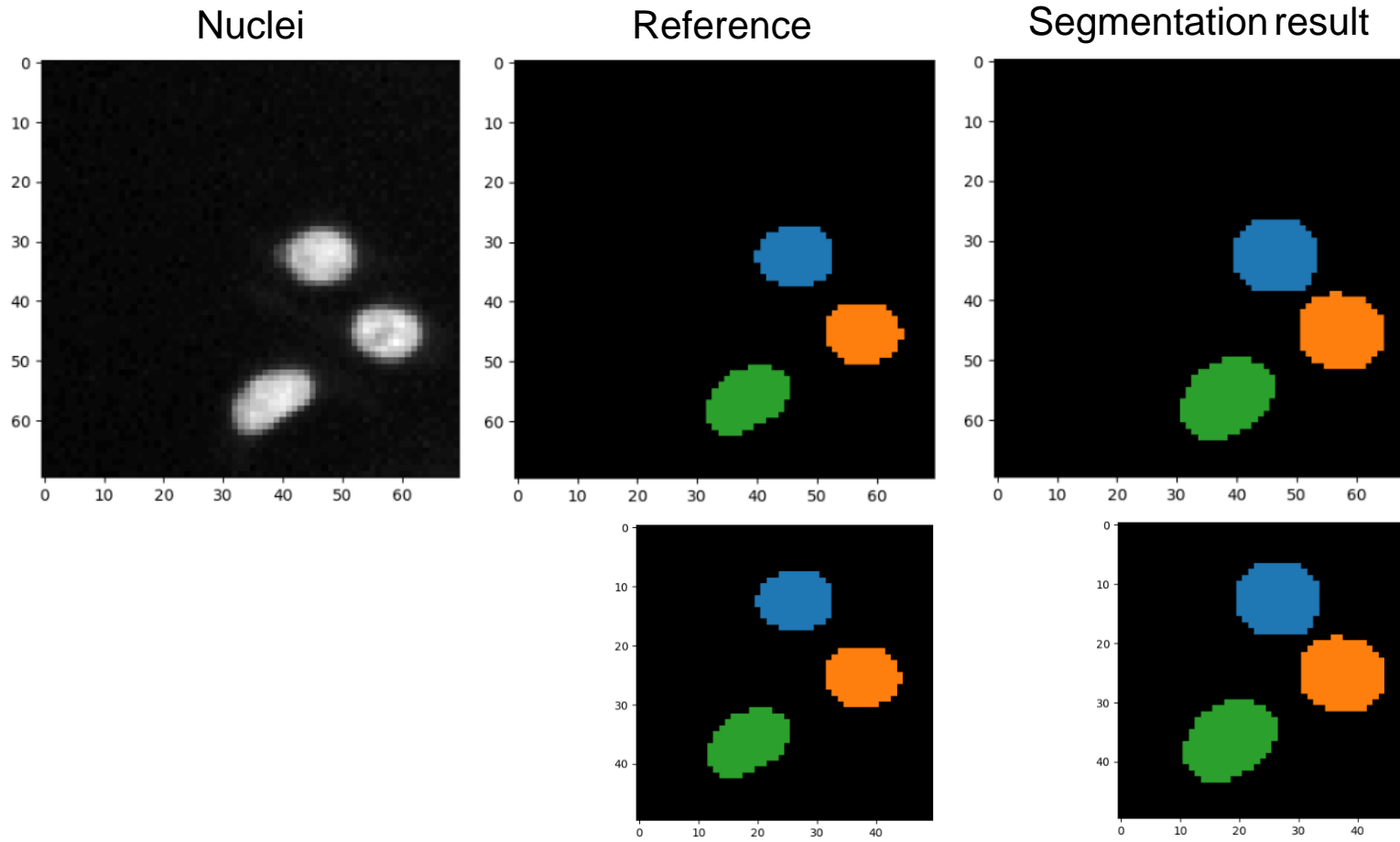
What fraction of points that were predicted as positives were really positive?

Recall
(a.k.a. sensitivity) $\frac{TP}{TP + FN}$

What fraction of positives points were predicted as positives?

Accuracy versus Jaccard Index (IoU)

- Side-effects of image size and number of nuclei



$$A = \frac{TP + TN}{FN + FP + TP + TN}$$

$$J = \frac{TP}{FN + FP + TP}$$

Accuracy: 0.97

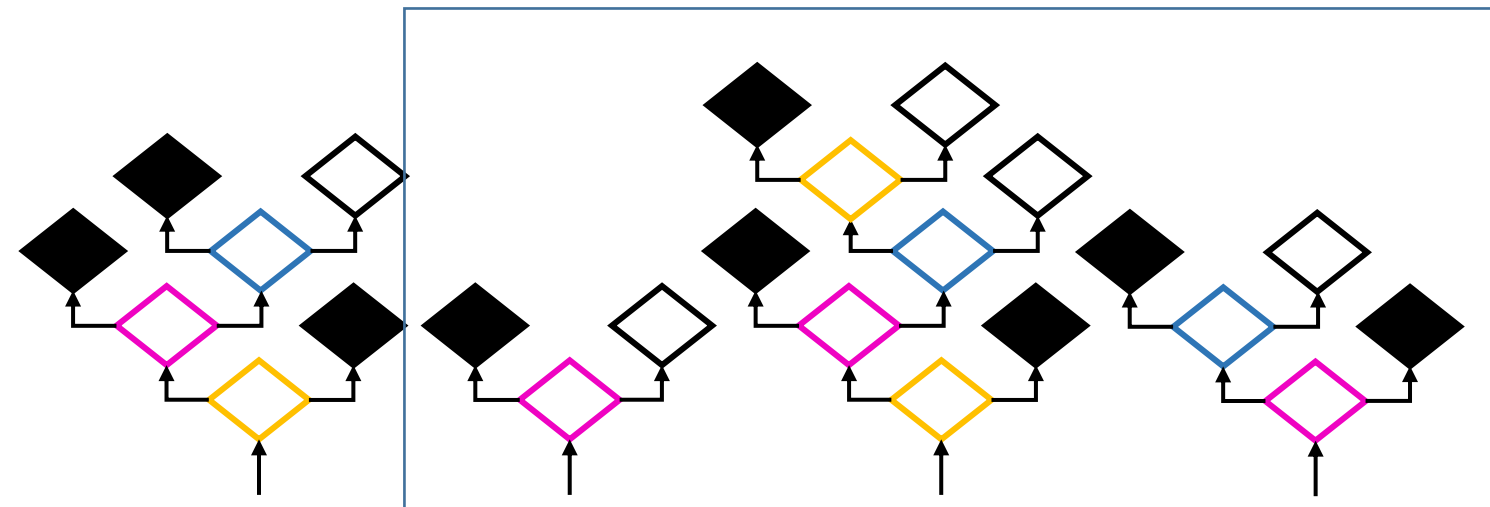
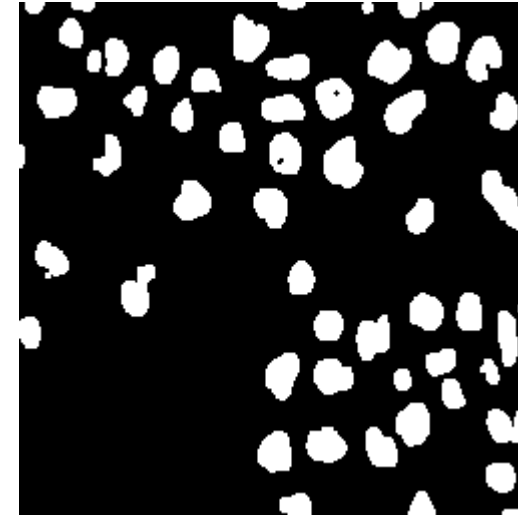
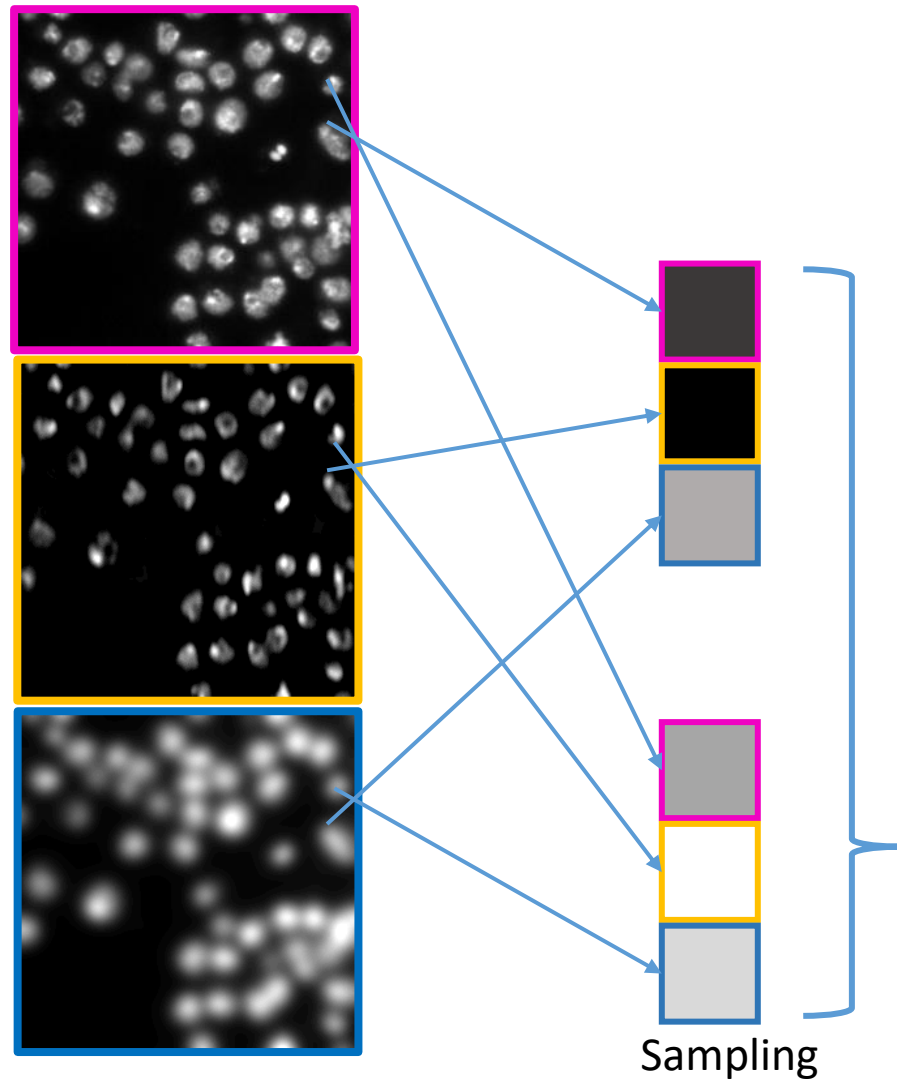
Jaccard Index: 0.73

Accuracy decreases because
there are less correct black
pixels (TN)

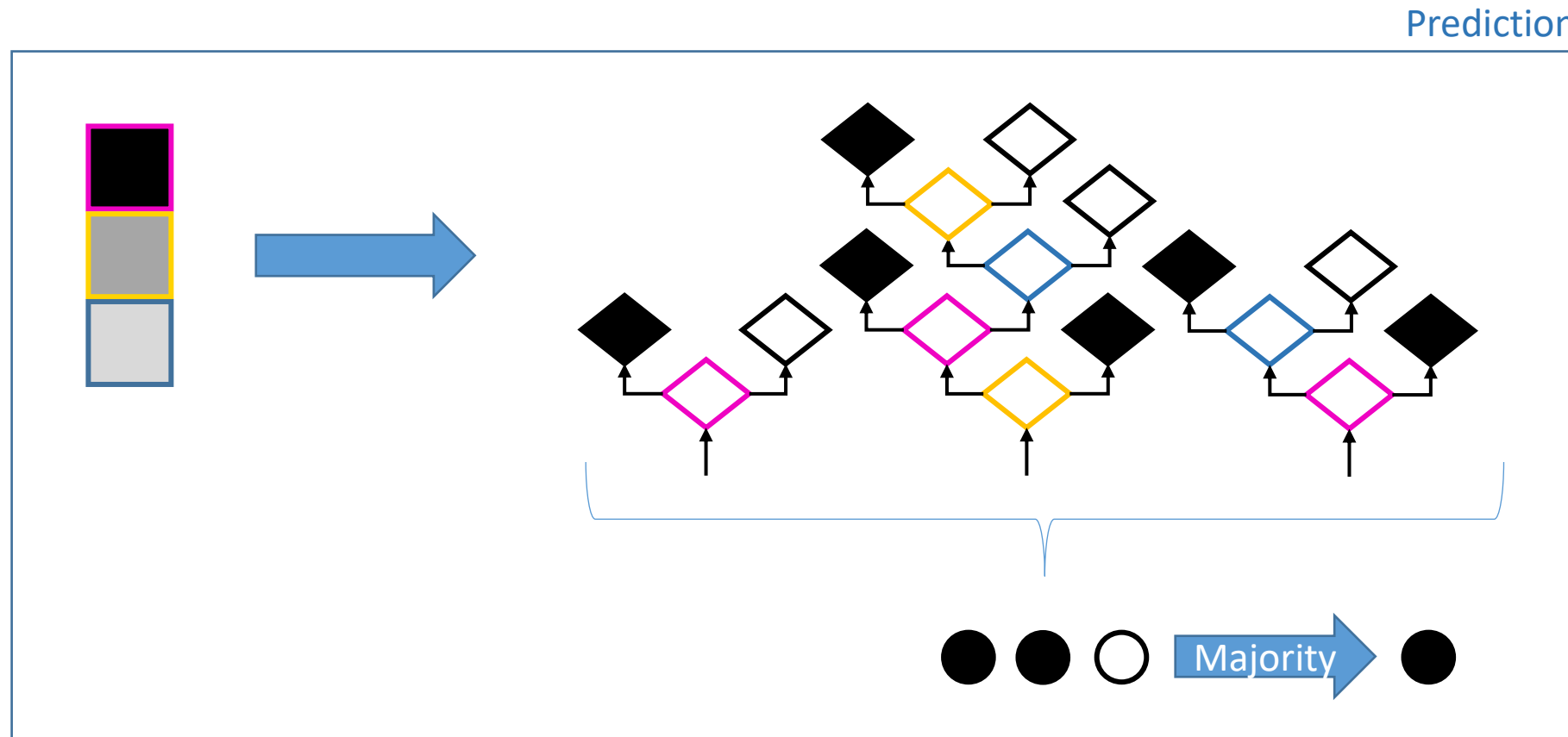
Accuracy: 0.95

Jaccard Index: 0.73

- By comparing performance of individual decision trees, good ones can be selected, bad ones excluded.

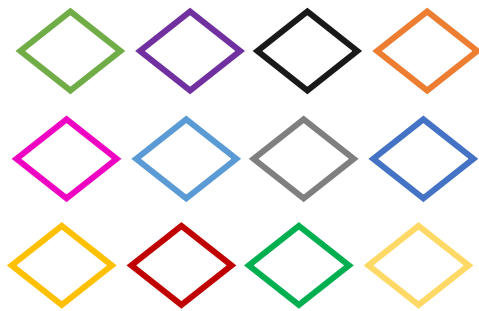


- Combination of individual tree decisions by voting or max / mean



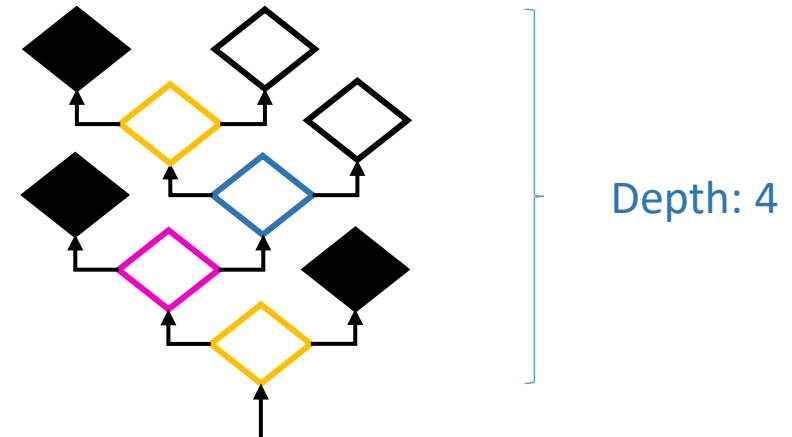
- Typical numbers for pixel classifiers in microscopy

Available features: > 20

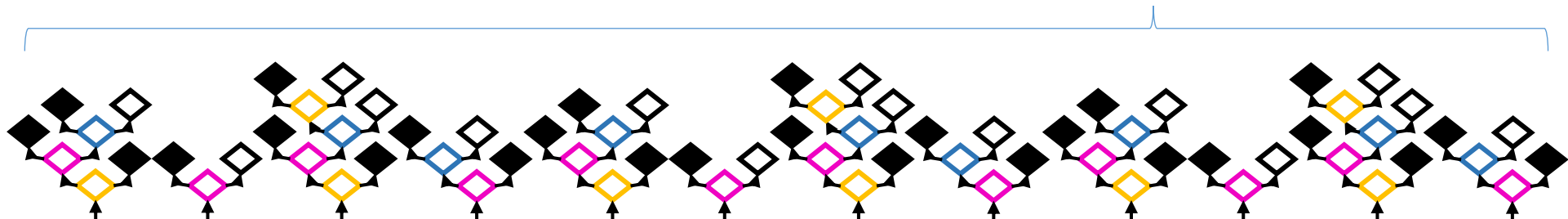


- Gaussian blur image
- DoG image
- LoG image
- Hessian
-

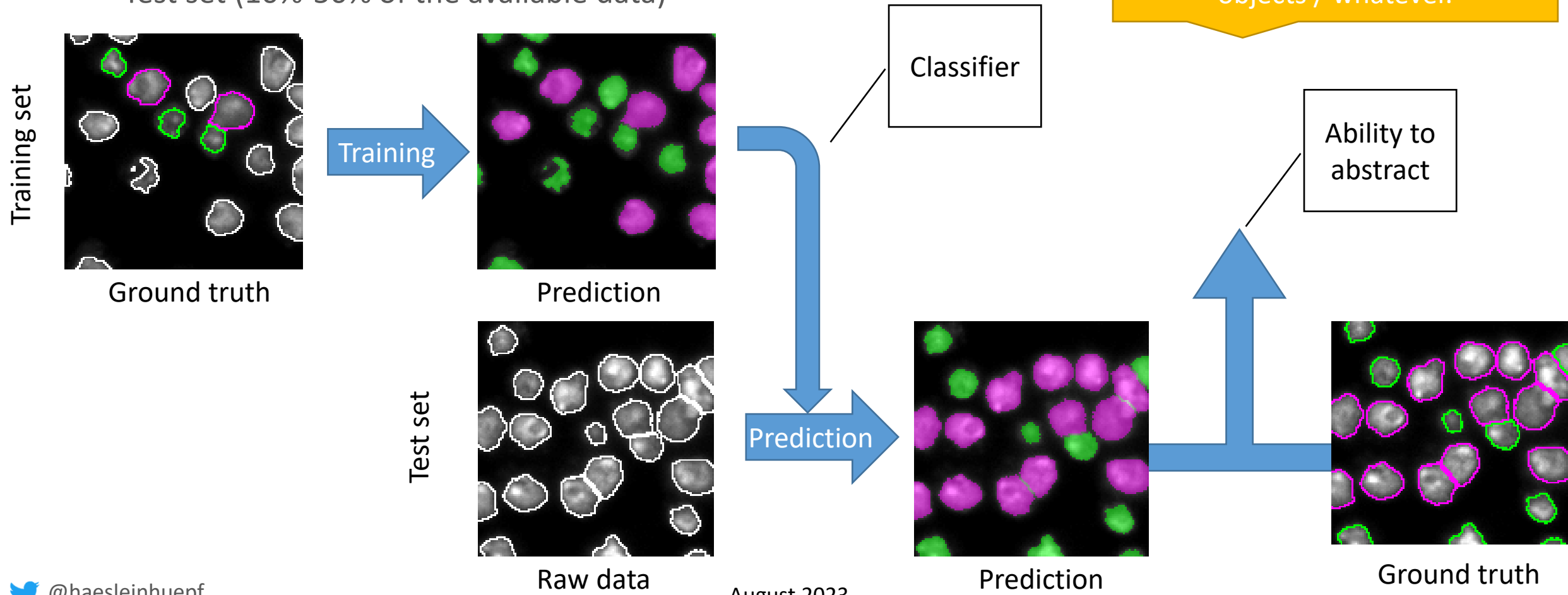
Selected features: $\leq \text{depth}$



Number of trees: > 100

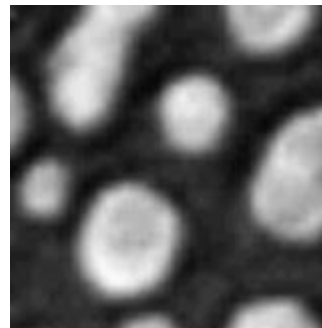
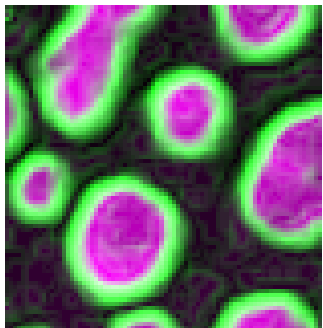
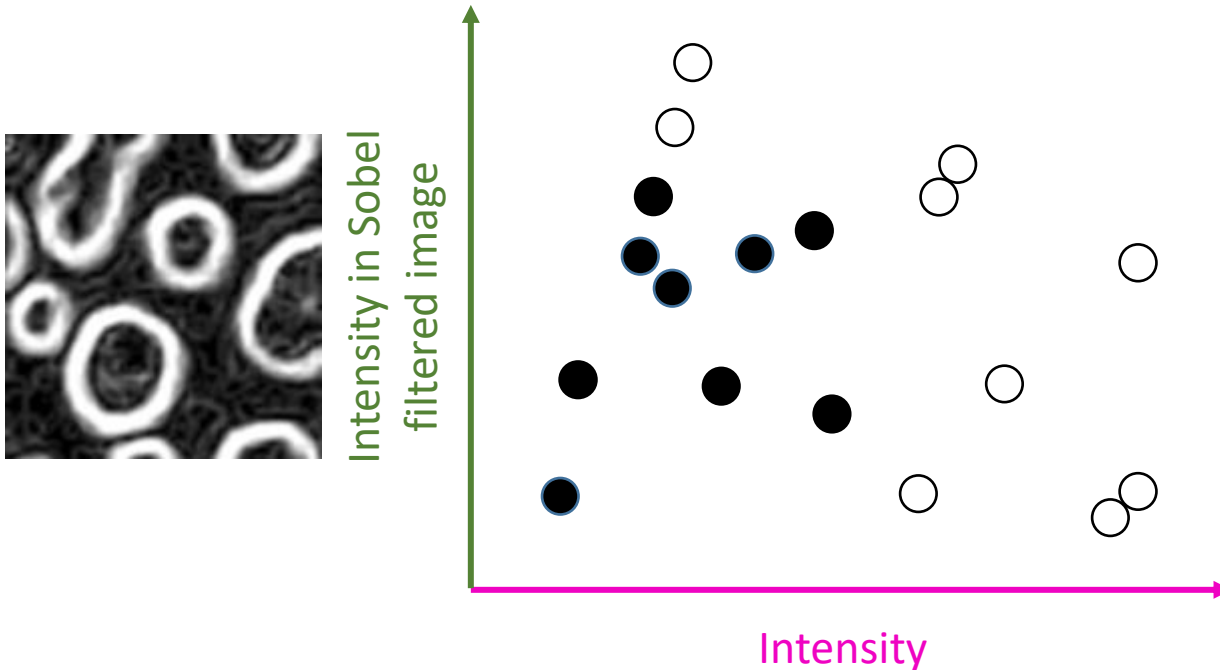


- A good classifier is trained on a hand full of datasets and works on thousands similarly well.
- In order to assess that, we split the ground truth into two set
 - Training set (50%-90% of the available data)
 - Test set (10%-50% of the available data)

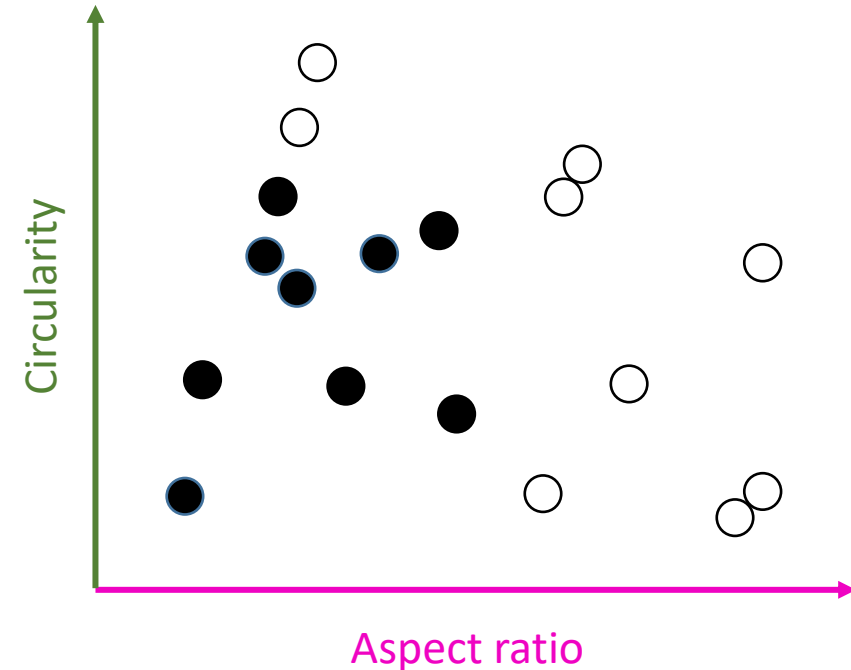


- What if we exchange pixel features with object features?

Pixel classification



Object classification



- The algorithms work the same but with different
 - Features
 - Number of features
 - Tree / forest parameters
 - Selection criteria

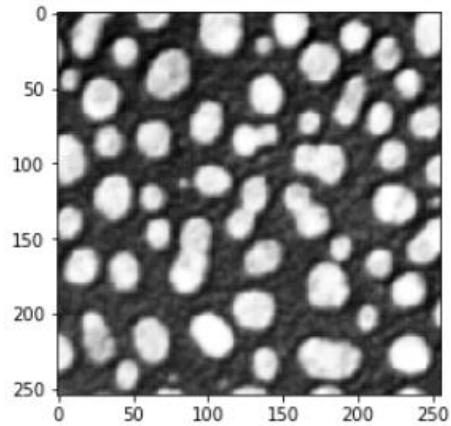
Accelerated pixel and object classification (APOC)

Robert Haase

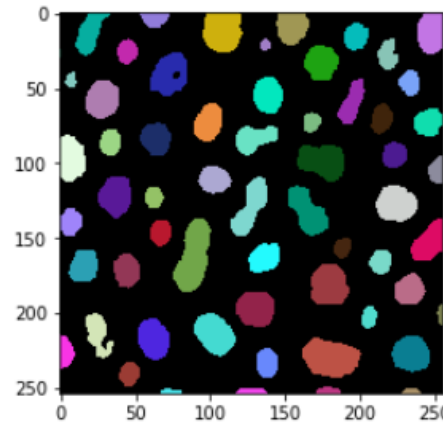
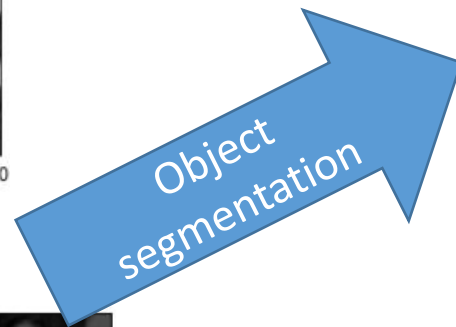
August 2023

Accelerated pixel and object classification

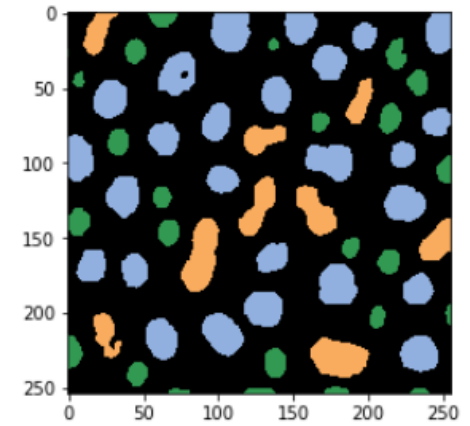
- APOC is a python library that makes use of OpenCL-compatible Graphics Cards to accelerate pixel and object classification



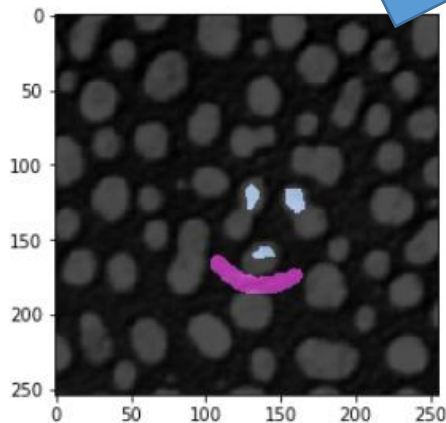
Raw image



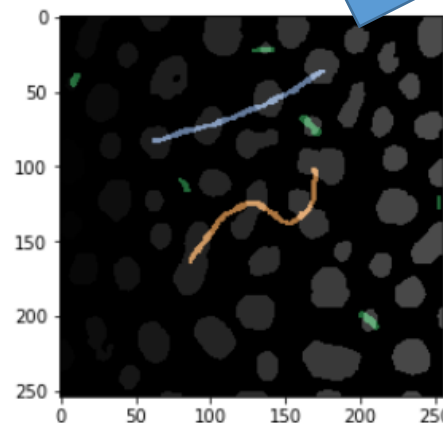
Object label image



Class label image

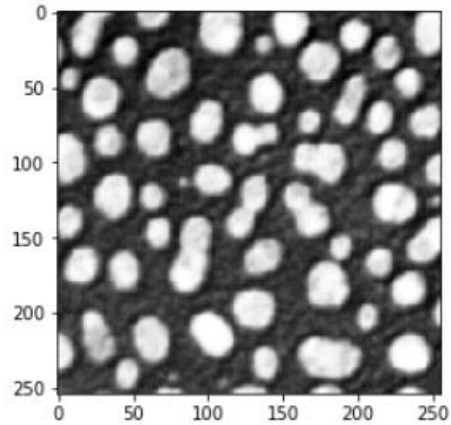


Pixel annotation

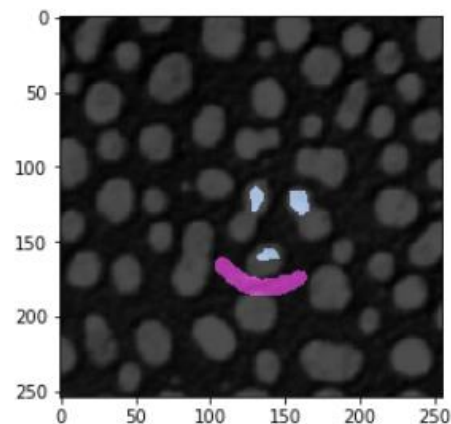


Object annotation

- Pixel classification + connected component labeling



Raw image



Pixel annotation

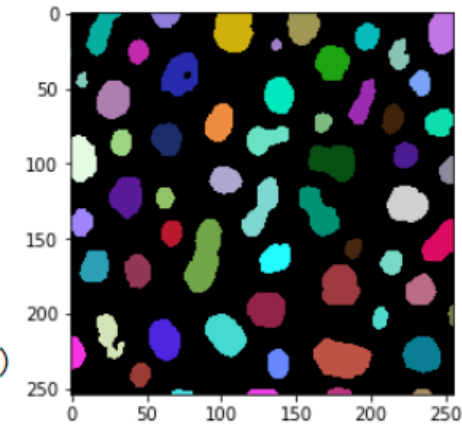
```
# define features
features = "gaussian_blur=1 gaussian_blur=5 sobel_of_gaussian_blur=1"

# this is where the model will be saved
cl_filename = 'my_object_segmenter.cl'

# delete classifier in case the file exists already
apoc.erase_classifier(cl_filename)

# train classifier
clf = apoc.ObjectSegmenter(opencl_filename=cl_filename, positive_class_identifier=2)
clf.train(features, manual_annotations, image)

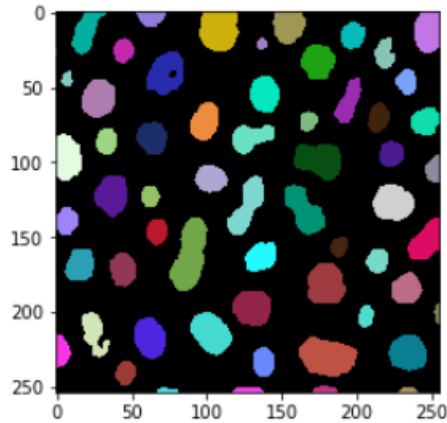
segmentation_result = clf.predict(features=features, image=image)
cle.imshow(segmentation_result, labels=True)
```



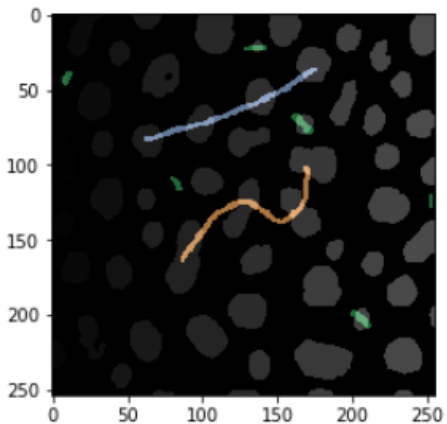
Object label image

Object segmentation

- Feature extraction + tabular classification



Object label image



Object annotation

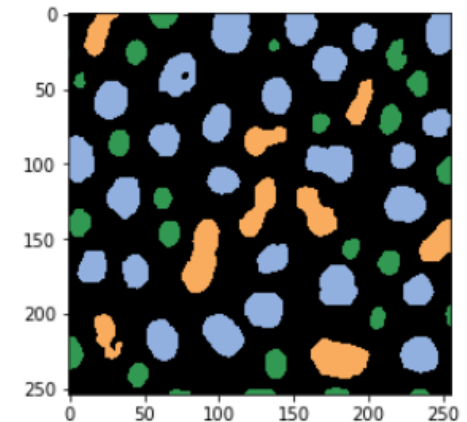
```
# for the classification we define size and shape as criteria  
features = 'area mean_max_distance_to_centroid_ratio'
```

```
# This is where the model will be saved  
cl_filename_object_classifier = "my_object_classifier.cl"
```

```
# delete classifier in case the file exists already  
apoc.erase_classifier(cl_filename_object_classifier)
```

```
# train the classifier  
classifier = apoc.ObjectClassifier(cl_filename_object_classifier)  
classifier.train(features, segmentation_result, annotation, image)
```

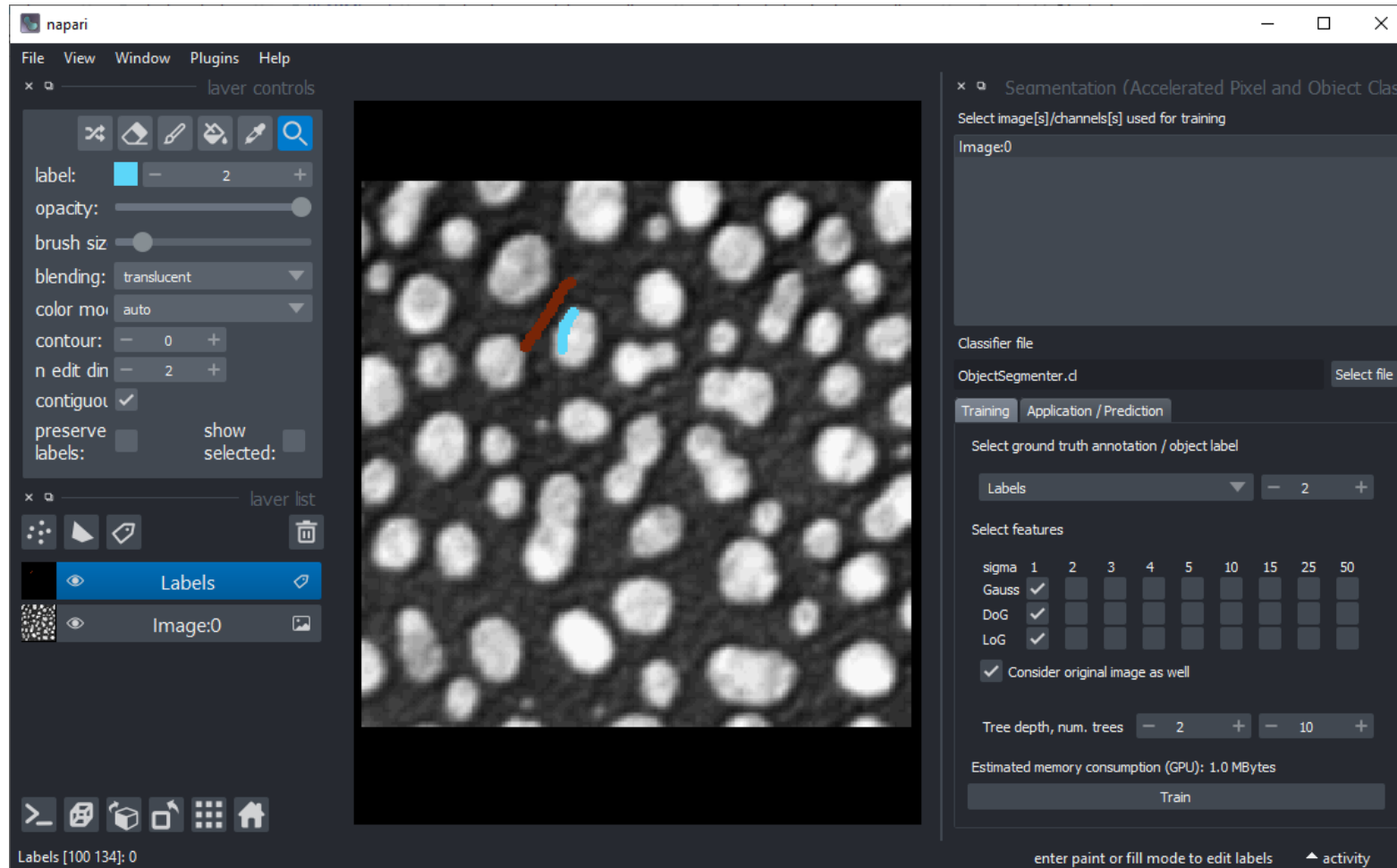
```
# determine object classification  
classification_result = classifier.predict(segmentation_result, image)  
cle.imshow(classification_result, labels=True)
```



Class label image

Object classification

- Object segmentation
- <https://github.com/haesleinhuepf/napari-accelerated-pixel-and-object-classification#object-and-semantic-segmentation>



Exercises

- Use Napari to segment objects

Interactive pixel classification and object segmentation in Napari

In this exercise we will train a [Random Forest Classifier](#) for pixel classification and convert the result in an instance segmentation. We will use the napari plugin [napari-accelerated-pixel-and-object-classification](#).

Getting started

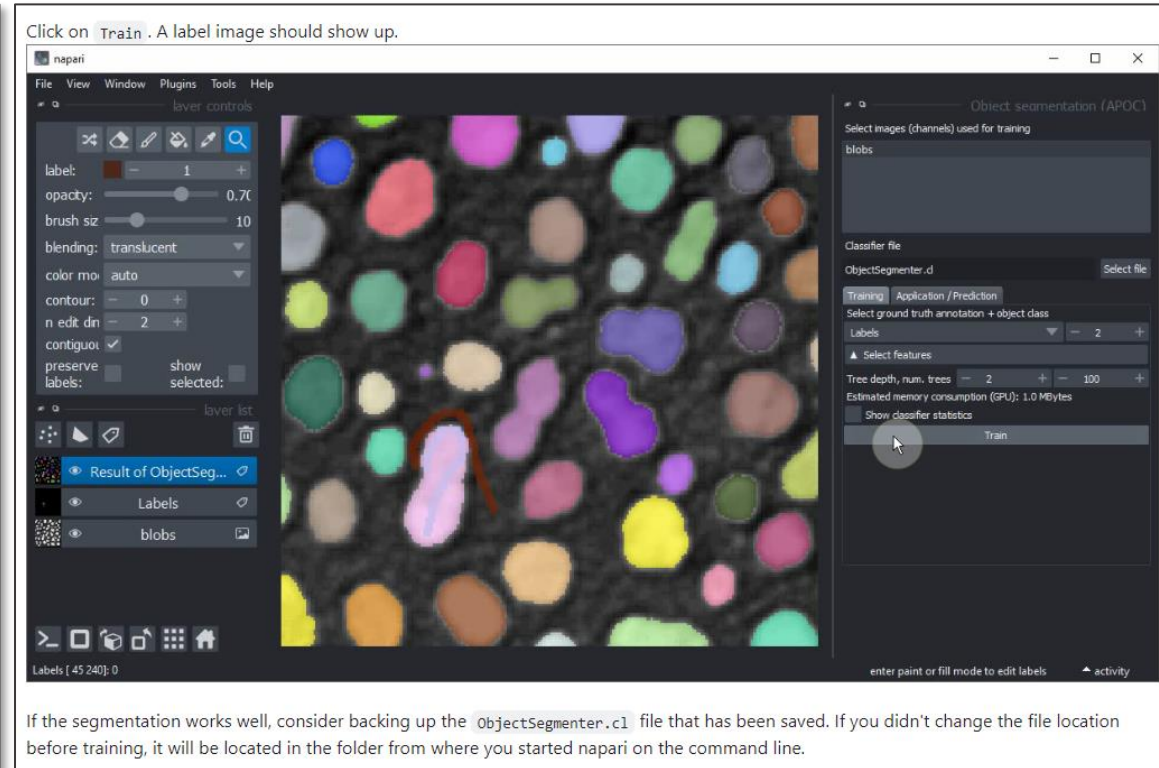
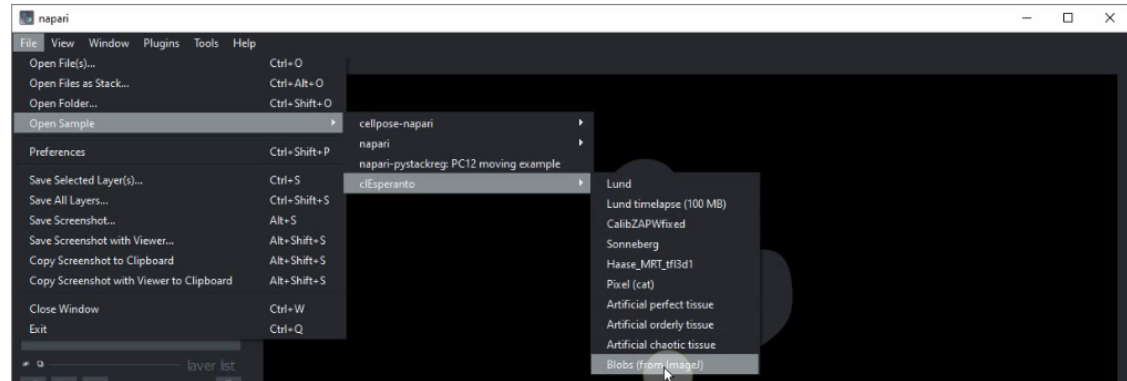
Open a terminal window and activate your conda environment:

```
conda activate devbio-napari-env
```

Afterwards, start up Napari:

```
napari
```

Load the "Blobs" example dataset from the menu `File > Open Sample > cIEsperanto > Blobs (from ImageJ)`



https://biapol.github.io/PoL-BioImage-Analysis-TS-Early-Career-Track/day2b_machine_learning_apoc/interactive_pixel_classification/intro.html

- Use Napari to group round and elongated objects

Interactive object classification in Napari

In this exercise we will train a [Random Forest Classifiers](#) for classifying segmented objects. We will use the napari plugin [napari-accelerated-pixel-and-object-classification](#).

Getting started

Open a terminal window and activate your conda environment:

```
conda activate devbio-napari-env
```

Afterwards, start up Napari:

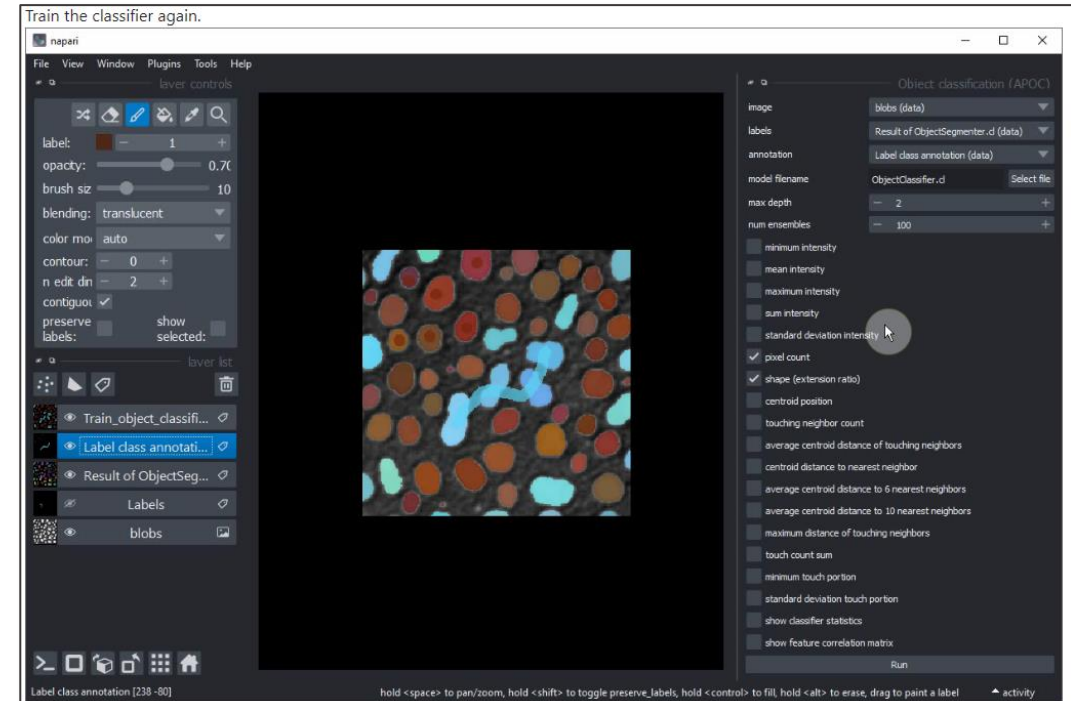
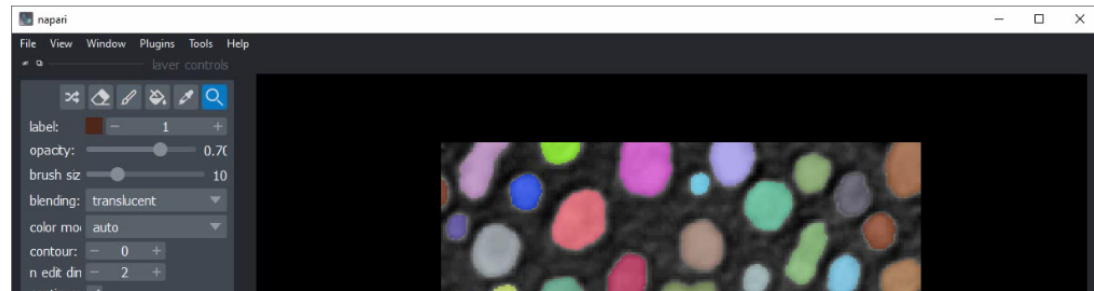
```
napari
```

Load the "Blobs" example dataset from the menu `File > Open Sample > clesperanto > Blobs (from ImageJ)`

We furthermore need a label image. You can create it using the pixel classifier trained earlier or using the menu `Tools > Segmentation / Labeling > Gauss-Otsu Labeling (clesperanto)`.

Object classification

Our starting point is a loaded image and a label image with segmented objects. The following procedure is also shown in [this video](#).



If you are happy with the trained classifier, copy the file to a safe place. When training the next classifier this one might be overwritten.

Extra exercise

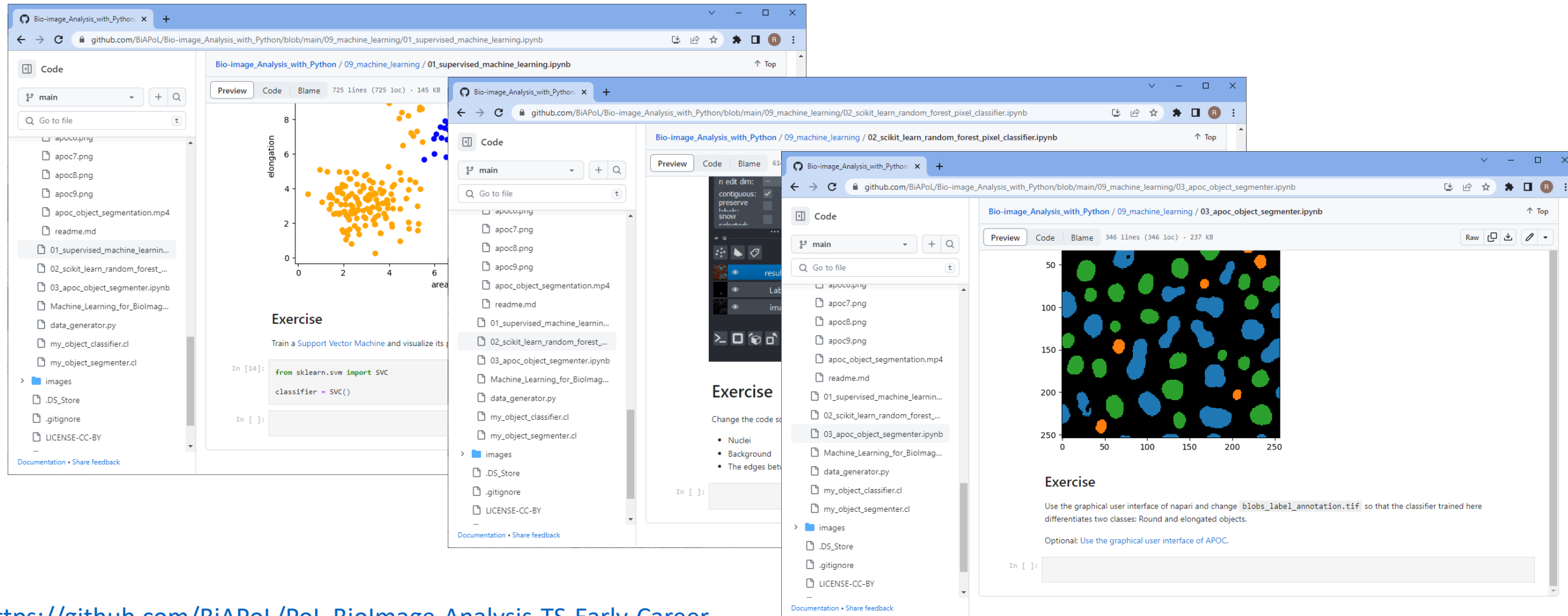
Retrain the classifier so that it can differentiate three different classes:

- Small round objects
- Large round objects
- Large elongated objects

https://biapol.github.io/PoL-BiolImage-Analysis-TS-Early-Career-Track/day2b_machine_learning_apoc/interactive_object_classification/intro.html

August 2023

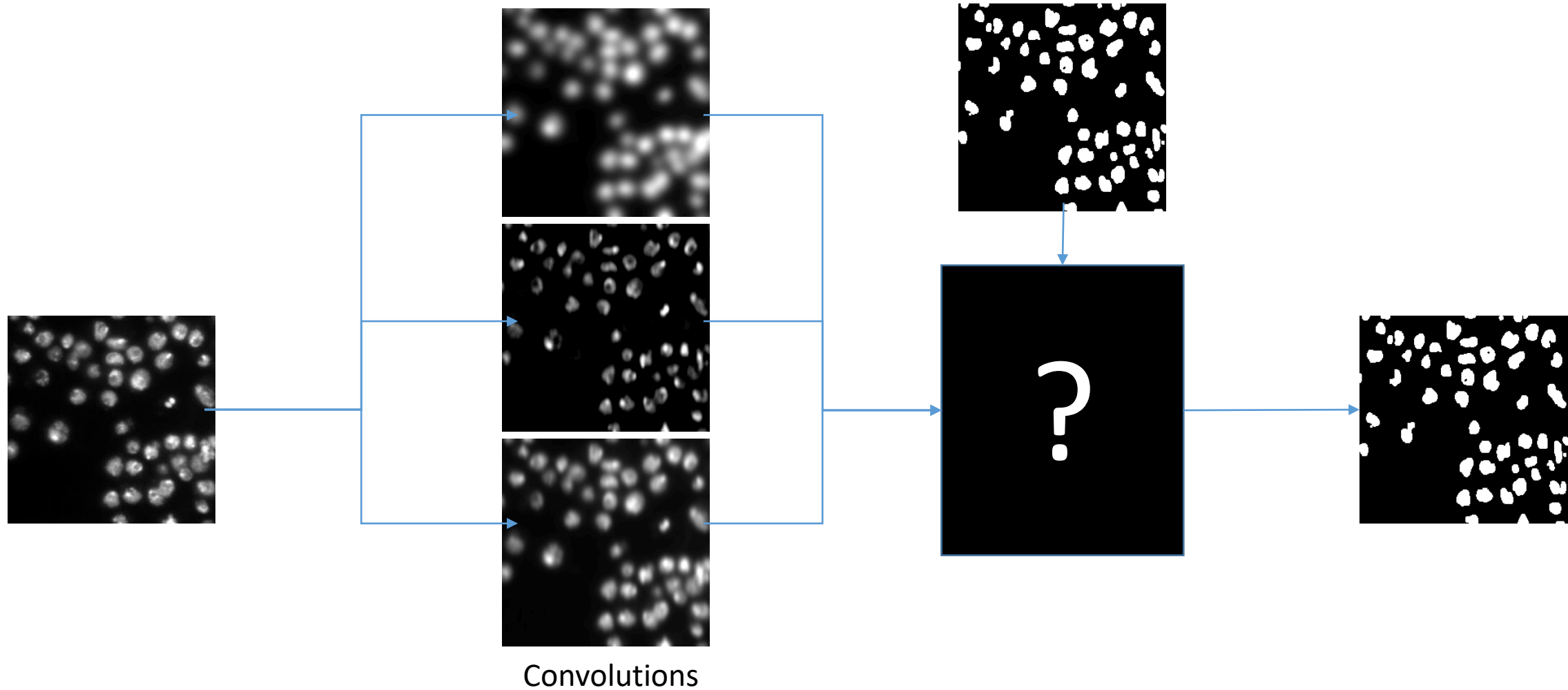
- Use scikit-learn and apoc in Jupyter Notebooks to train and apply Random Forest Classifiers and Support Vector Machines



https://github.com/BiAPoL/PoL-BioImage-Analysis-TS-Early-Career-Track/blob/main/docs/day2b_machine_learning_apoc/04_demo_object_segmenter.ipynb

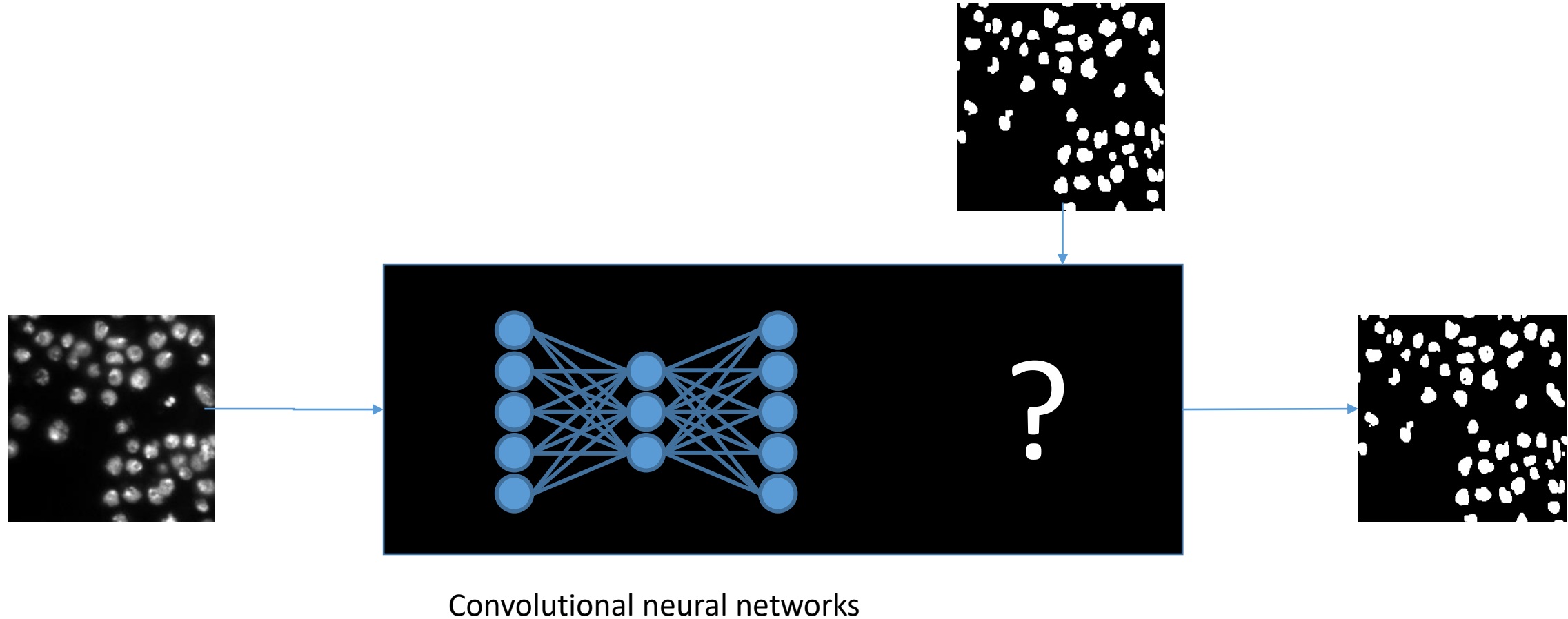
Summary & outlook

- In classical machine learning, we typically select features for training our classifier



Outlook: Deep learning for image analysis

- In deep learning, this selection becomes part of the black box



Today, you learned

- Machine learning for Pixel and Object segmentation
- Python
 - Scikit-learn / napari
 - Accelerated pixel and object classifiers (APOC)

Coming up next:

- Feature extraction

