

Machine Learning for Pixel and Object Classification

Robert Haase

With Material from

Deborah Schmidt, Jug Lab, MPI CBG

Uwe Schmidt, Myers Lab, MPI CBG

Martin Weigert, EPFL

Ignacio Arganda-Carreras, Universidad del País Vasco

Lecture overview

Today

- Machine learning for Pixel and Object Classification
 - Random Forest Classifiers
 - Algorithm evaluation
 - Outlook: Deep learning
- Fiji
 - Trainable Weka Segmentation
 - Labkit
 - CLIJ object classifier
- Python
 - scikit-learn / napari

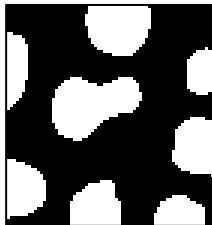
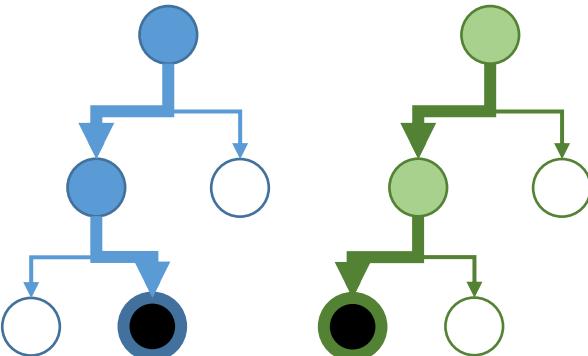
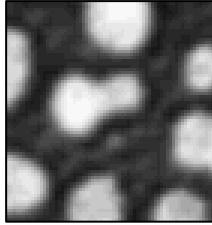


Image segmentation using thresholding

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

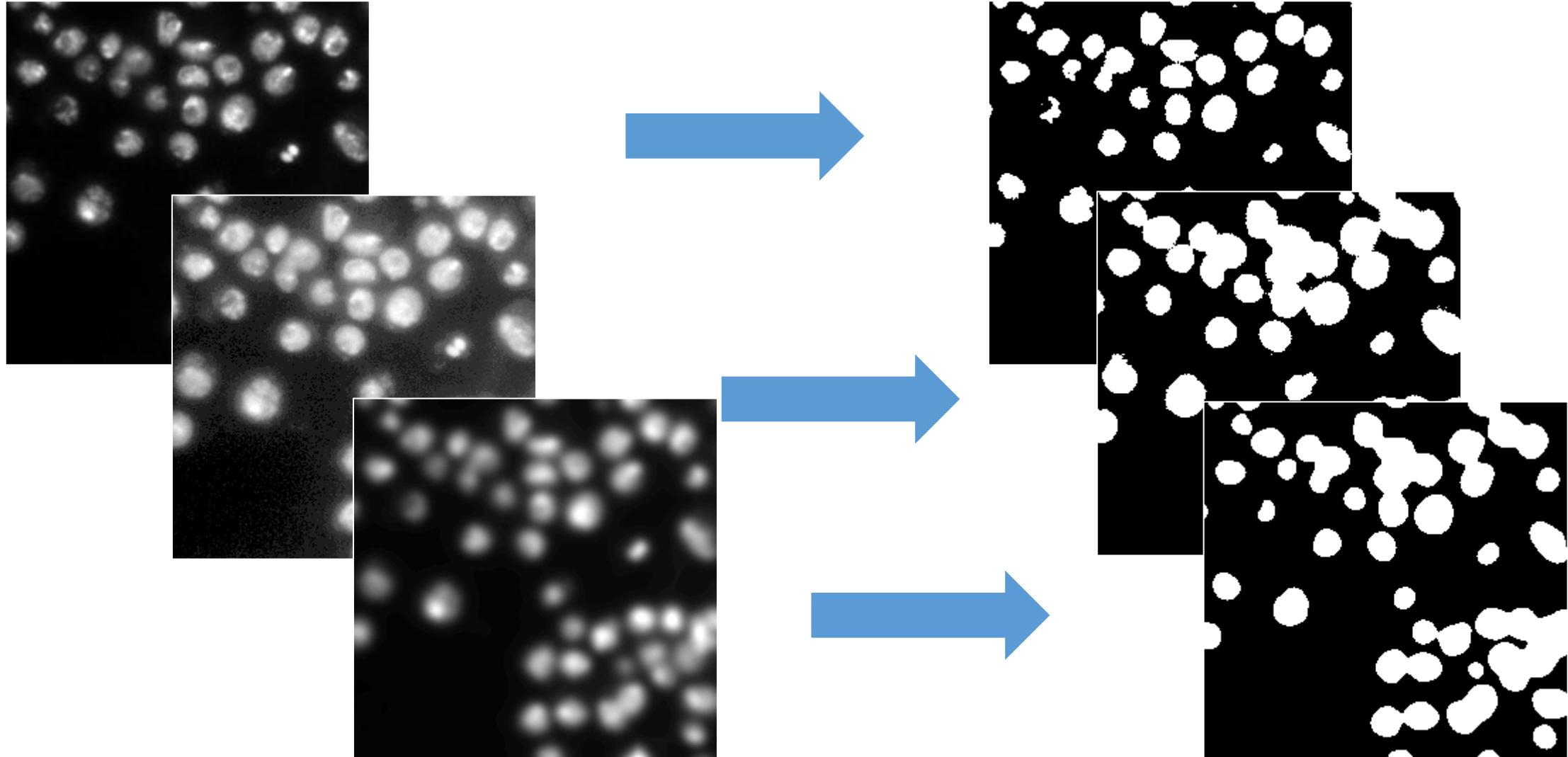


Image segmentation using thresholding

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

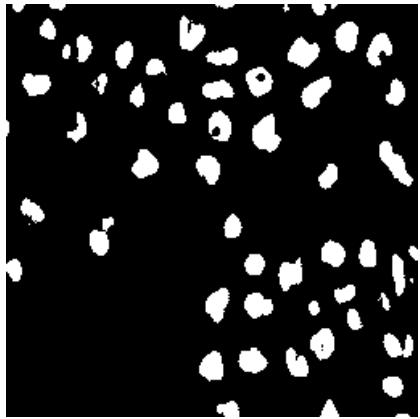
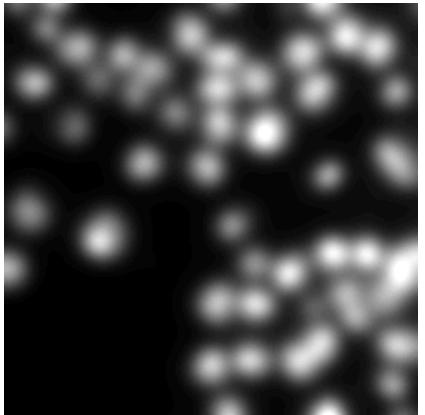
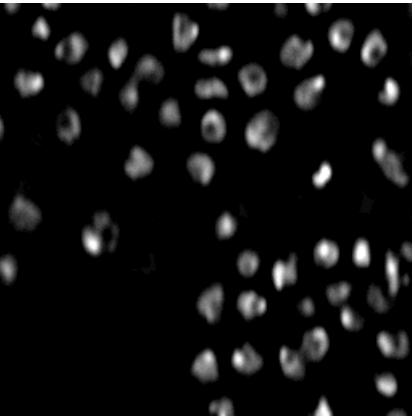
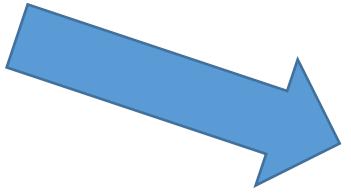
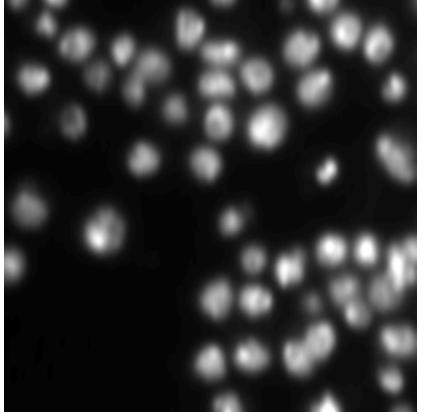
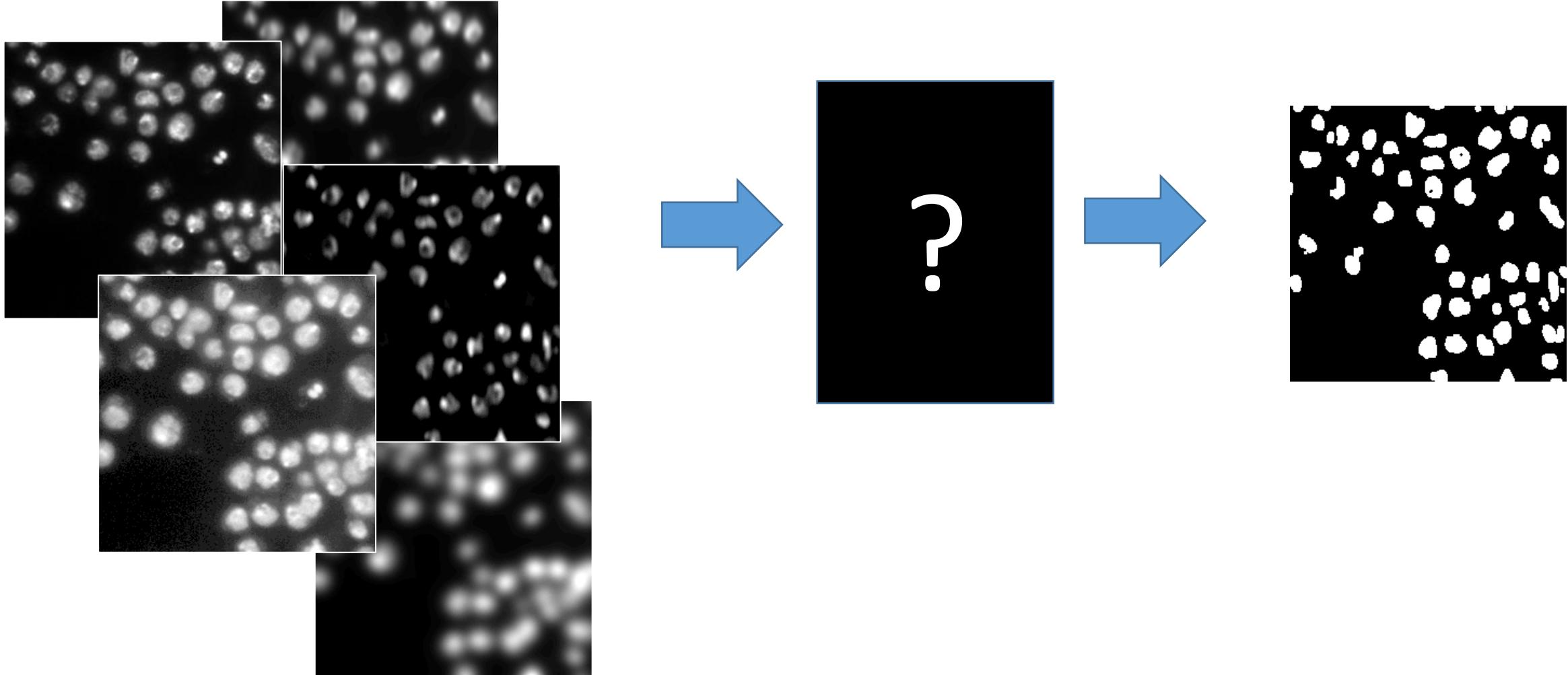


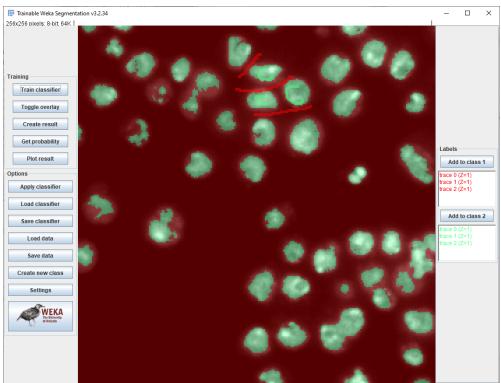
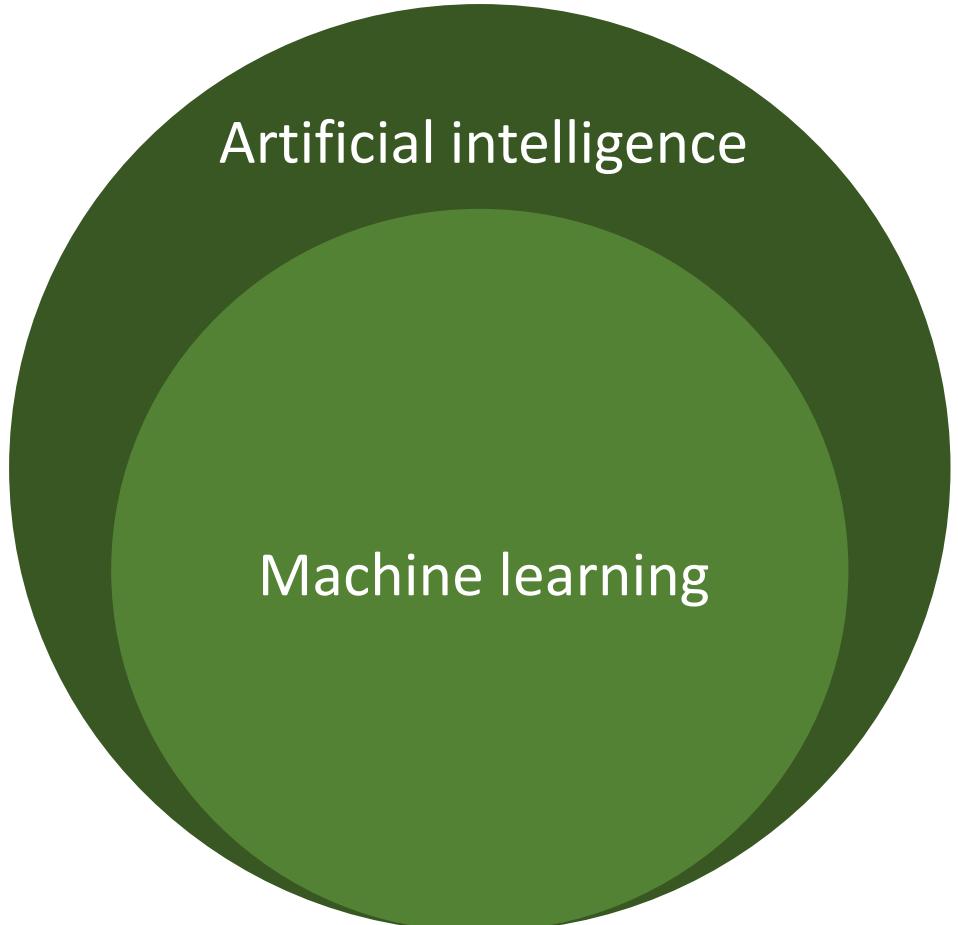
Image segmentation using thresholding

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

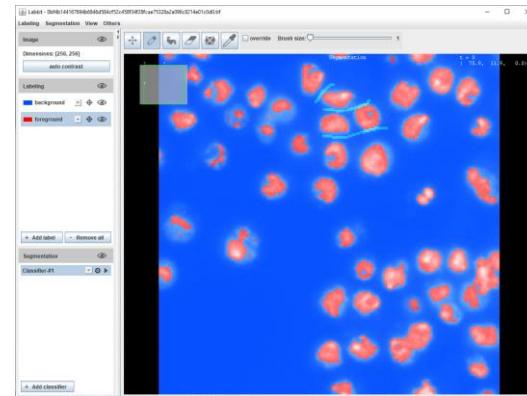


Machine learning

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

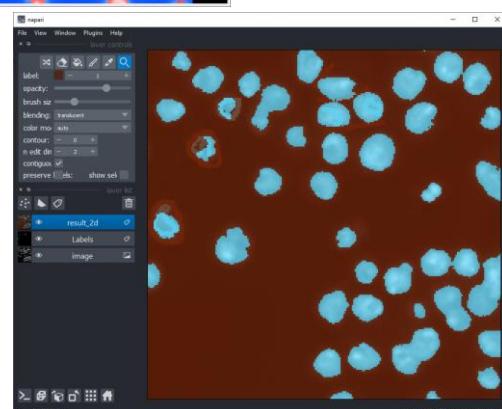


Trainable Weka Segmentation



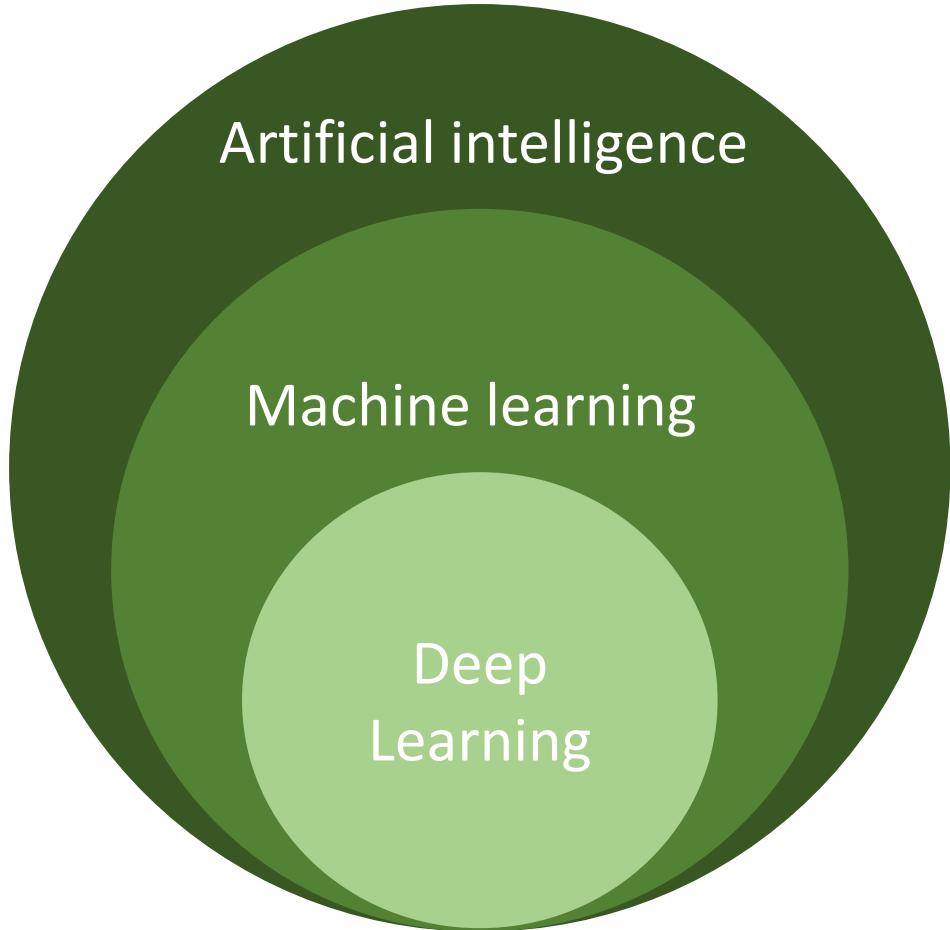
LabKit

Python /
scikit-learn /
napari

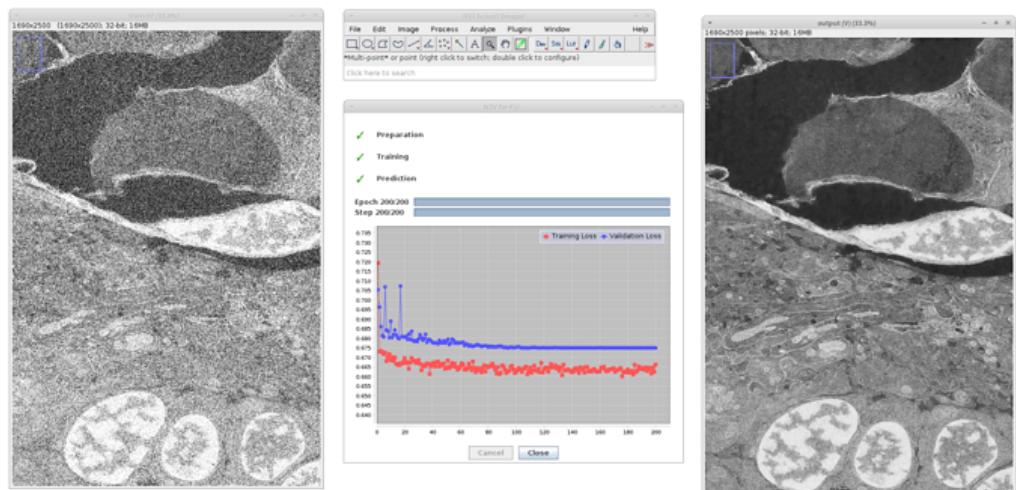


Machine learning

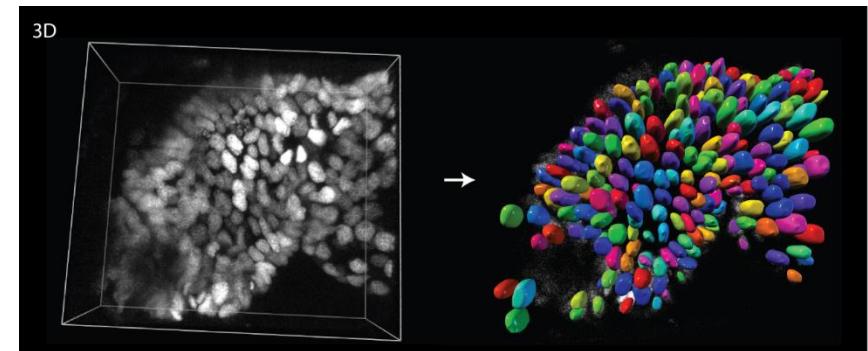
- A research field in computer science
- Finds more and more applications, also in life sciences.



Noise2Void



<https://forum.image.sc/t/noise2void-for-fiji/34552>

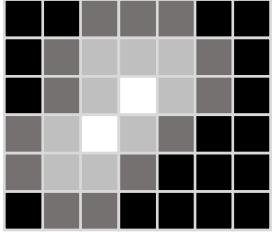


<https://imagej.net/StarDist>

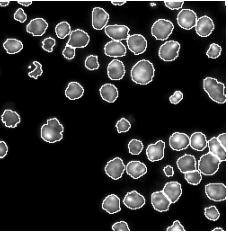
Machine learning

- Automatic construction of predictive models from given data

Pixels,



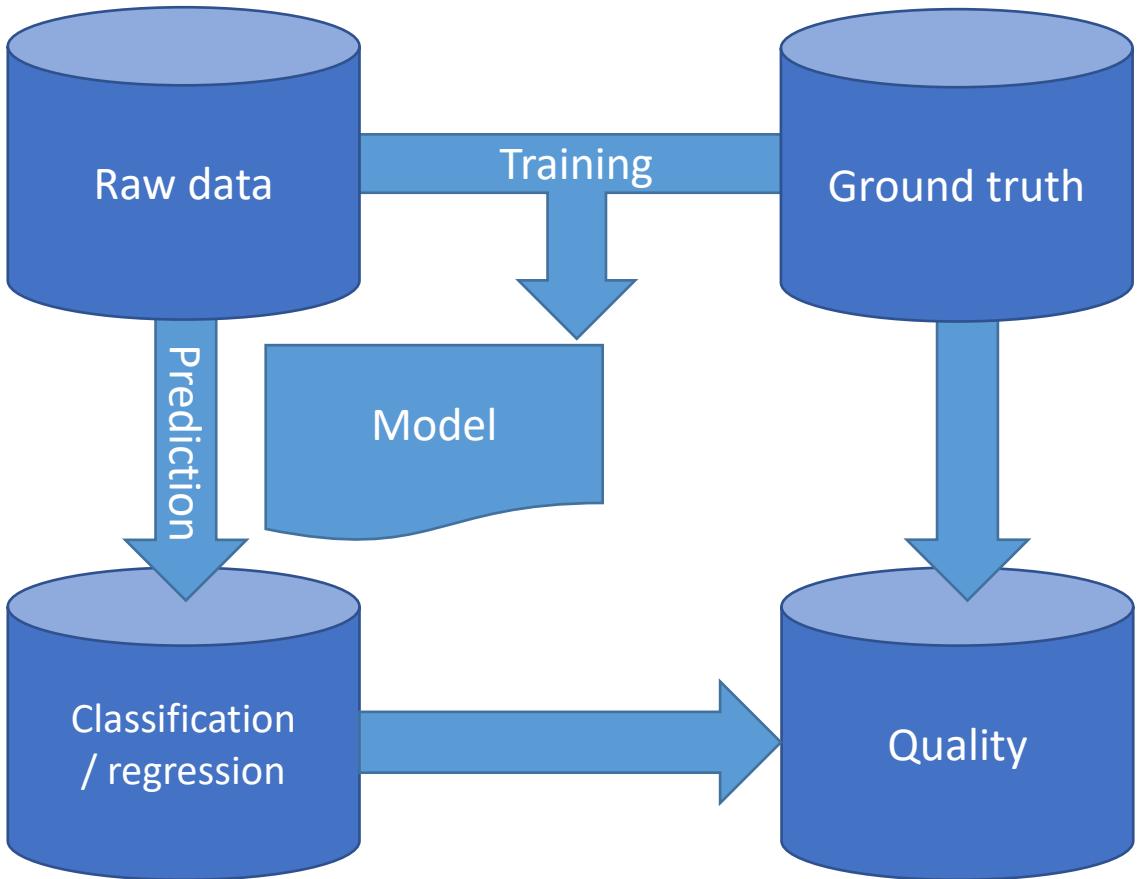
Objects,



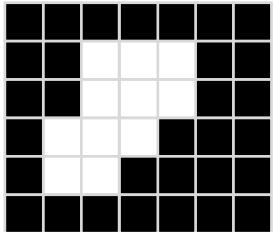
Images, Audio, Text, Measurements, ...



Annotated raw data, usually generated by humans



Dense Segmentation / Binarization



Object classification

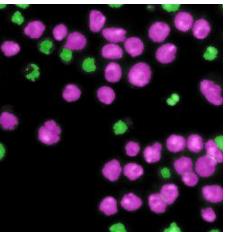
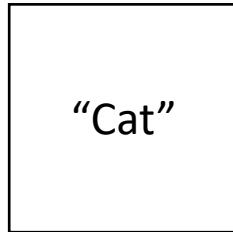
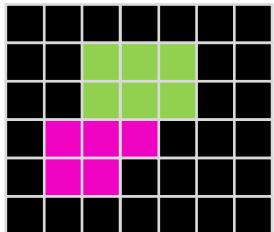


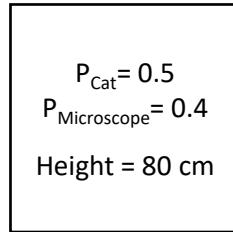
Image classification
“Cat”



Instance segmentation



Cont. quantity



Precision,
Recall

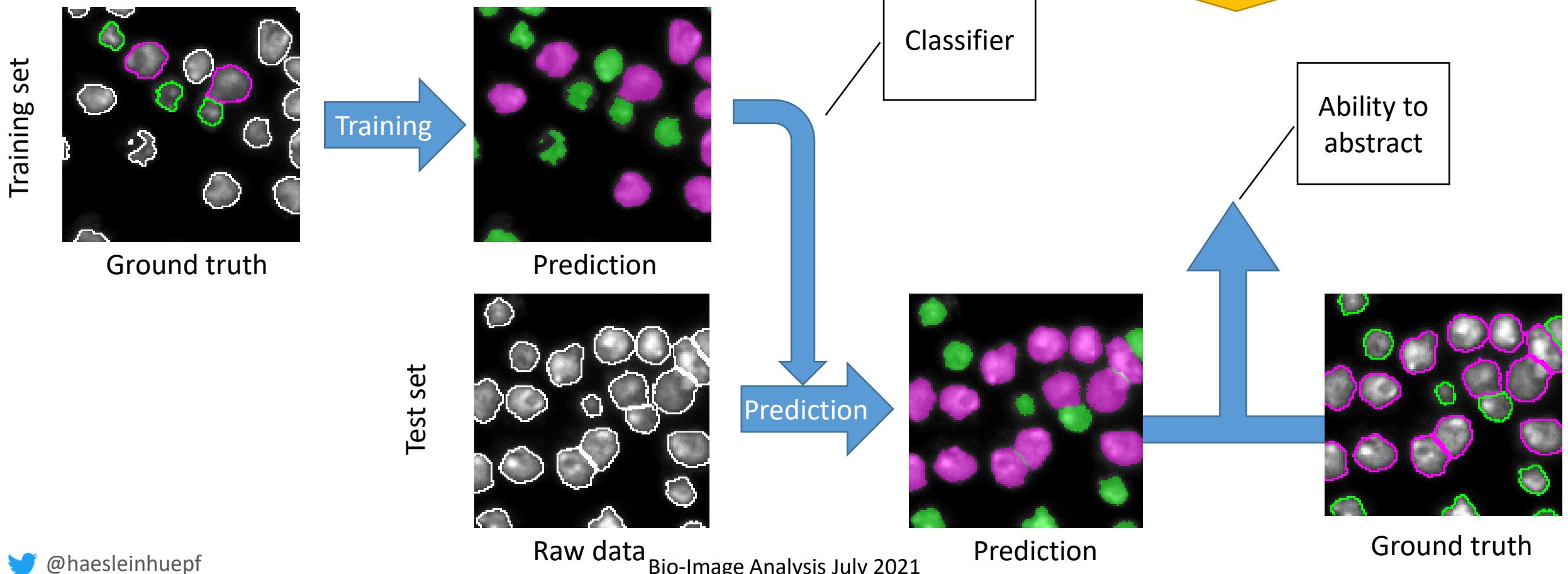
Overfitting / abstraction – selection of trees for the forest



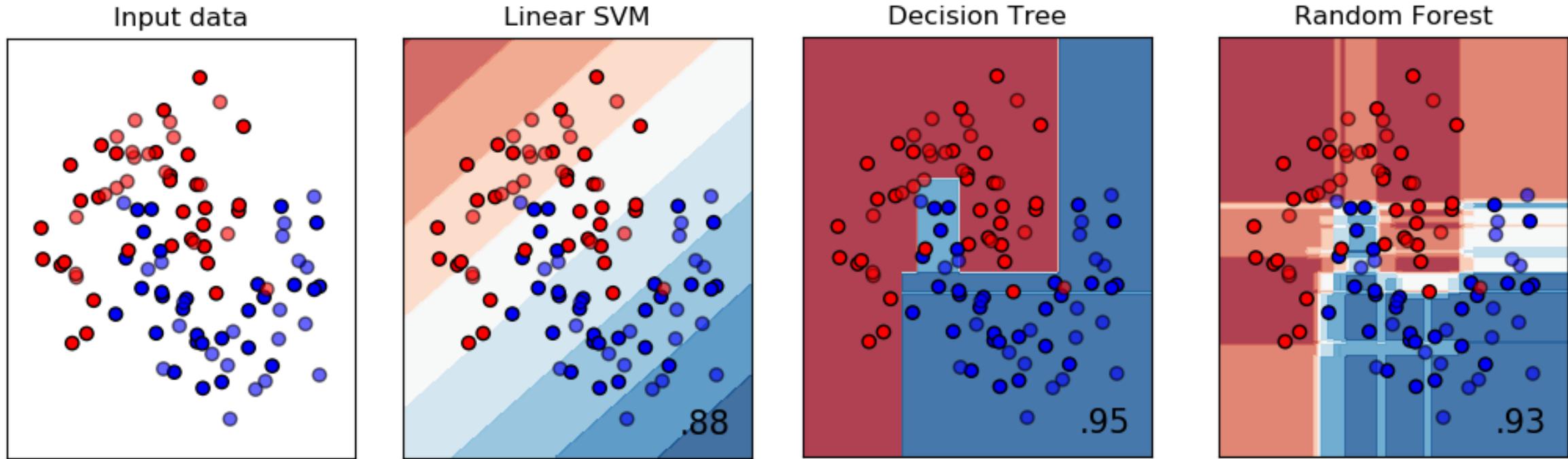
PoL
Physics of Life
TU Dresden



- 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)



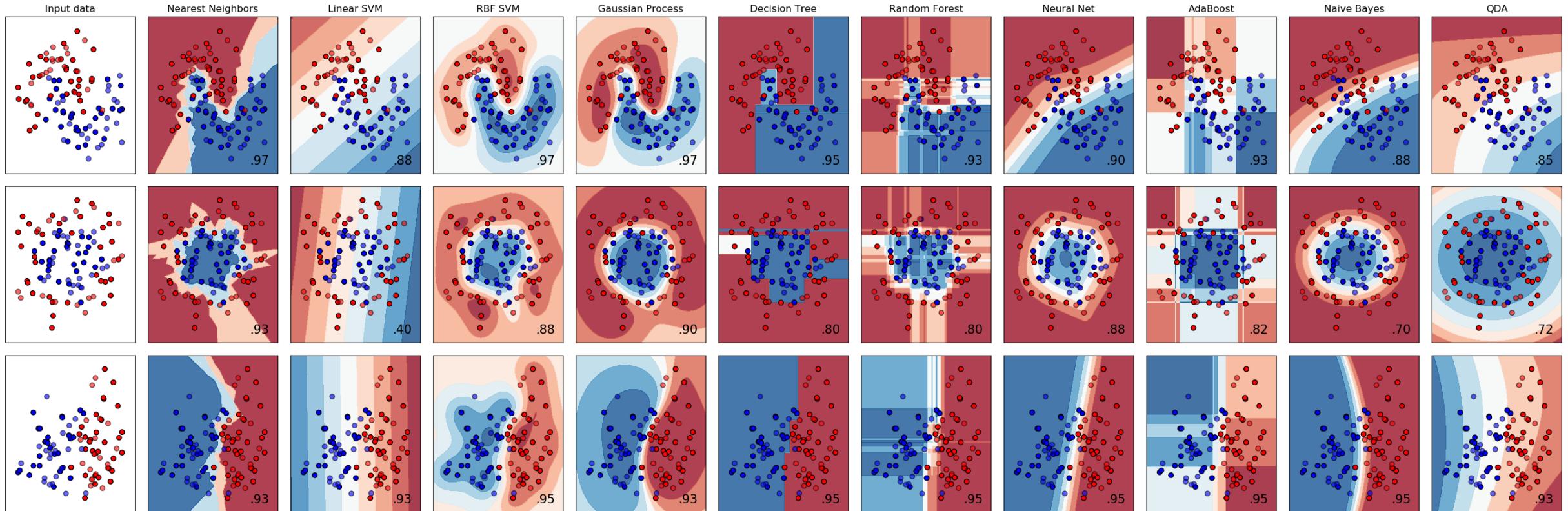
- Guess classification (**color**) from position of a sample in parameter space.



Adapted from https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Approaches

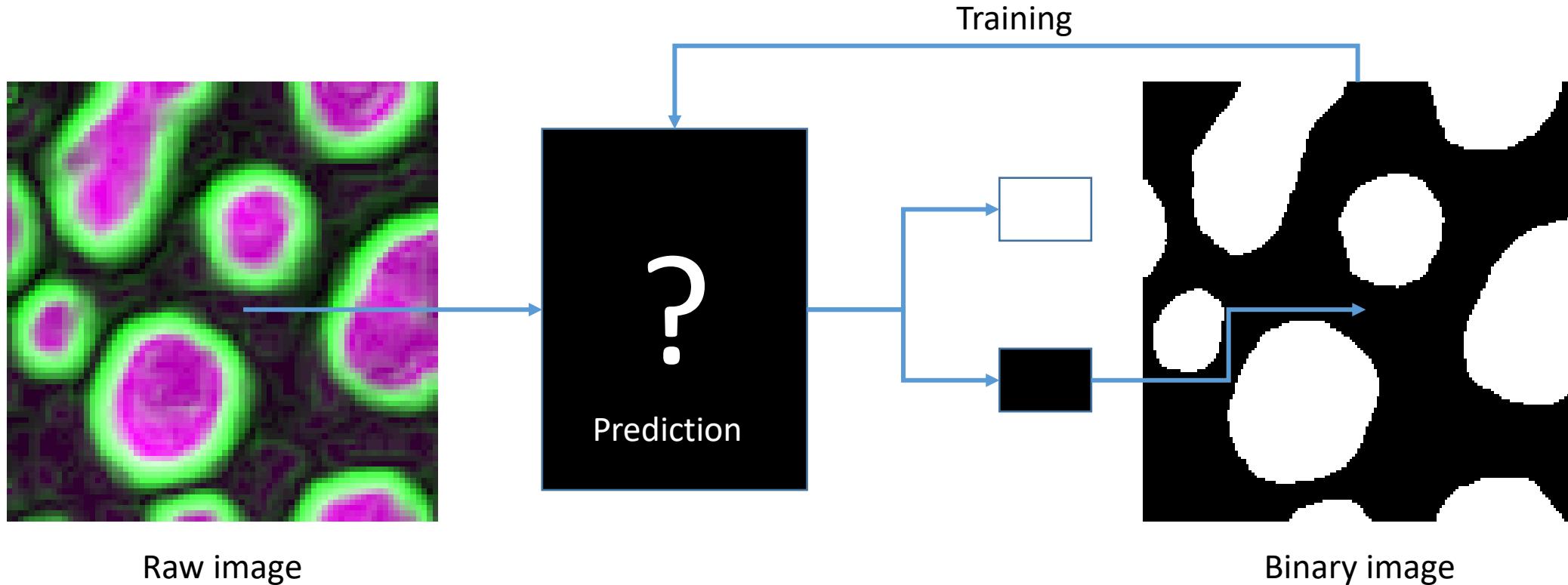
- The right approach depends on data, computational resources and desired quality



Adapted from https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

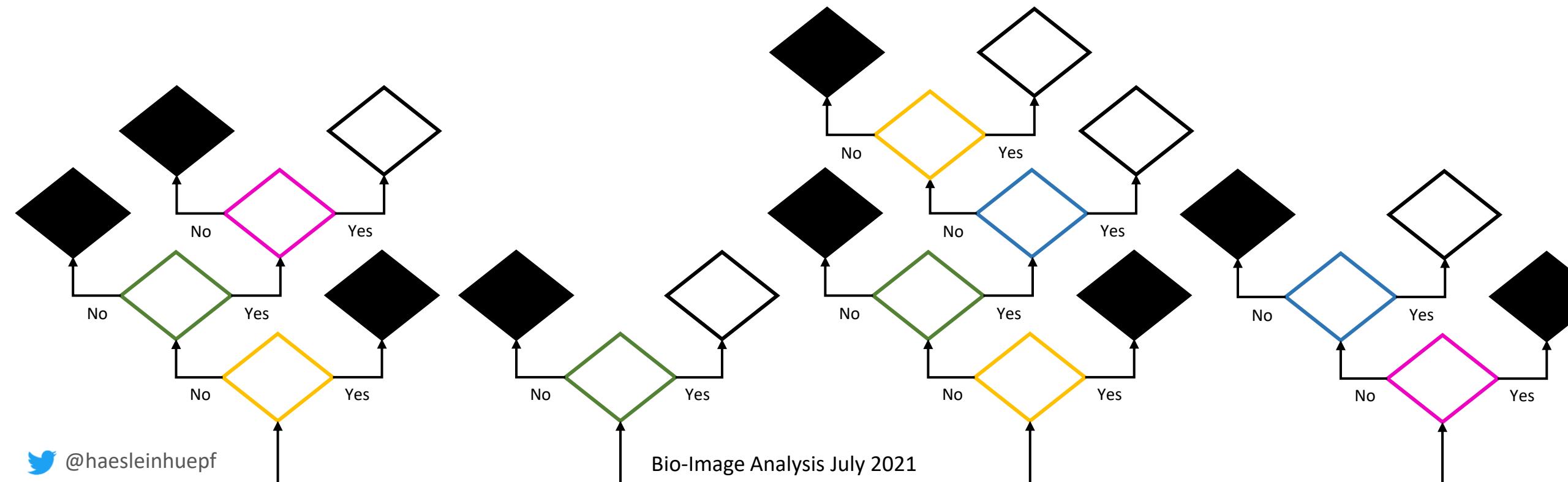
Machine learning for image segmentation

- *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



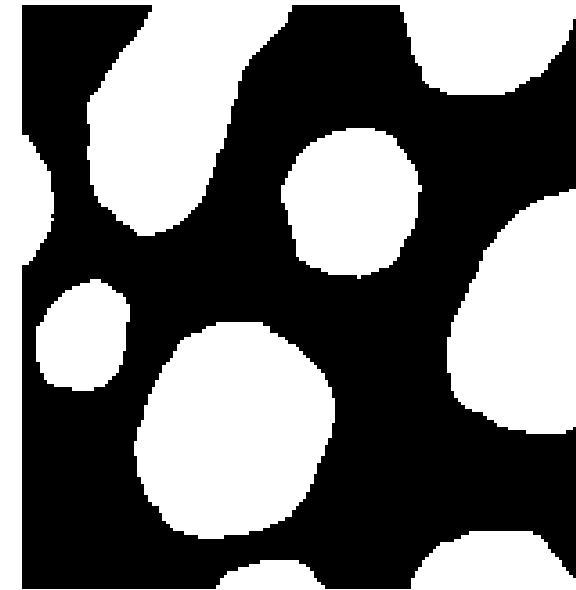
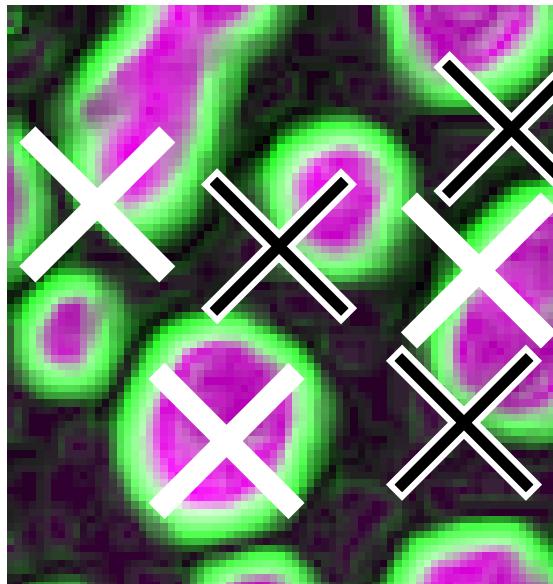
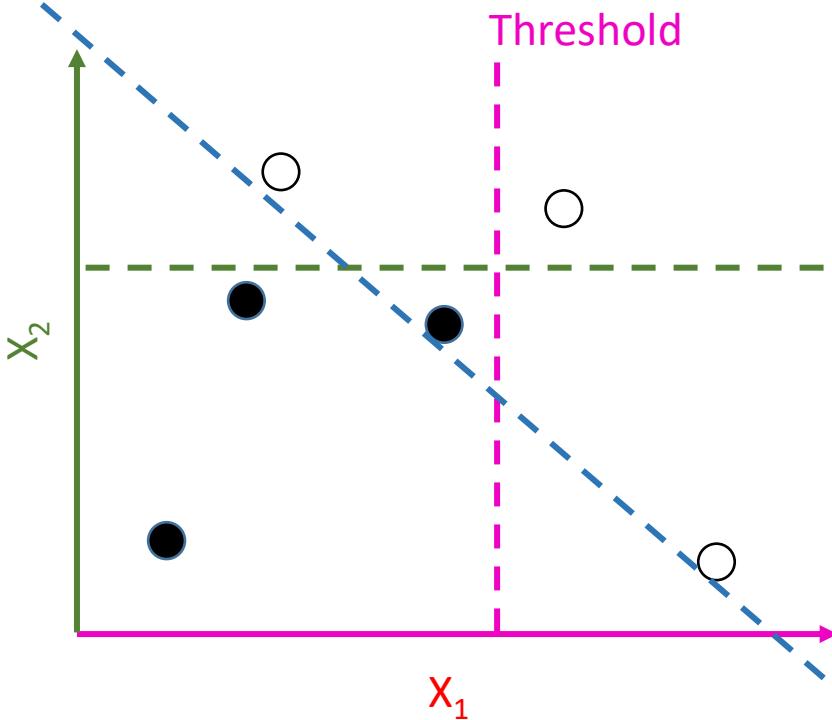
Random forest based image segmentation

- 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



Deriving random decision trees

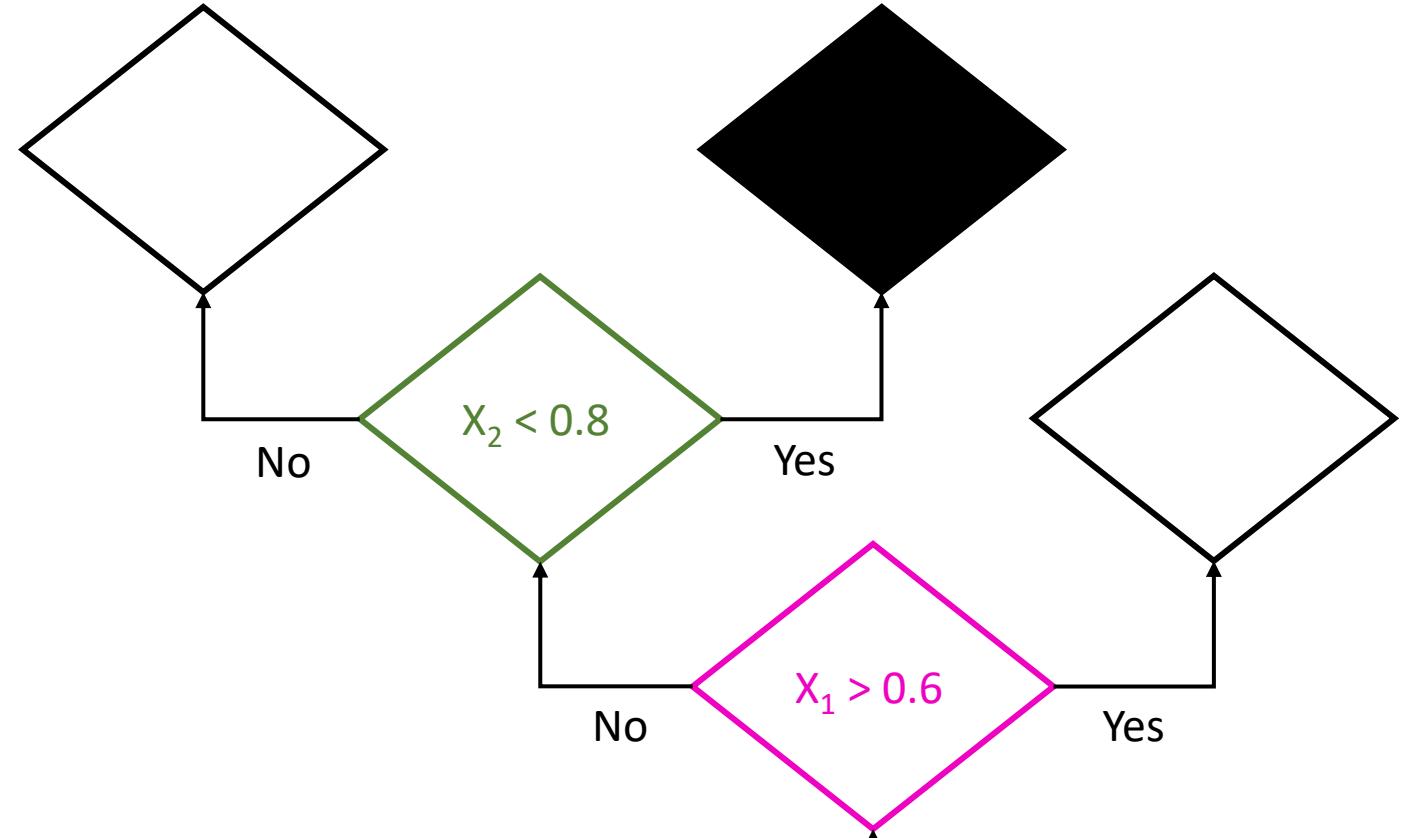
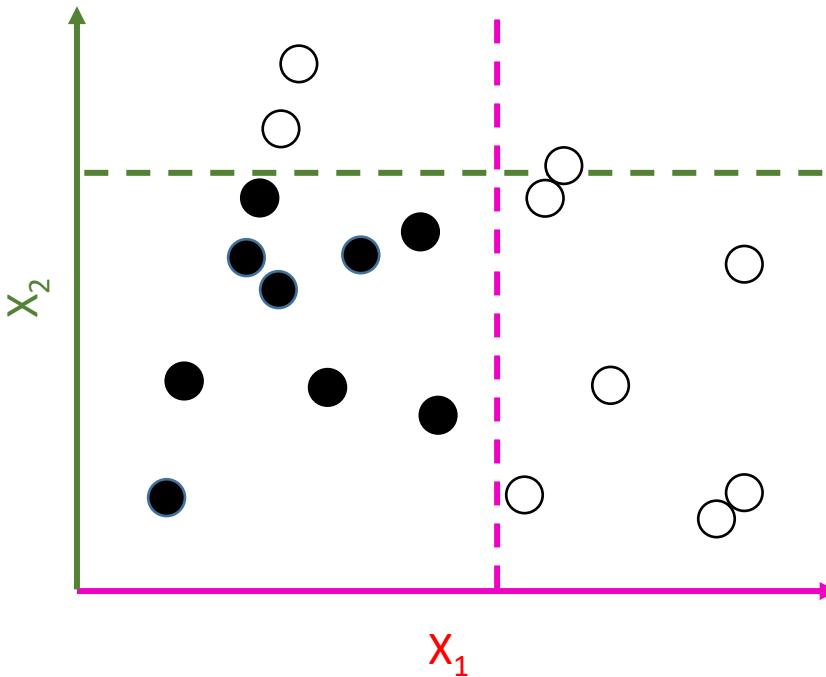
- For efficient processing, we randomly *sample* our data set
 - Individual pixels, their intensity and their classification



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

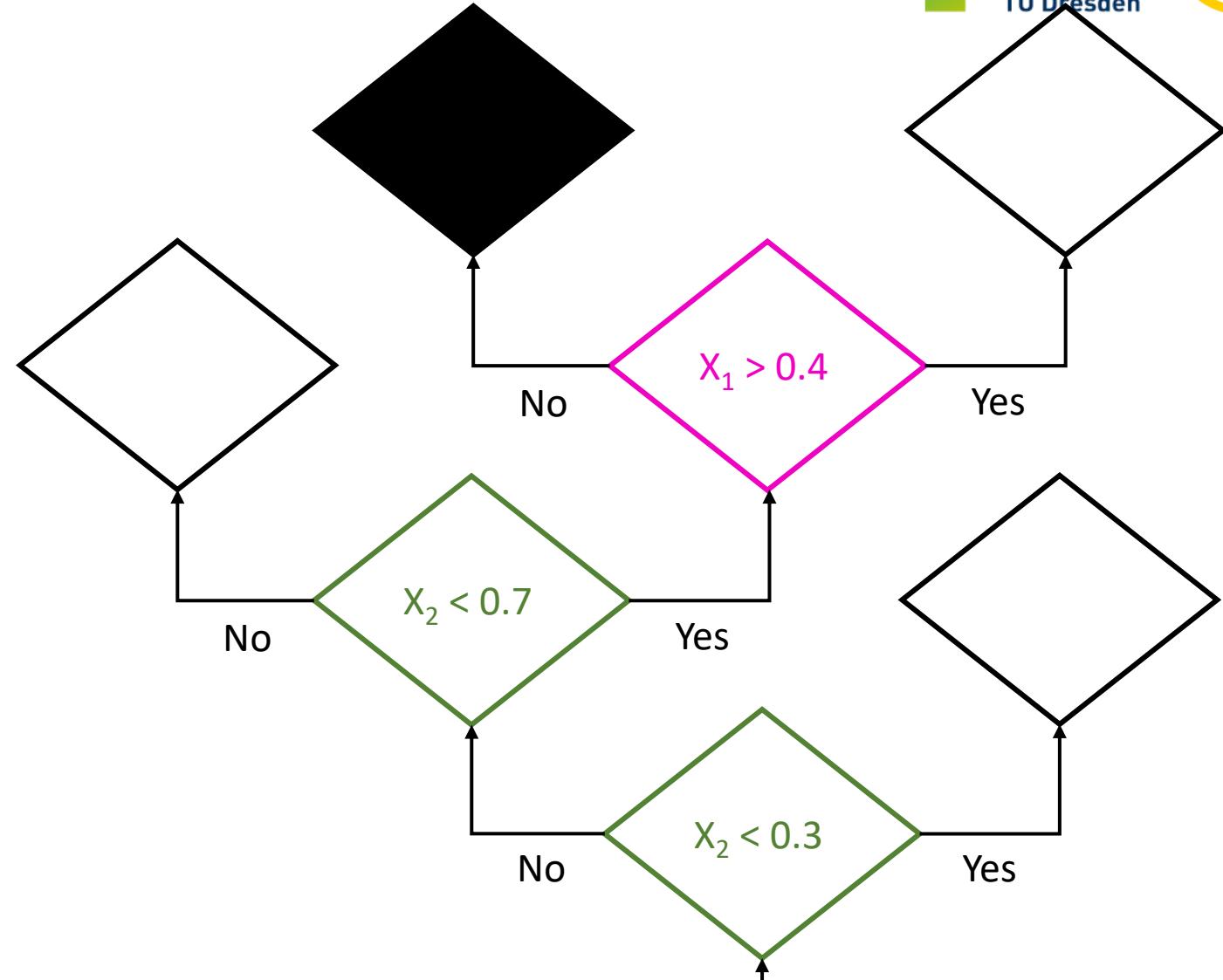
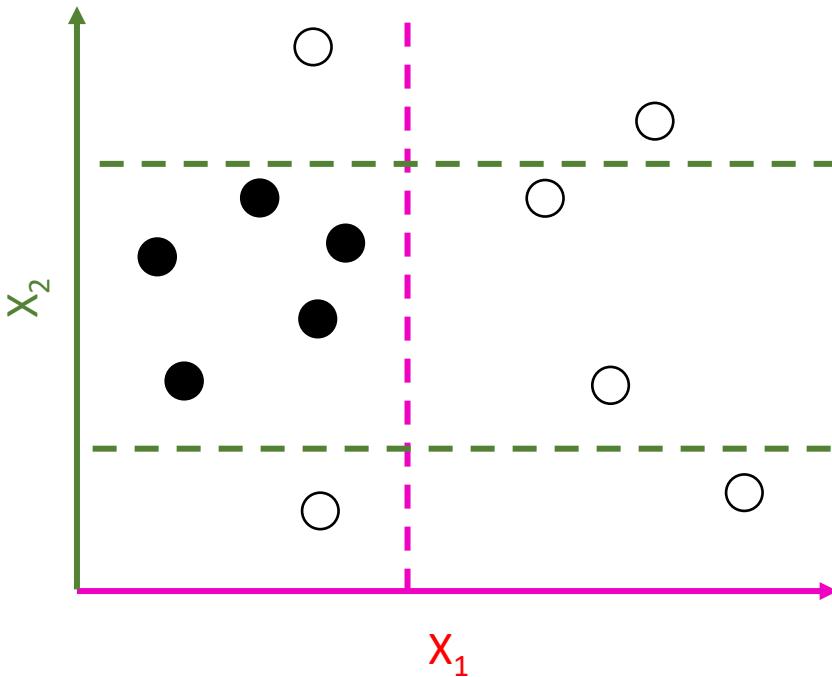
Deriving random decision trees

- Decision trees combine several thresholds on several parameters



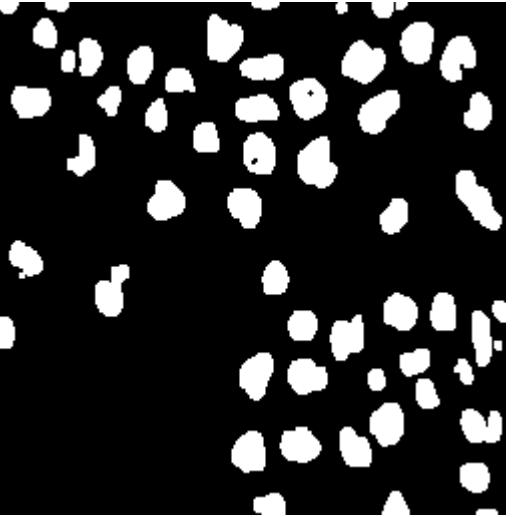
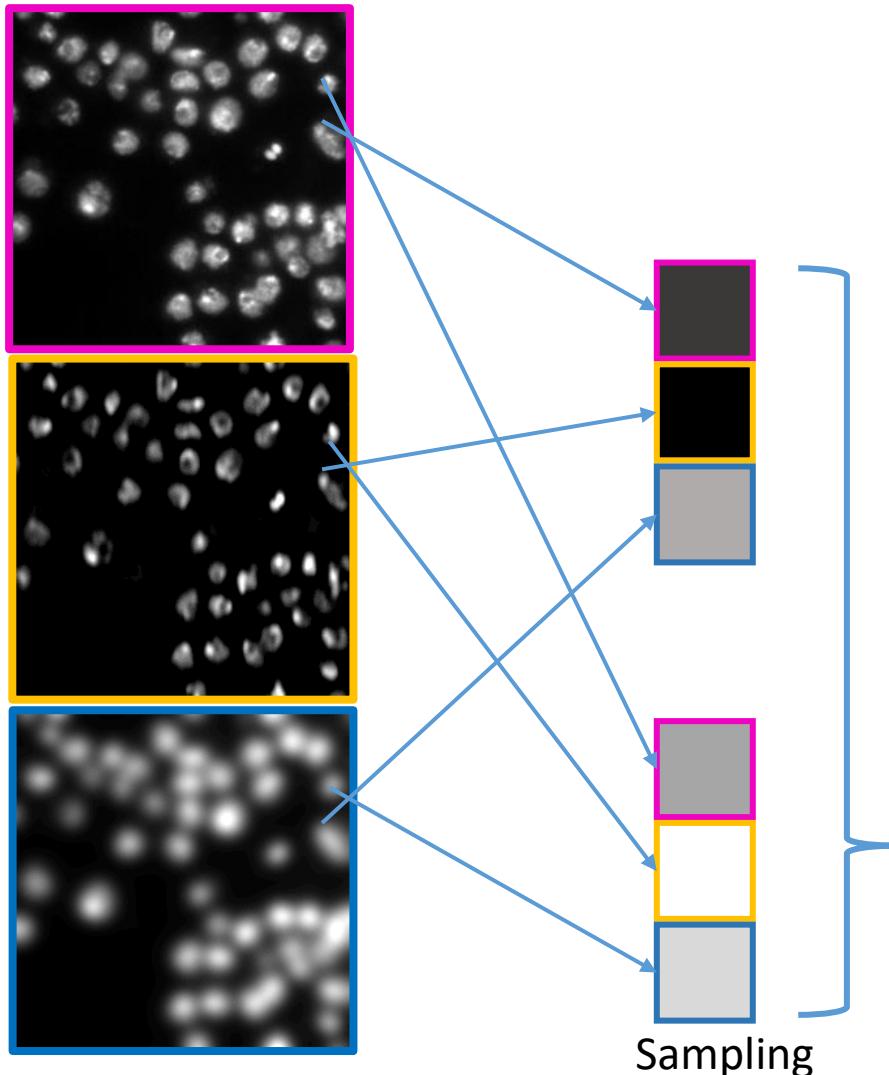
Deriving random decision trees

- Depending on sampling, the decision trees are different

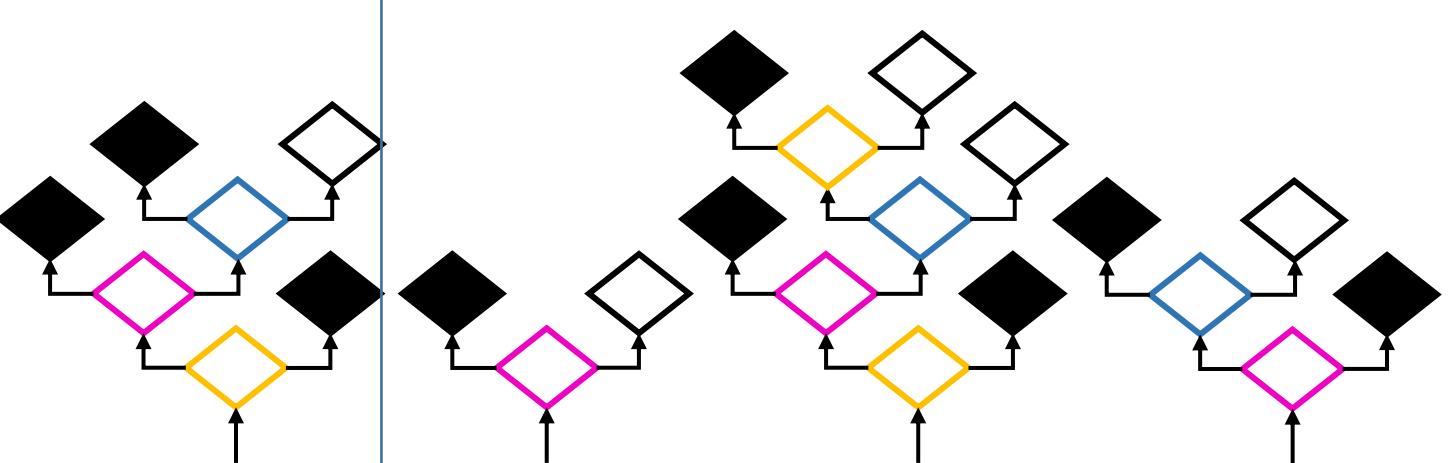


Random Forest Pixel Classifiers

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



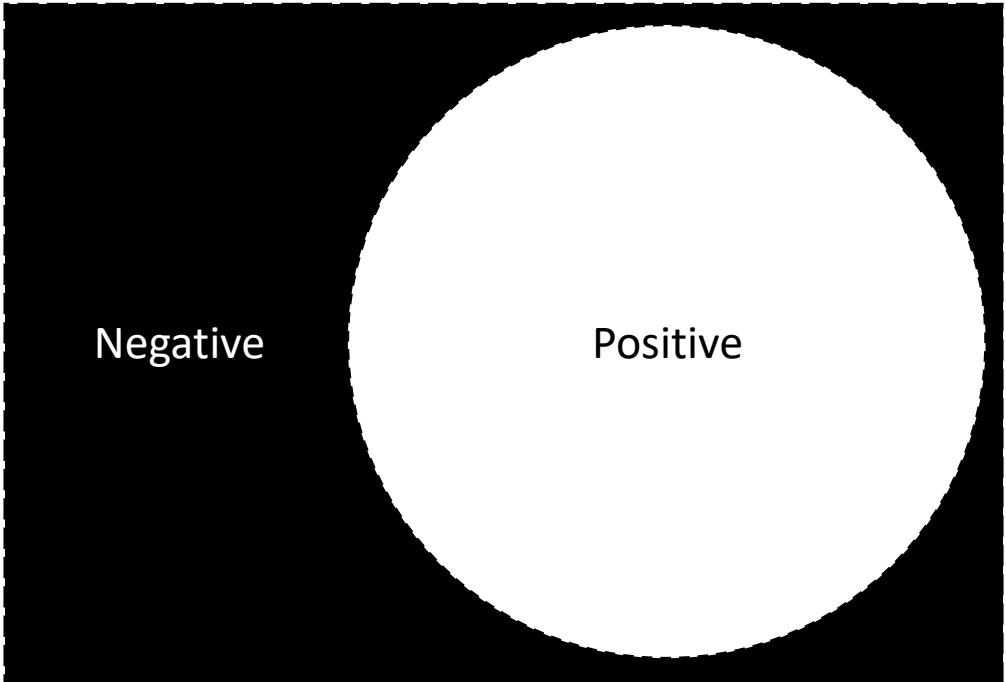
Selection



Bio-Image Analysis July 2021

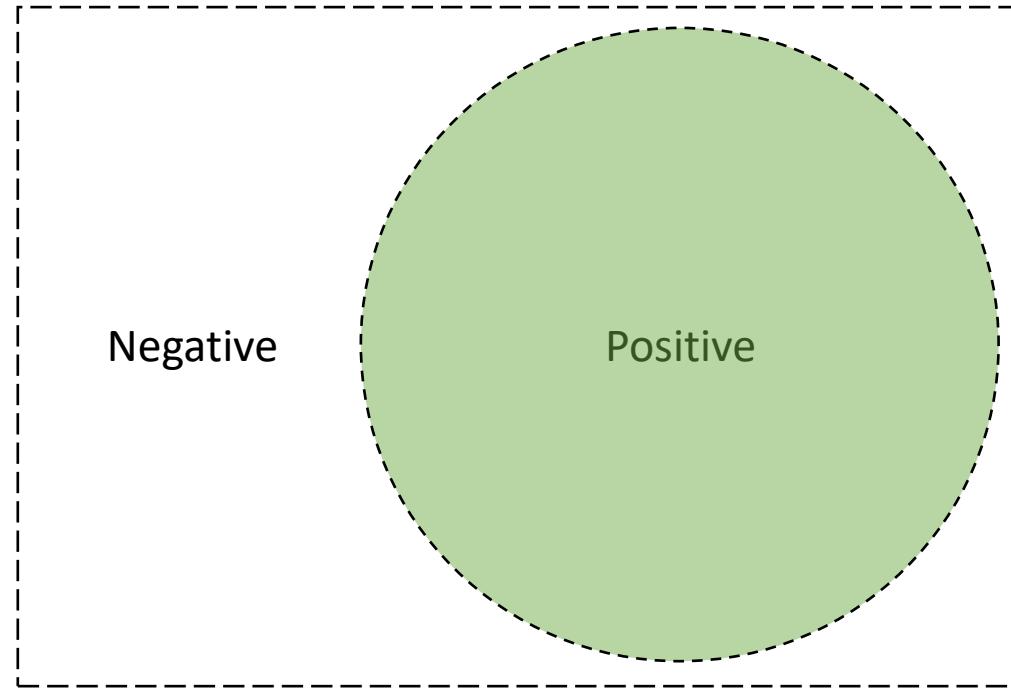
Algorithm evaluation

- In general
 - Define what's positive and what's negative.

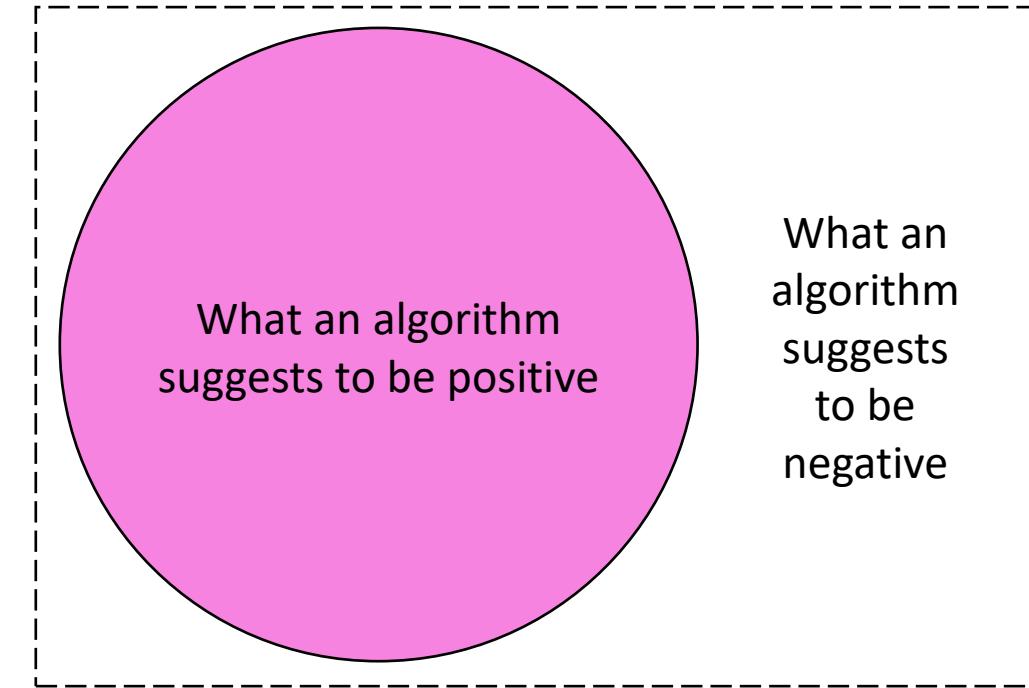


Algorithm evaluation

- In general
 - Define what's positive and what's negative.

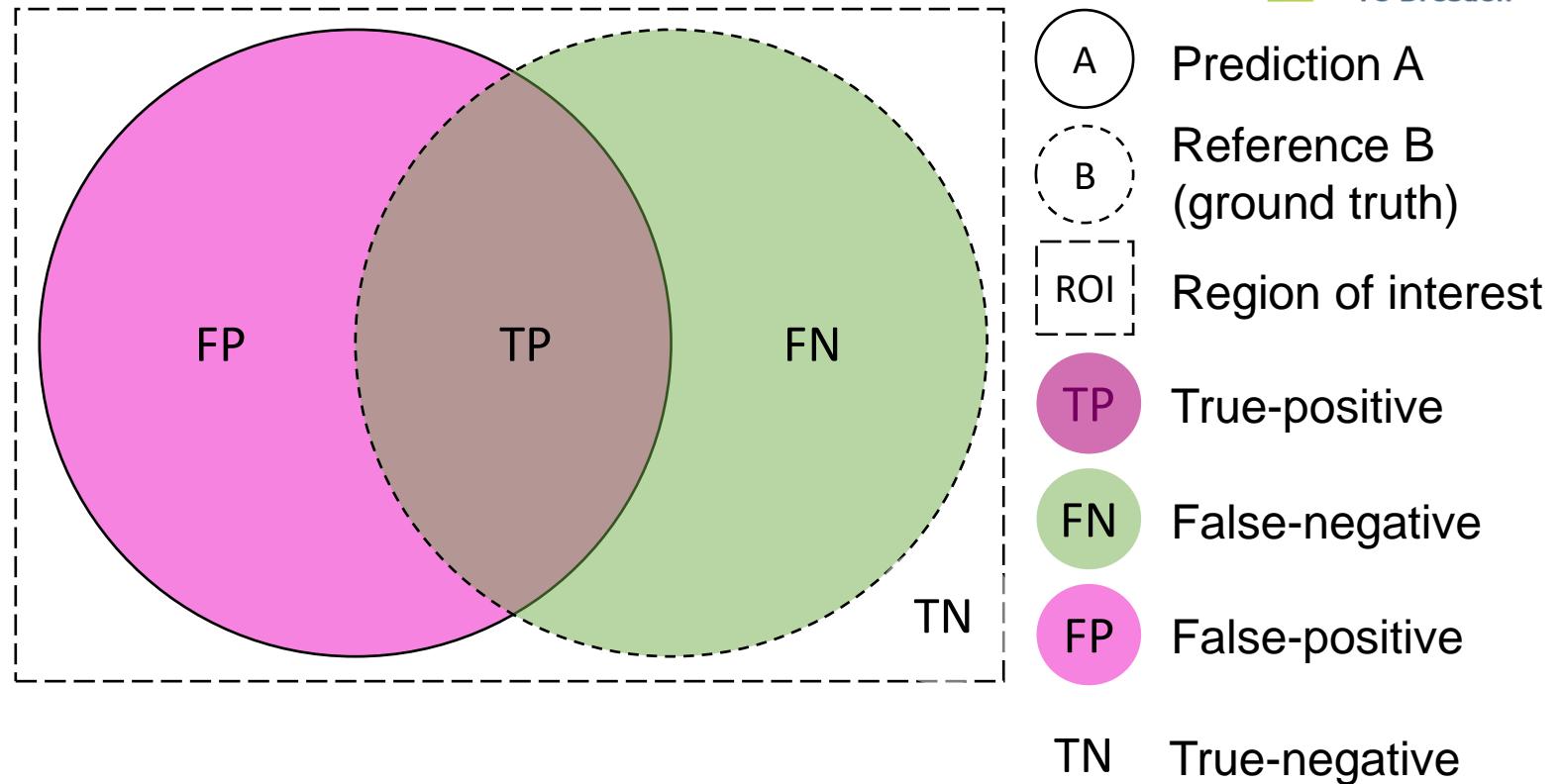


- In general
 - Define what's positive and what's negative.



Algorithm evaluation

- 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



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?

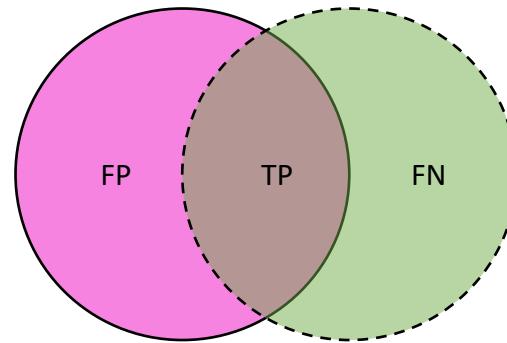
Pixel-wise versus Object-wise evaluation

- Pixel wise: Segmentation quality
- Object wise: Detection quality



Prediction

Ground truth

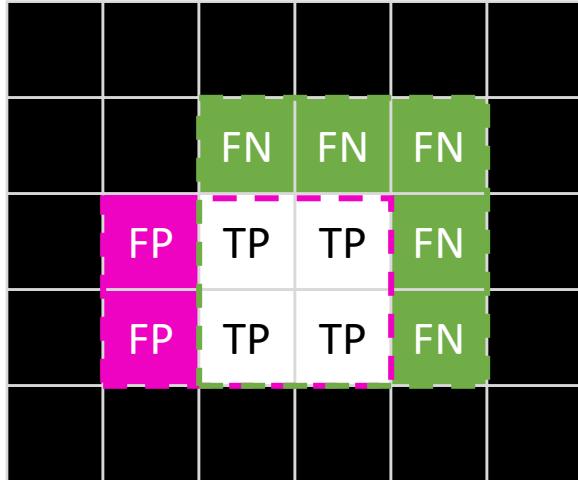


Precision

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

Recall
(a.k.a. sensitivity)

$$\frac{TP}{TP + FN}$$



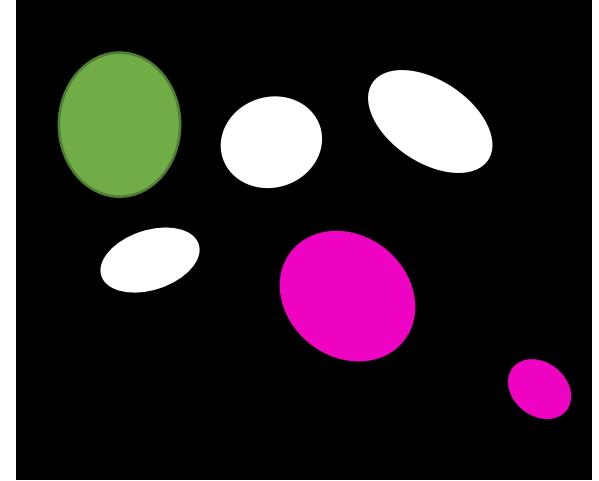
True-positive: 4

False-negative: 5

False-positive: 2

Precision: 4/6 = 66%

Recall: 4/9 = 44%



True-positive: 3

False-negative: 1

False-positive: 2

Precision: 3/4 = 75%

Recall: 4/9 = 44%

Pixel-wise versus Object-wise evaluation

- In practice: Mixed



Overlap
(a.k.a. Jaccard index)

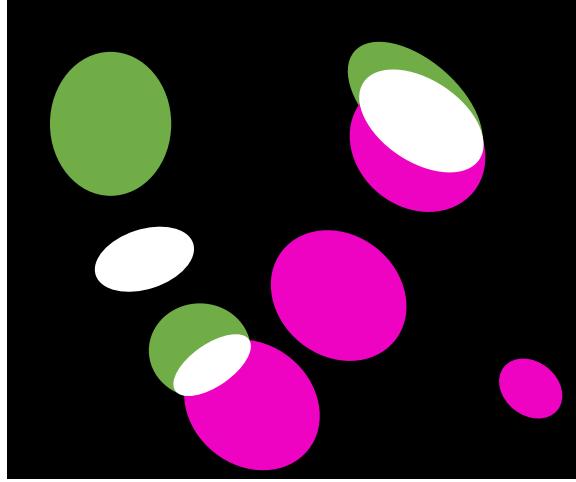
$$\frac{TP}{TP + FN + FP}$$

Precision

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

Recall
(a.k.a. sensitivity)

$$\frac{TP}{TP + FN}$$



Objects with at least 50% *pixel-wise overlap* between P and GT
True positive: 2

False positives: 3

False negatives: 2

Precision: $2/5 = 40\%$

Recall: $2/4 = 50\%$

- Voxel-wise Youden-Index

$$YI = p_{TP} + p_{TN} - 1$$

- Volume error

$$\Delta_V = V_A - V_B$$

$$\delta_V = \frac{\Delta_V}{V_B}$$

- Dice Index

$$DI(A, B) = \frac{2|A \cap B|}{|A| + |B|}$$

- Jaccard Index

$$JI(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{DI}{2 - DI}$$

- Contour distance

$$d_{e,min}(a, B) = \min(d_e(a, b) | b \in B)$$

$$\bar{d}_c(A, B) = \frac{\sum_{\forall a \in C(A)} d_{e,min}(a, C(B))}{|C(A)|}$$

$$\bar{d}_{bil,c}(A, B) = \frac{\bar{d}_c(A, B) + \bar{d}_c(B, A)}{2}$$

- Hausdorff distance

$$d_H(A, B) = \max(d_{e,min}(a, B) | a \in A)$$

$$d_{bil,H}(A, B) = \max(d_H(A, B), d_H(B, A))$$

- Simplified Hausdorff distance

$$d_H(A, B) = \max(d_{e,min}(a, C(B)) | a \in C(A))$$

- Volume standard deviation

$$\delta_{\bar{V}} = 2 \frac{|V_A - V_B|}{|V_A + V_B|}$$

- Classification error

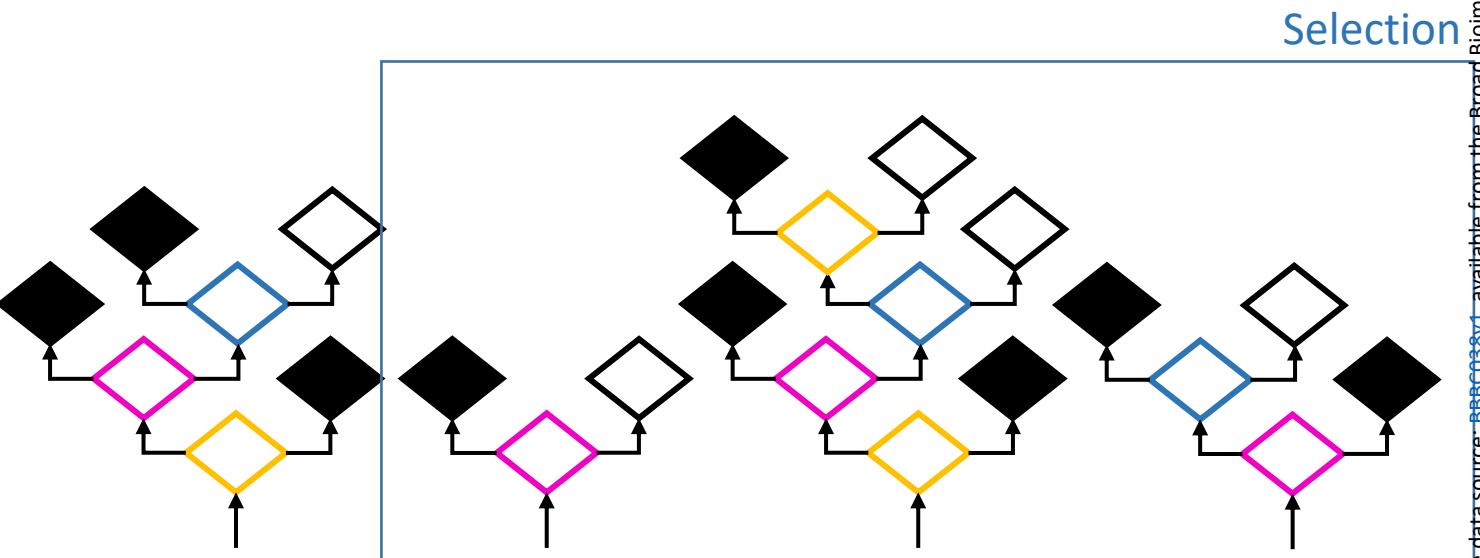
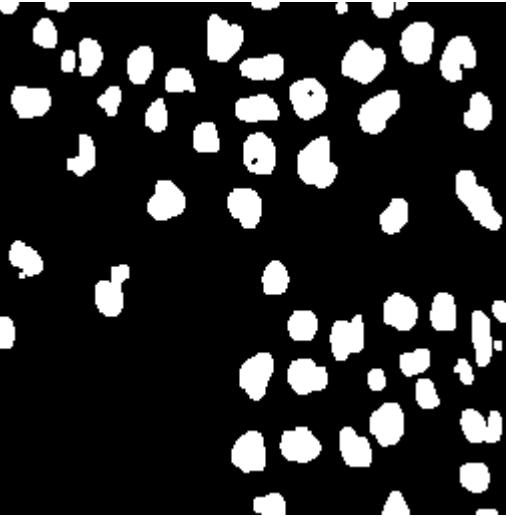
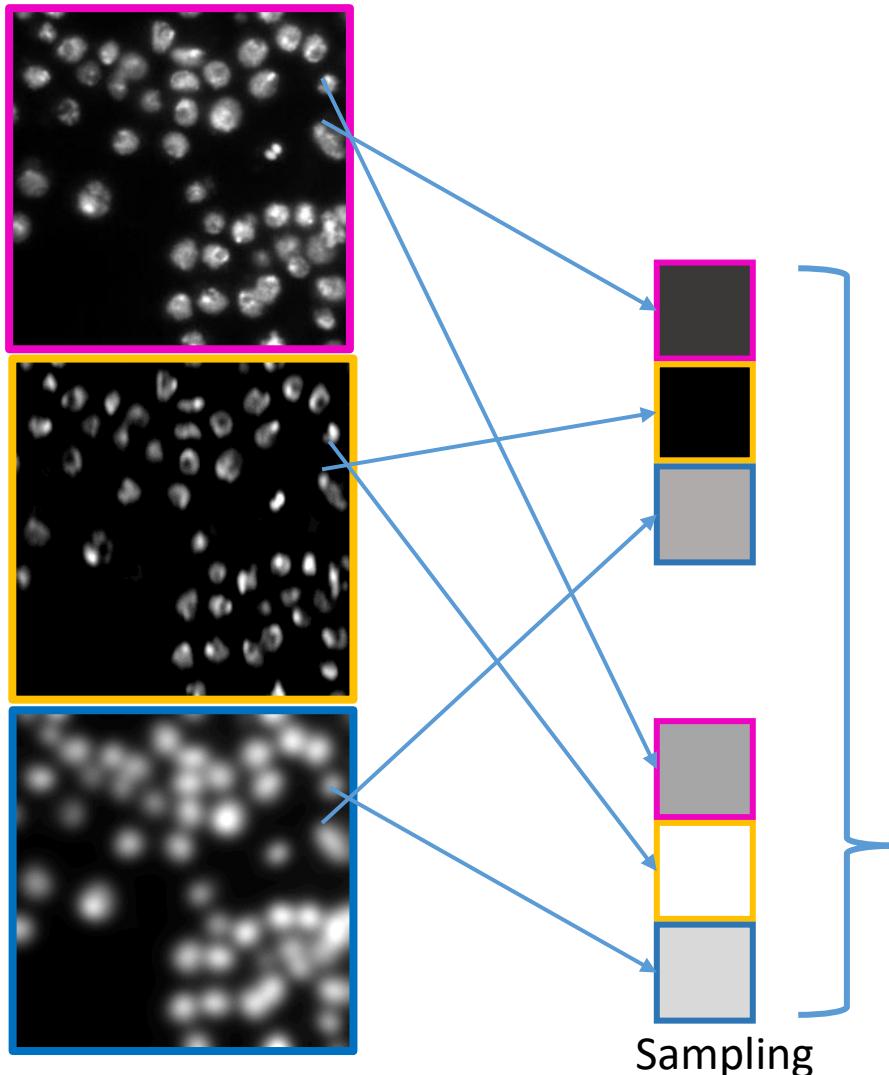
$$e_{Class} = \frac{H}{|TP| + |FN|}$$

- Hamming distance

$$d_h = |A \cup B| - |A \cap B| \\ = |FP| + |FN|$$

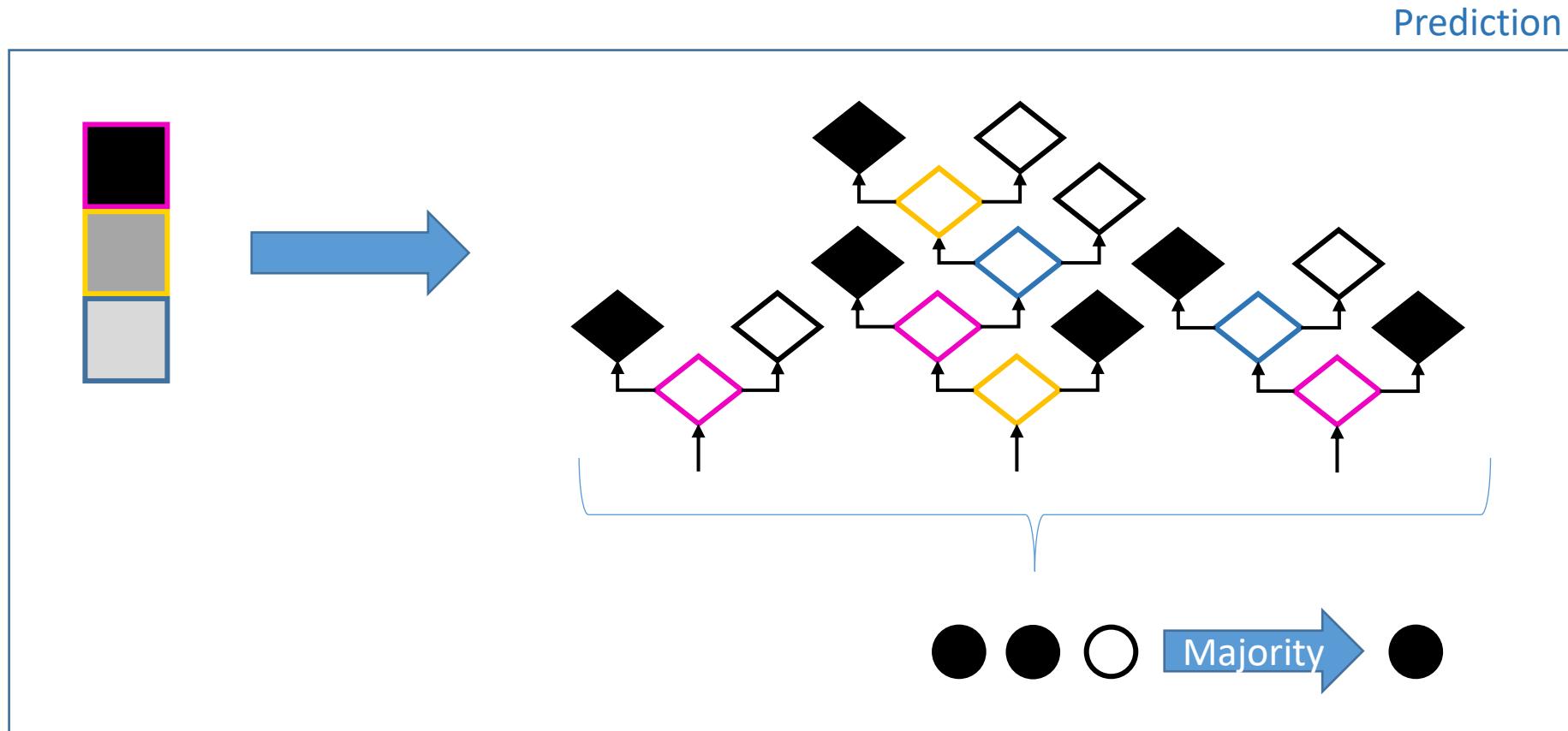
Random Forest Pixel Classifiers

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



Random Forest Pixel Classifiers

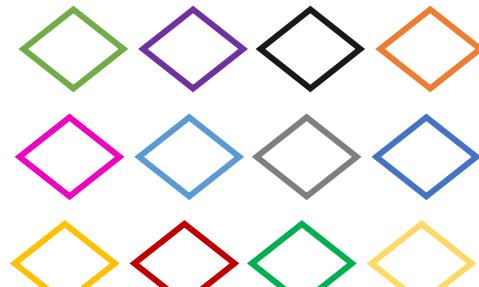
- Combination of individual tree decisions by voting or max / mean



Random Forest Pixel Classifiers

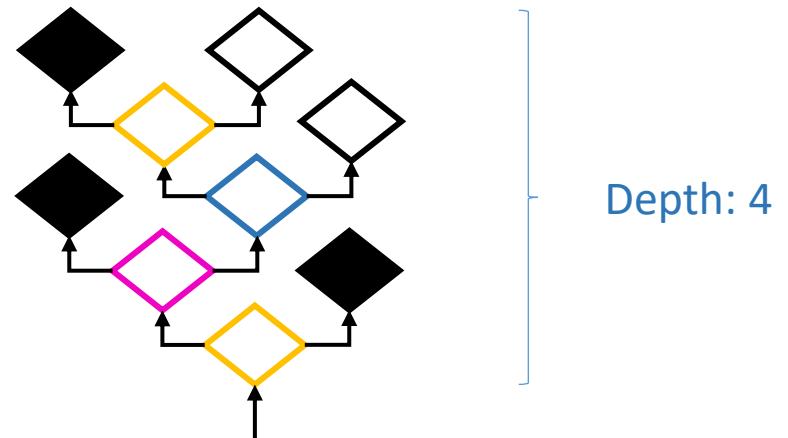
- Typical numbers for pixel classifiers in microscopy

Available features: > 20



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

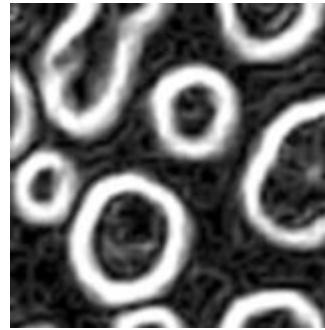
Selected features: <= depth



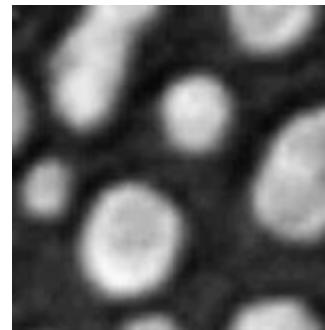
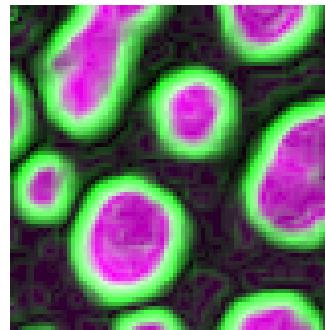
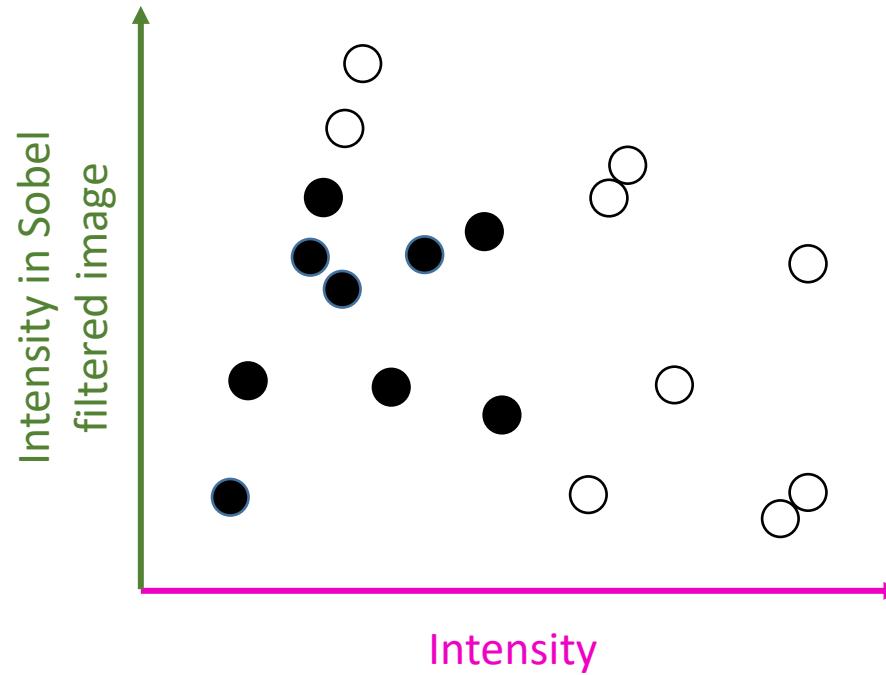
Number of trees: > 100



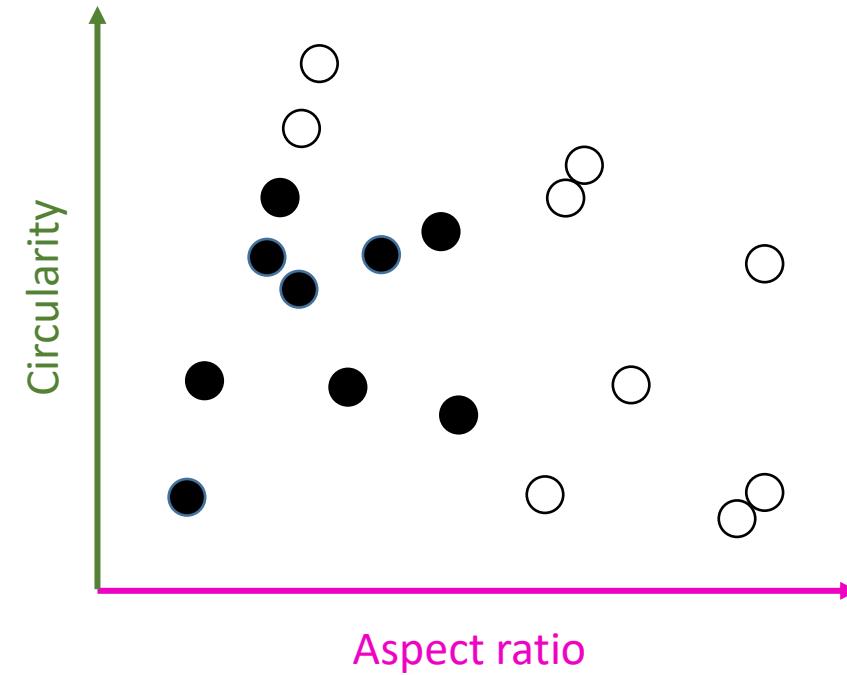
- What if we exchange pixel features with object features?



Pixel classification



Object classification

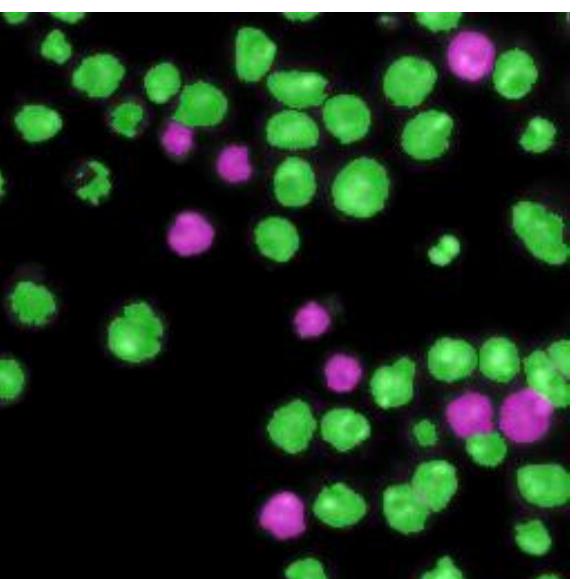
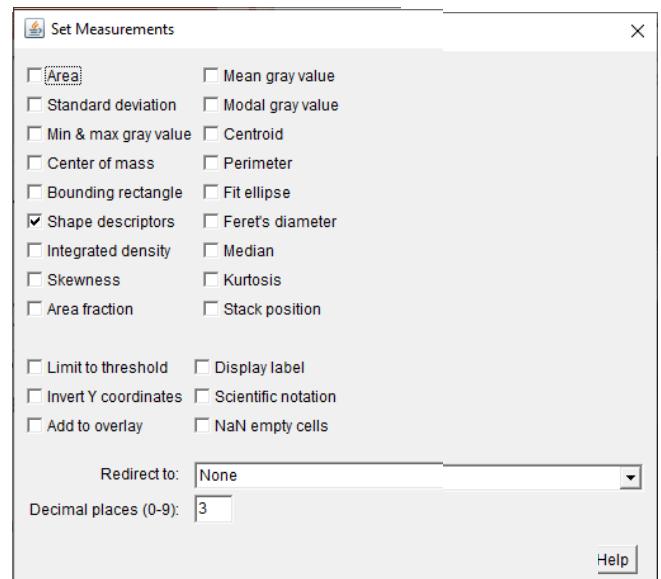
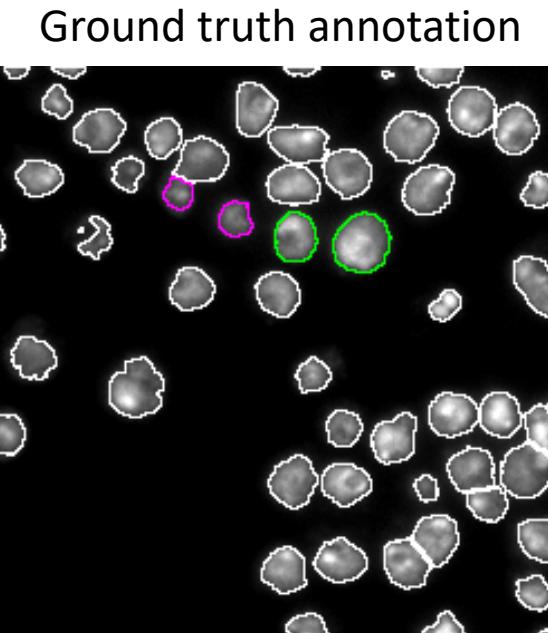
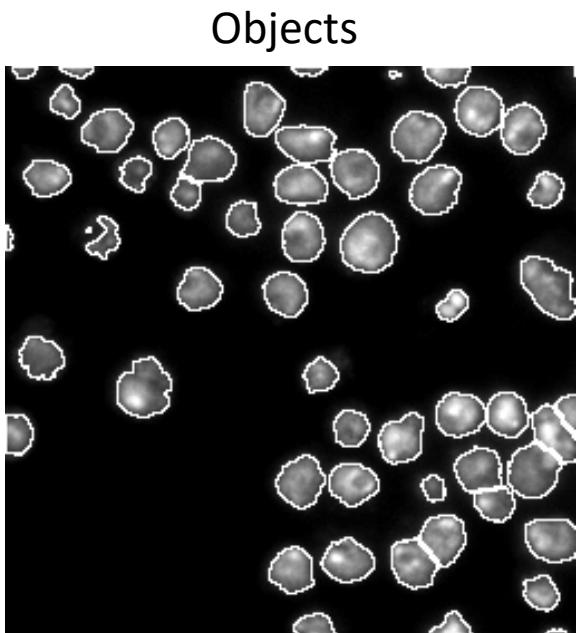


- The algorithms work the same but with different
 - Features
 - Number of features
 - Tree / forest parameters
 - Selection criteria

Object classification

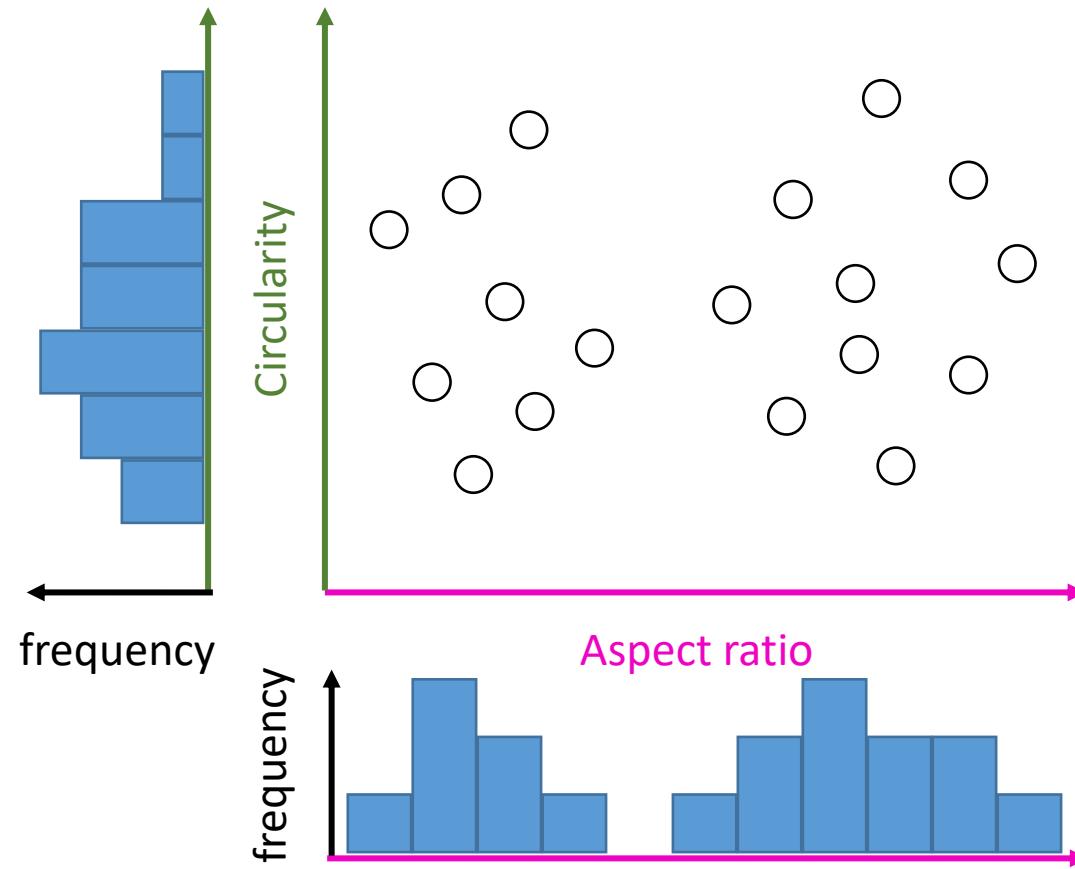
- Typical use cases
 - Shape differentiation
 - Cell-cycle differentiation
 - Eliminate wrong / bad segmentations

Warning: If you exclude nuclei depending on their shape, you're not allowed to analyse shape further.



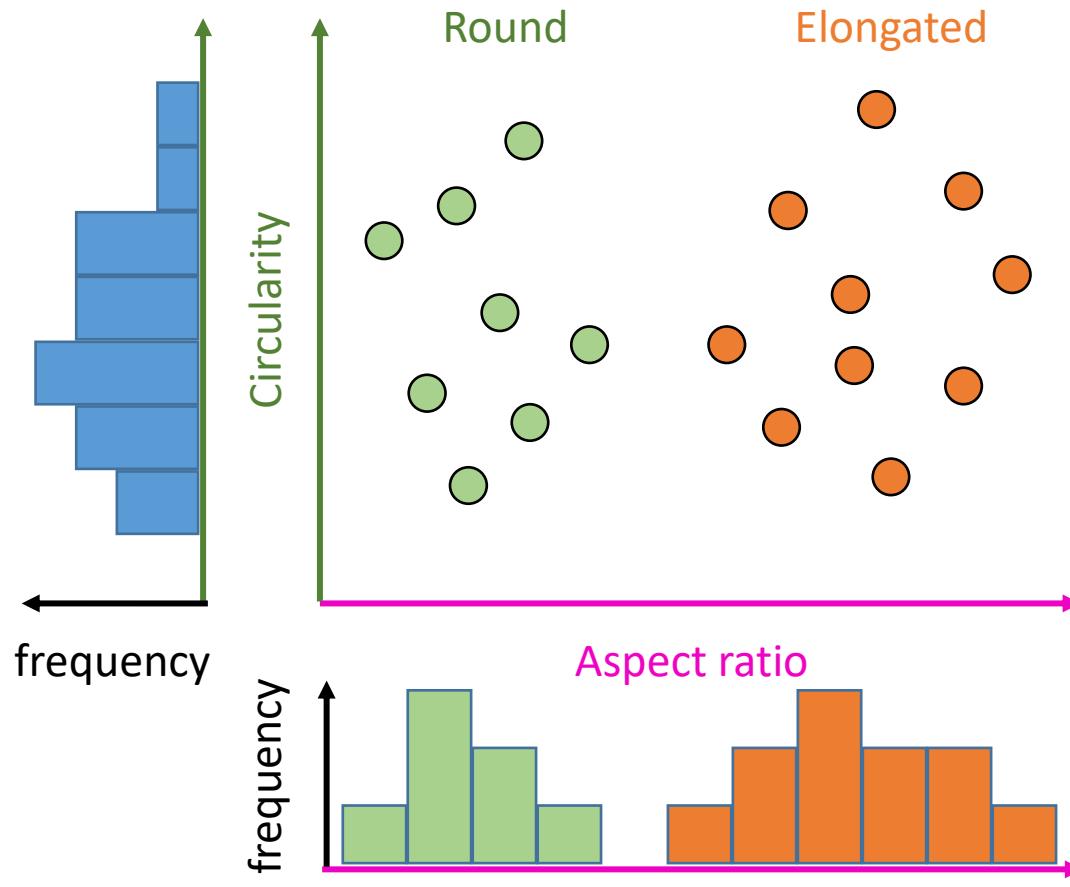
Unsupervised machine learning

- If you don't provide ground truth, the algorithm is *unsupervised*.



Unsupervised machine learning

- If you don't provide ground truth, the algorithm is unsupervised.
- Nevertheless, algorithms can tell us something about the data

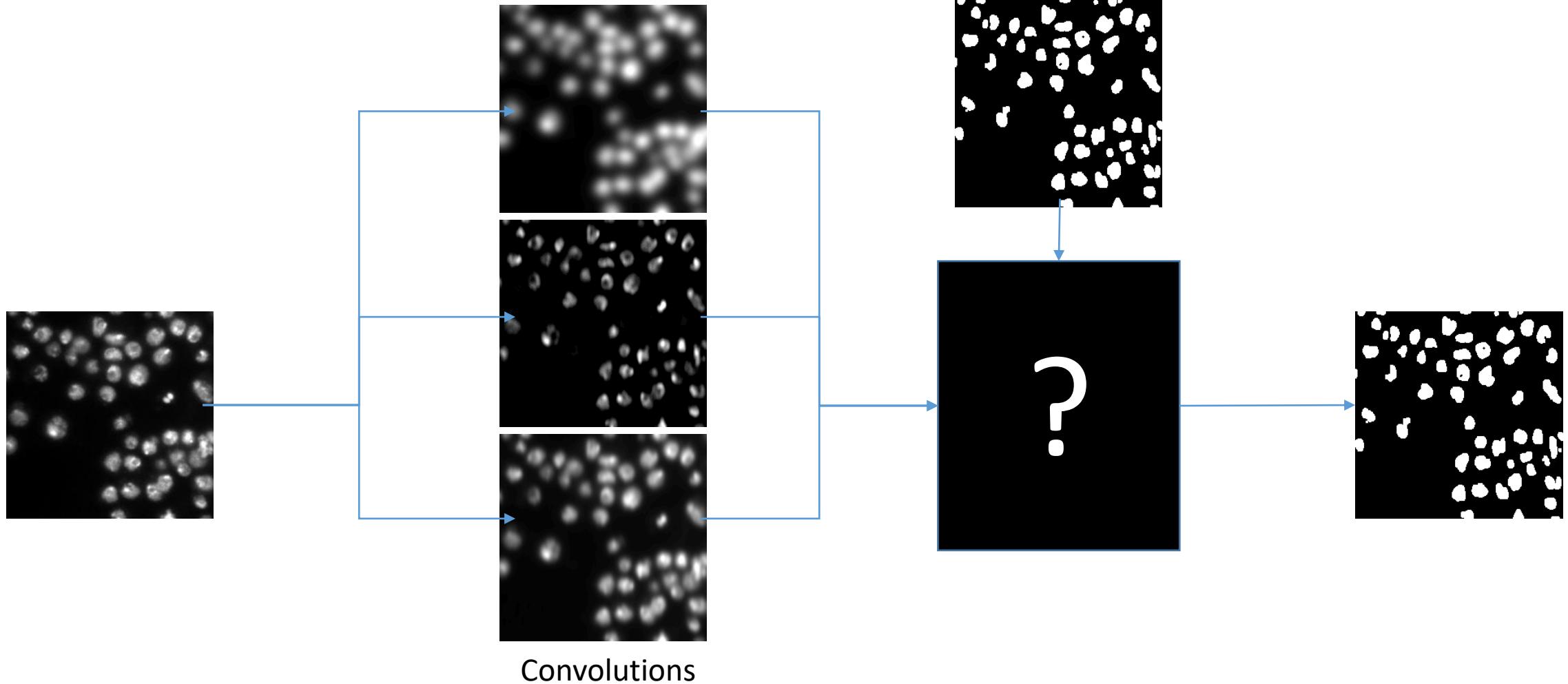


Further reading:

- Principal component analysis
- Cluster analysis

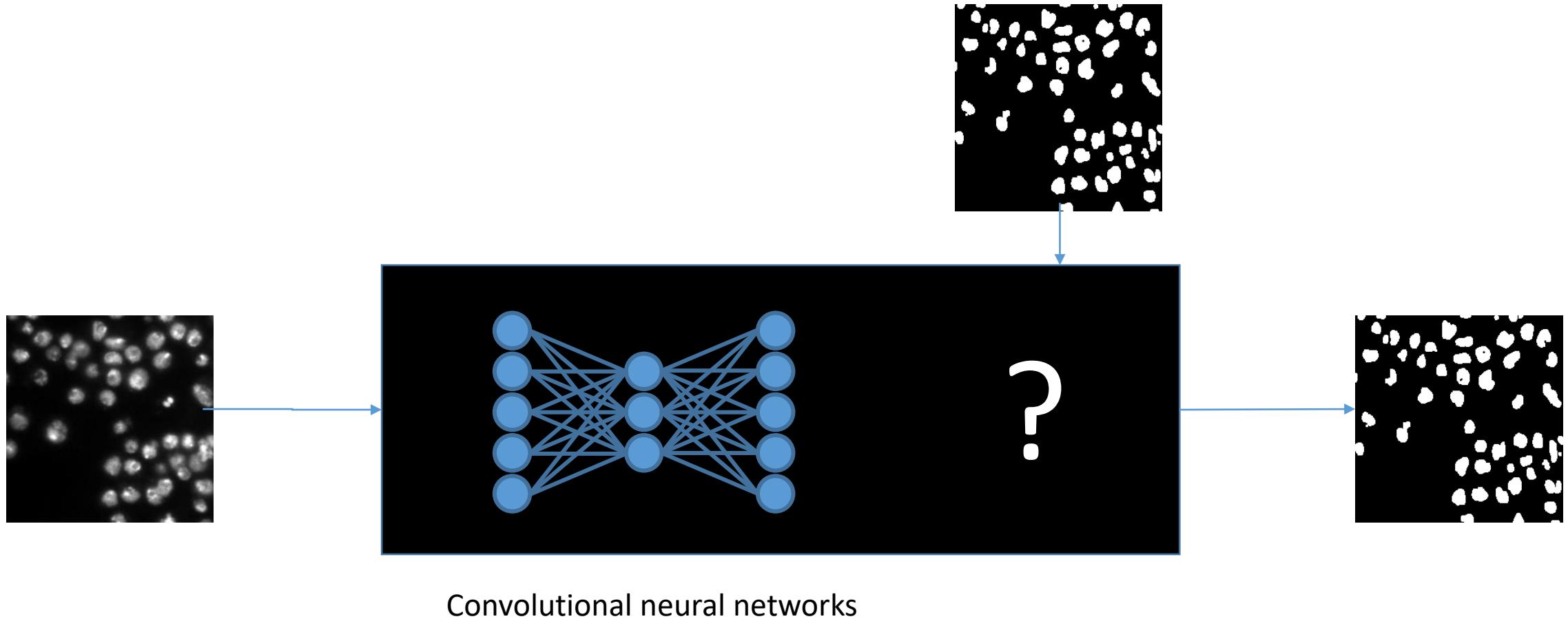
Machine learning for image analysis

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



Deep learning for image analysis

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



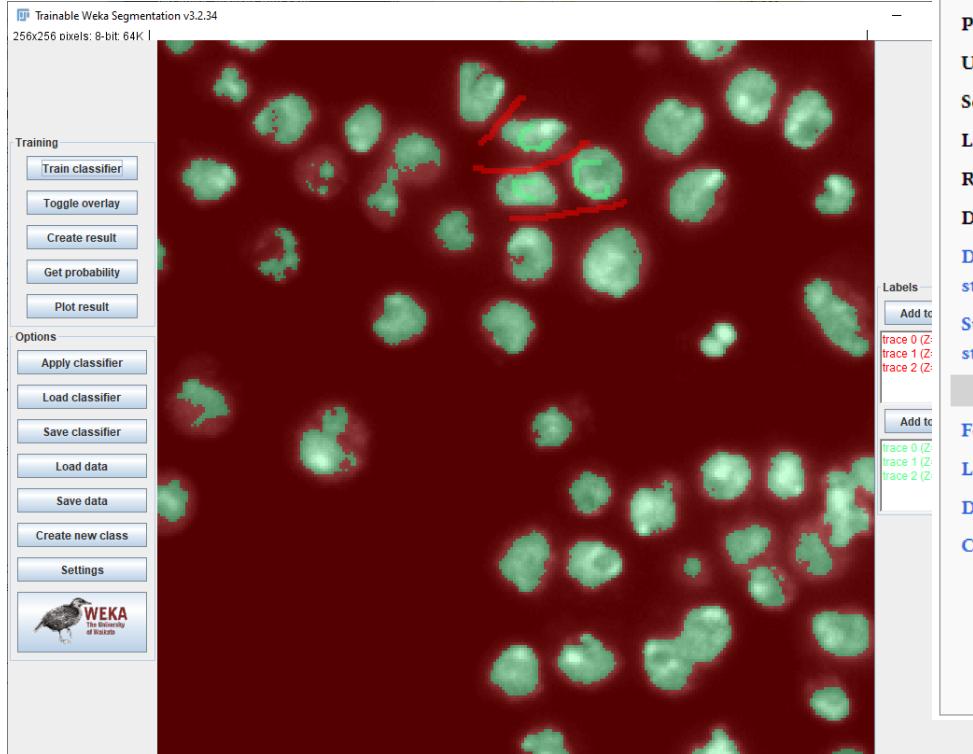
Machine Learning for Pixel Classification in Fiji

Robert Haase

With Material from
Ignacio Arganda-Carreras, Universidad del País Vasco
Matthias Arzt, Jug/Myers labs MPI CBG

Trainable Weka Segmentation

- Fiji plugin for machine learning based image segmentation / pixel classification
- **Waikato Environment for Knowledge Analysis (Weka)**, developed at the University of Waikato, New Zealand
- http://imagej.net/Trainable_Weka_Segmentation

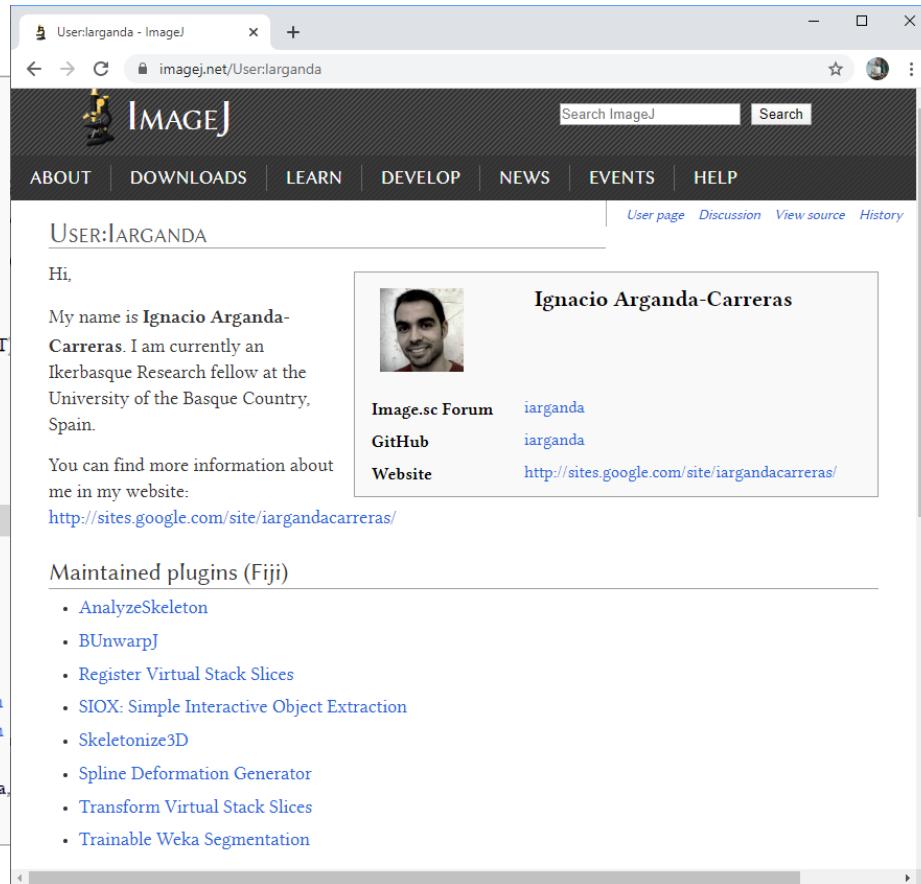


Trainable Segmentation

Project	Fiji
URL	https://imagej.net/Trainable_Segmentation
Source	on GitHub
License	GPLv3
Release	3.2.33
Date	Mon May 13 2019 12:01:00 GMT-0100 (CEST)
Development status	Active
Support status	Active

TEAM

Founders	Verena Kaynig, Johannes Schindelin
Leads	Ignacio Arganda-Carreras
Developers	Ignacio Arganda-Carreras
Contributors	Verena Kaynig, Johannes Schindelin, Albert Cardona, Jan Eglinger, Patrice Freydiere, Jan Funke, Mark Hiner, Larry Lindsey, Christian Tischer, Eibe Frank, Richard Kirkby, Julien Prados, Fran Supek, Len Trigg, Santi Villalba, Yong Wang



User:iarganda - ImageJ <https://imagej.net/User:iarganda>

ABOUT DOWNLOADS LEARN DEVELOP NEWS EVENTS HELP

User page Discussion View source History

USER: IARGANDA

Hi,

My name is Ignacio Arganda-Carreras. I am currently an Ikerbasque Research fellow at the University of the Basque Country, Spain.

You can find more information about me in my website: <http://sites.google.com/site/iargandacarreras/>

Ignacio Arganda-Carreras

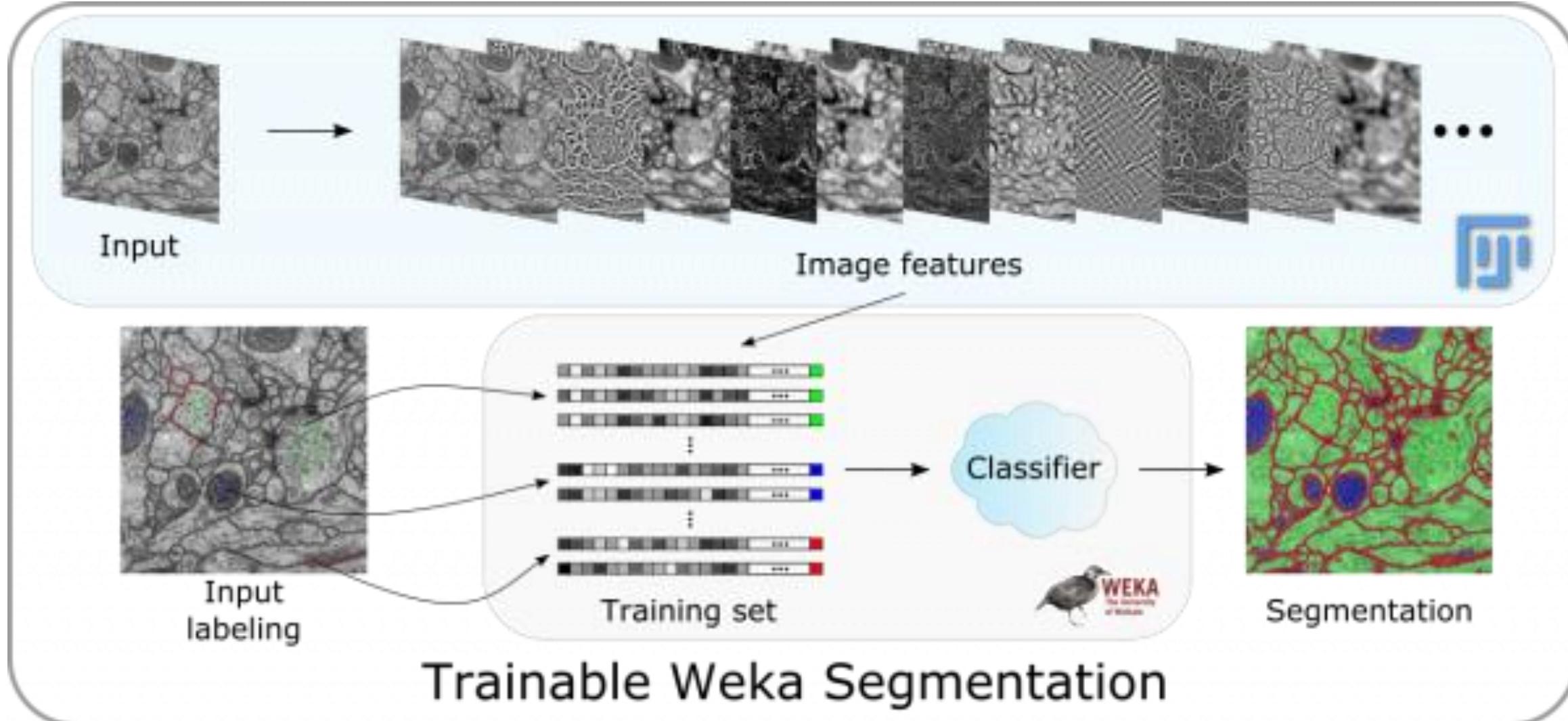


Image.sc Forum iarganda
GitHub iarganda
Website <http://sites.google.com/site/iargandacarreras/>

Maintained plugins (Fiji)

- AnalyzeSkeleton
- BUnwarpJ
- Register Virtual Stack Slices
- SIOX: Simple Interactive Object Extraction
- Skeletonize3D
- Spline Deformation Generator
- Transform Virtual Stack Slices
- Trainable Weka Segmentation

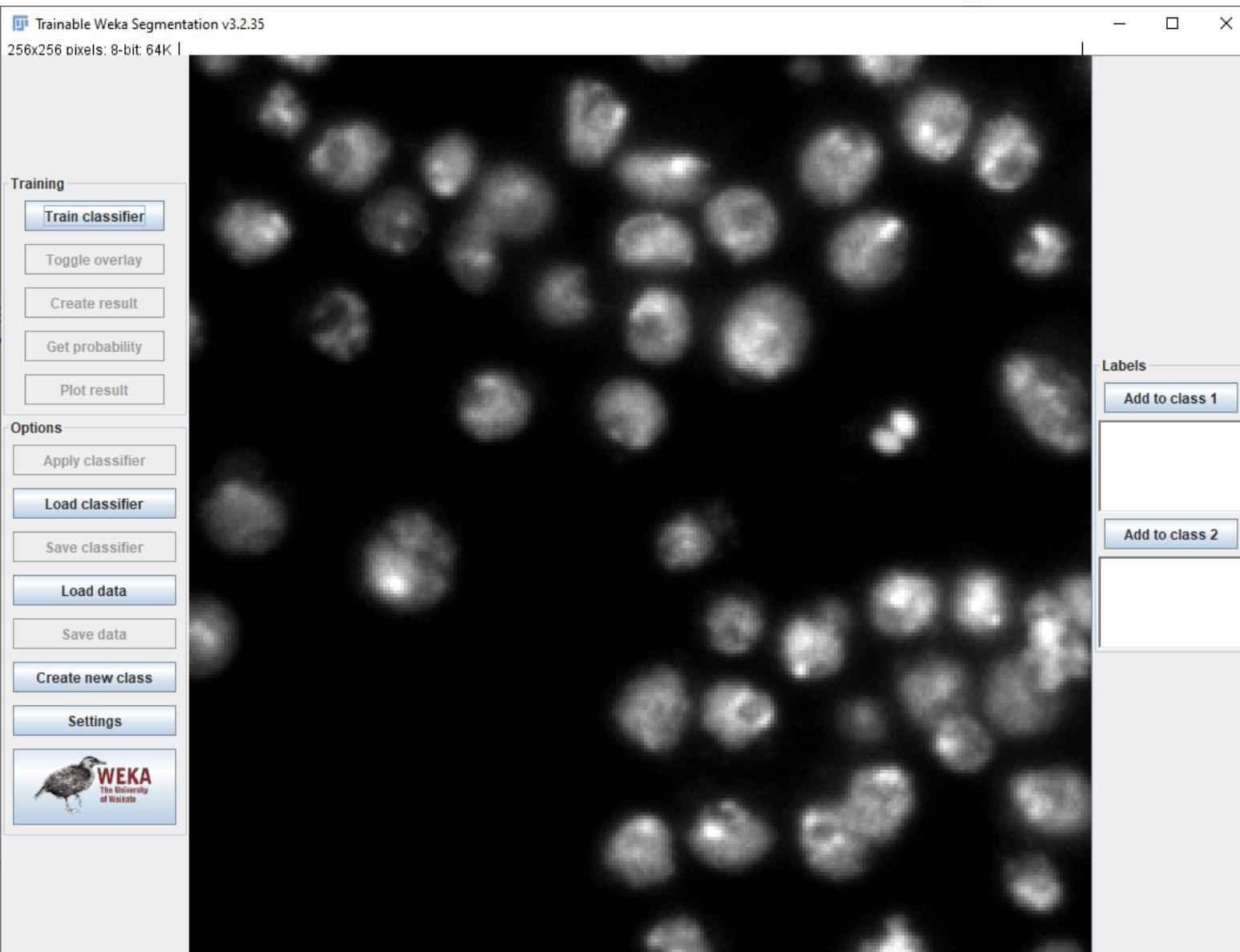
Trainable Weka Segmentation



Source: http://imagej.net/Trainable_Weka_Segmentation

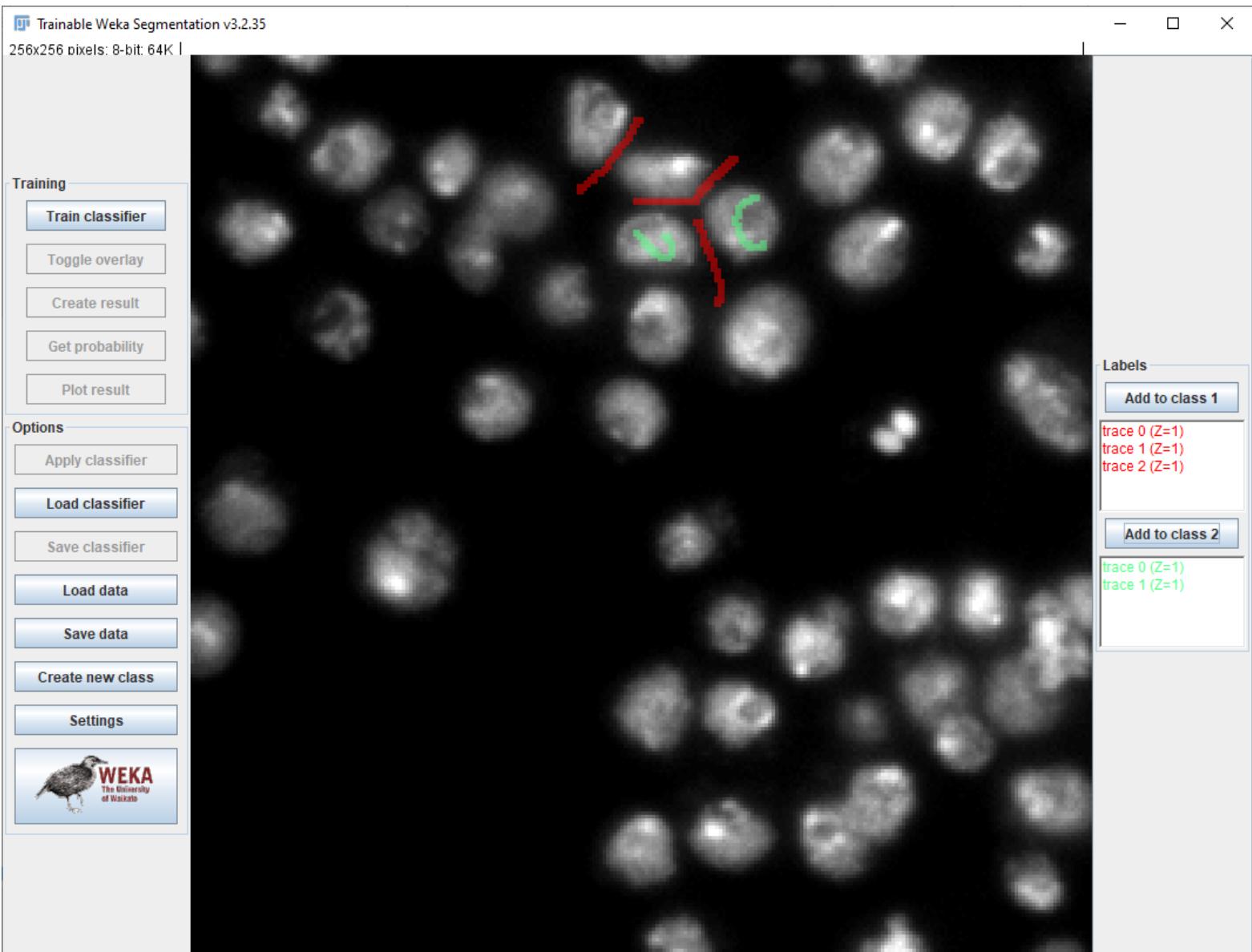
Random Forest Pixel Classifiers

- In Practice
- Plugins > Segmentation > Trainable Weka Segmentation



Random Forest Pixel Classifiers

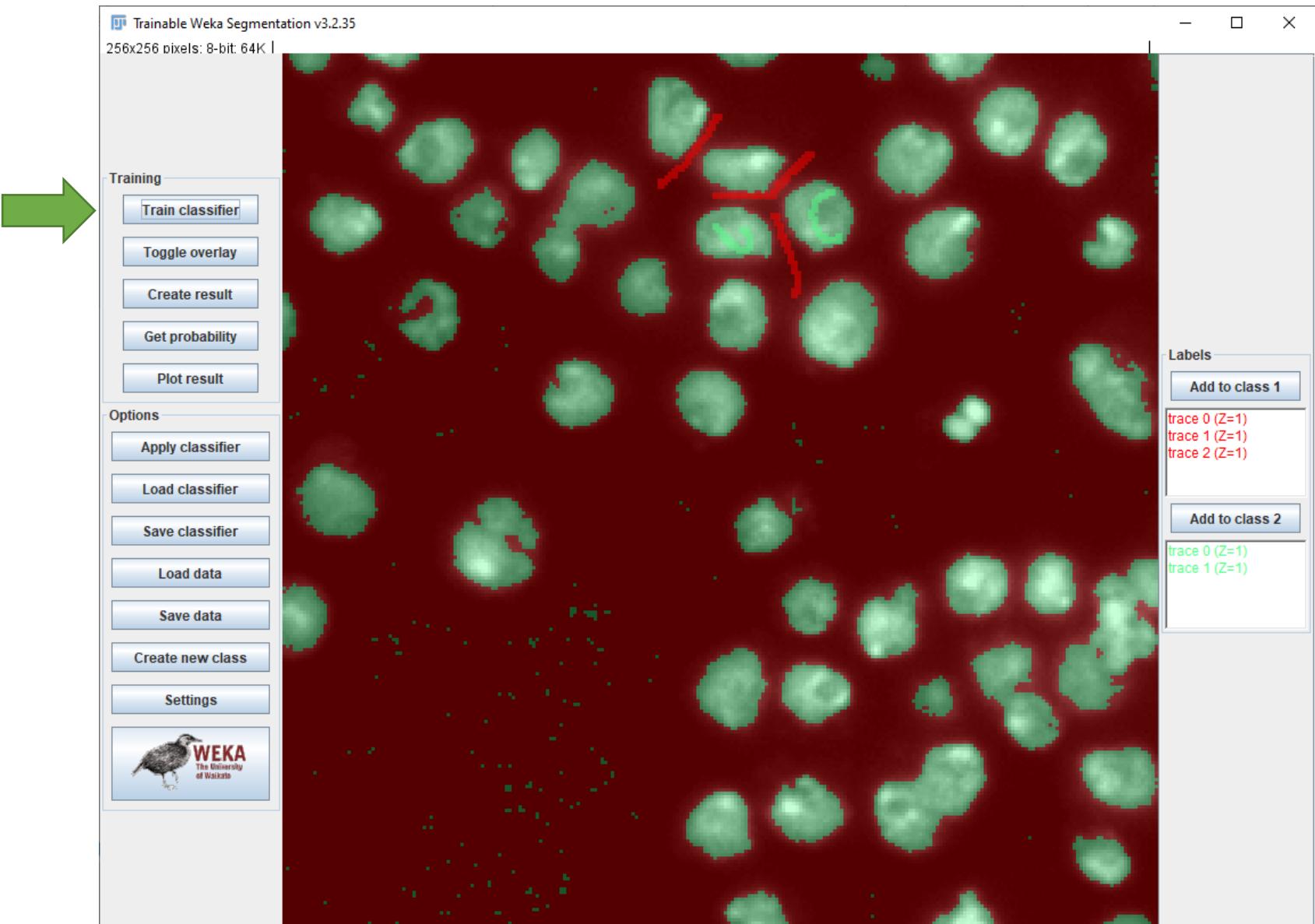
- In Practice
- Manually classify some pixels by drawing lines and adding them to class 1 or class 2



Random Forest Pixel Classifiers

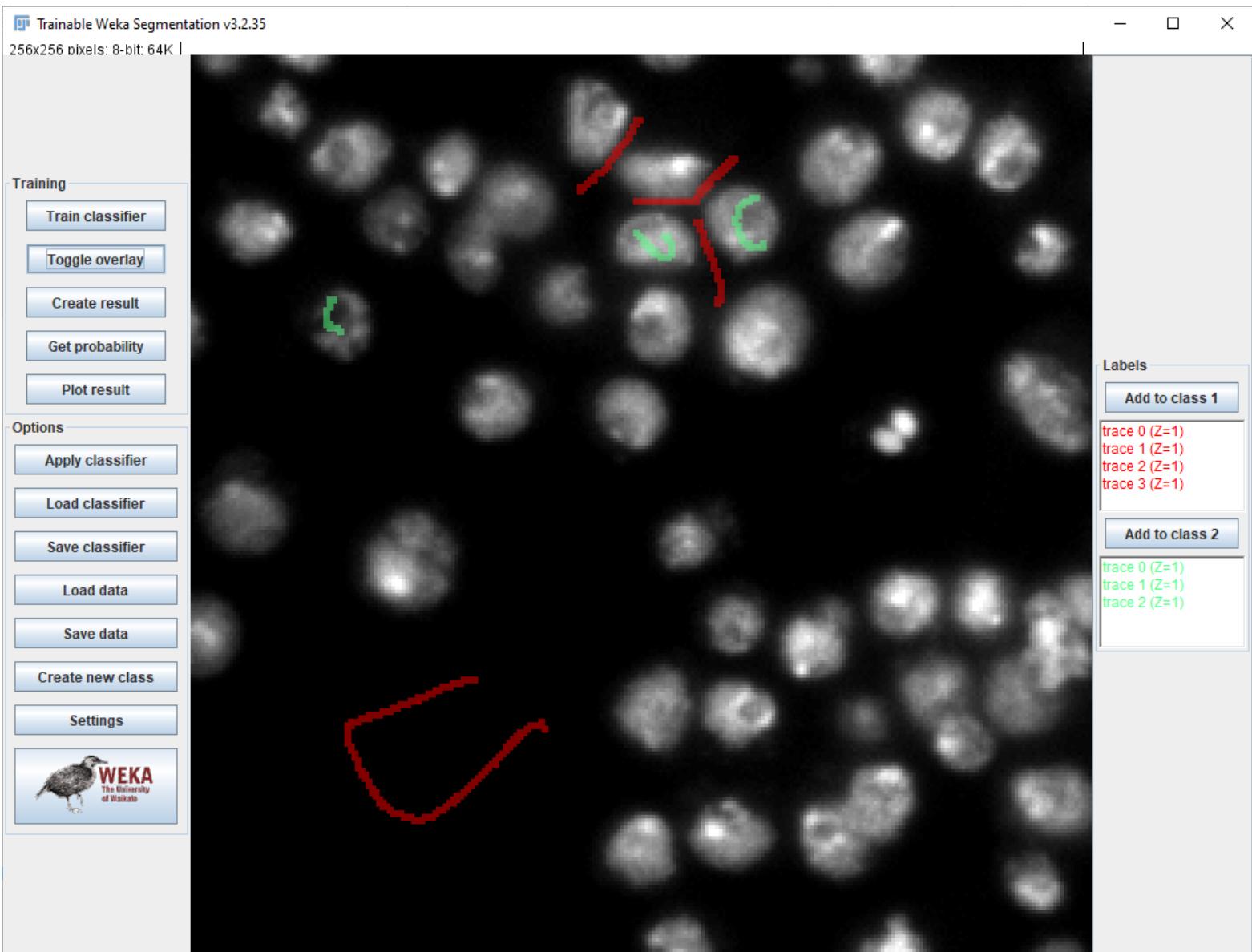
- In Practice

- Train a classifier!



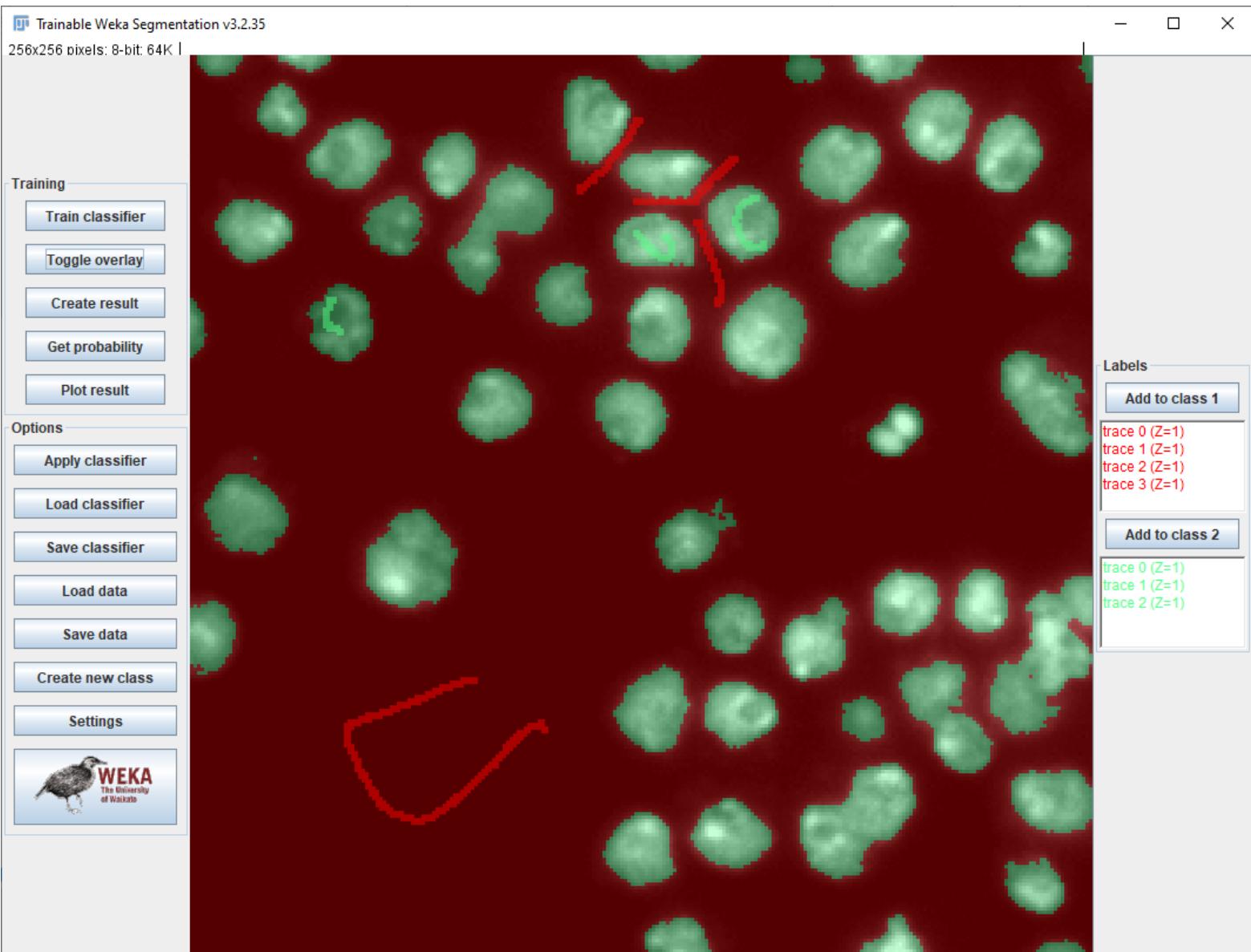
Random Forest Pixel Classifiers

- In Practice
- Refine manual annotation to guide the training



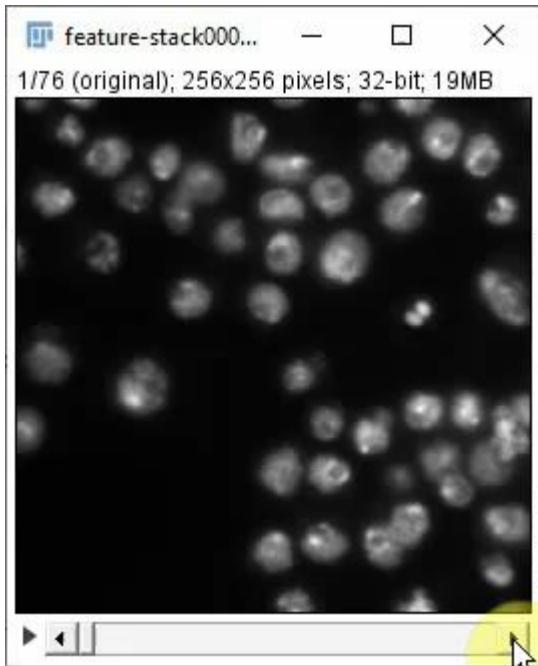
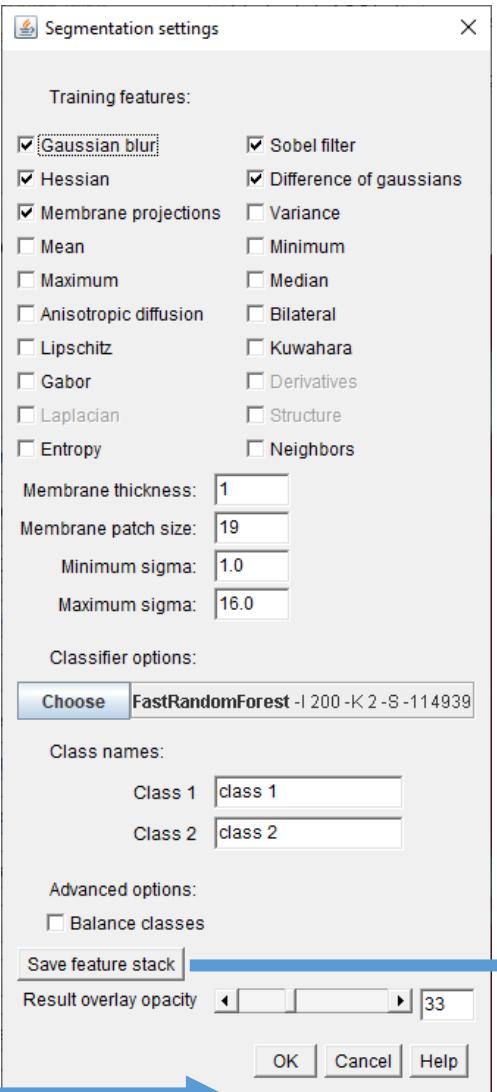
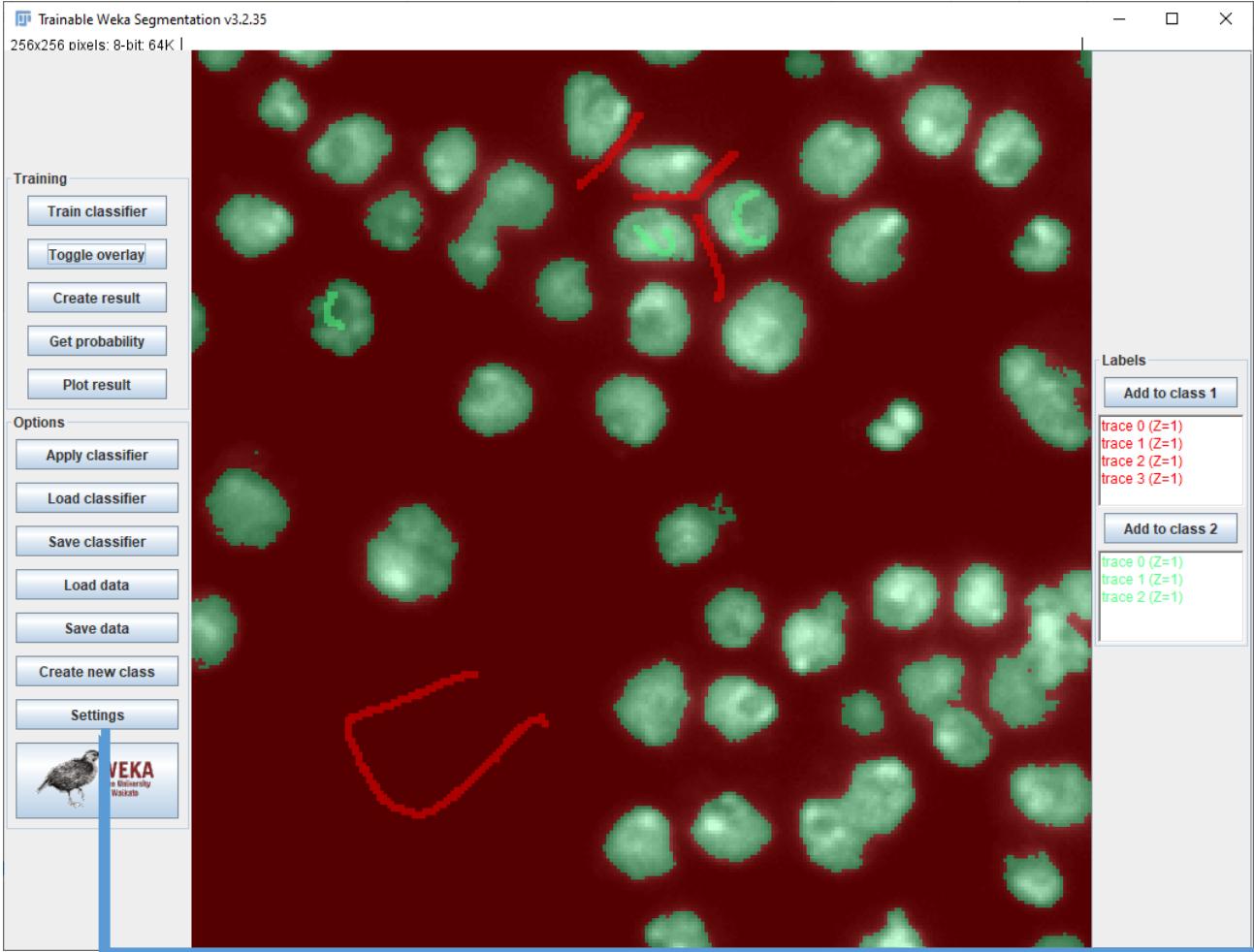
Random Forest Pixel Classifiers

- In Practice
- Train an improved classifier



Random Forest Pixel Classifiers

- Inspect the feature stack



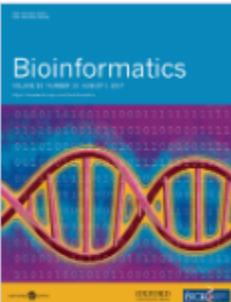
- If you use it, please cite it:
 - <https://academic.oup.com/bioinformatics/article/33/15/2424/3092362>

 Sign In ▾ Register

Bioinformatics

 INTERNATIONAL SOCIETY FOR COMPUTATIONAL BIOLOGY

Issues Advance articles Submit ▾ Purchase Alerts About ▾ All Bioinformatics ▾ Advanced Search


Volume 33, Issue 15
01 August 2017

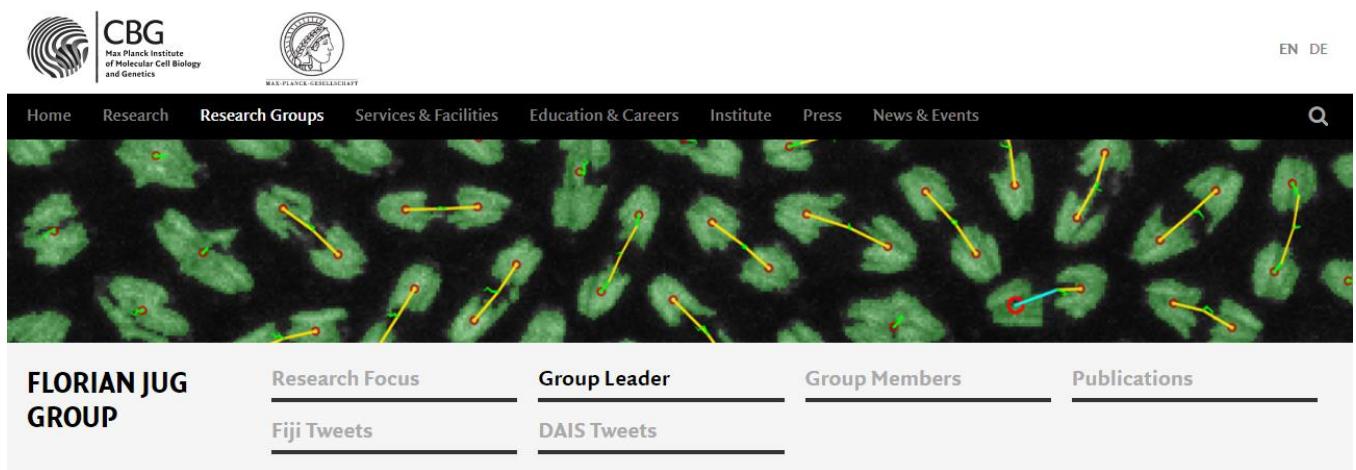
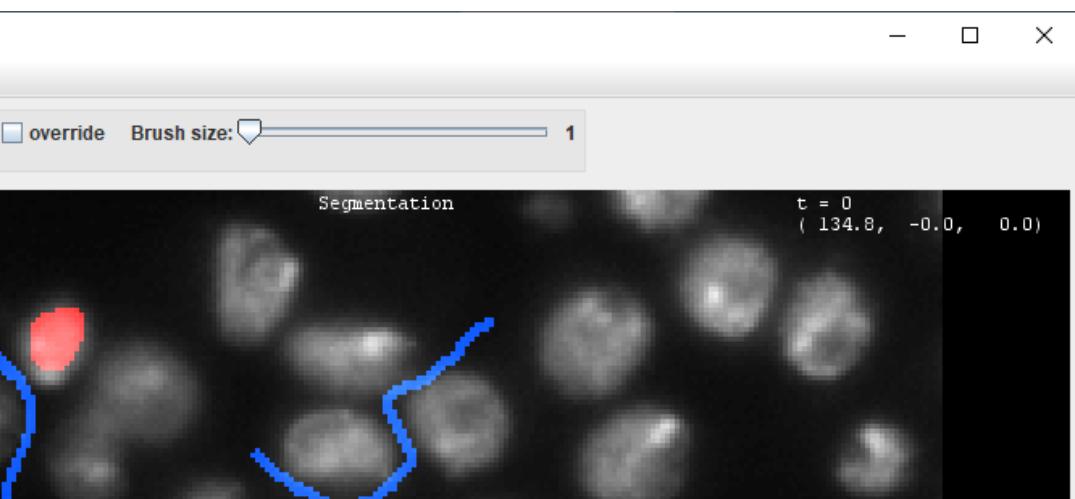
Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification

Ignacio Arganda-Carreras , Verena Kaynig, Curtis Rueden, Kevin W Eliceiri, Johannes Schindelin, Albert Cardona, H Sebastian Seung

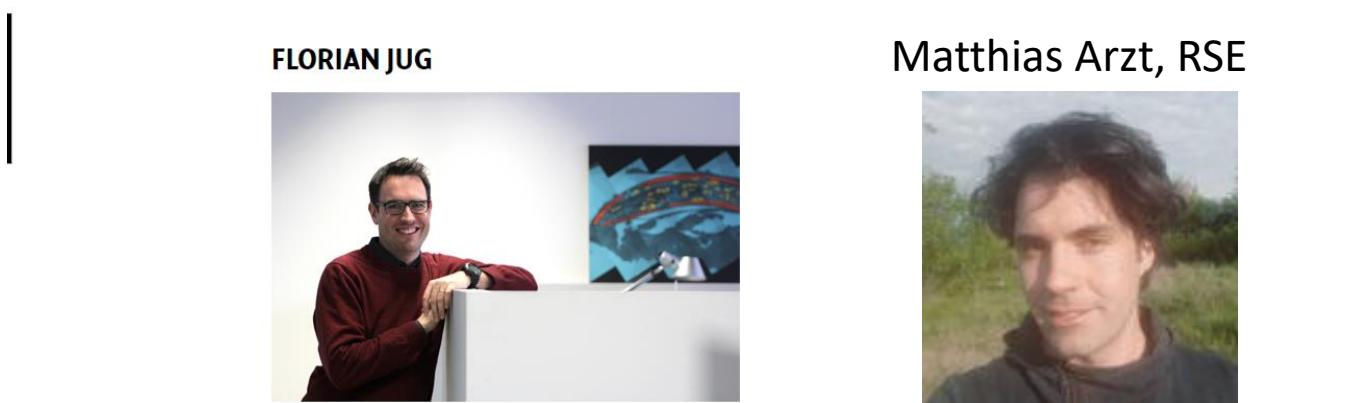
Bioinformatics, Volume 33, Issue 15, 01 August 2017, Pages 2424–2426,
<https://doi.org/10.1093/bioinformatics/btx180>

Published: 30 March 2017 **Article history** ▾

- “Labeling kit”: A Fiji plugin for annotating objects in big image data (> larger than RAM)
- Weka based pixel classification

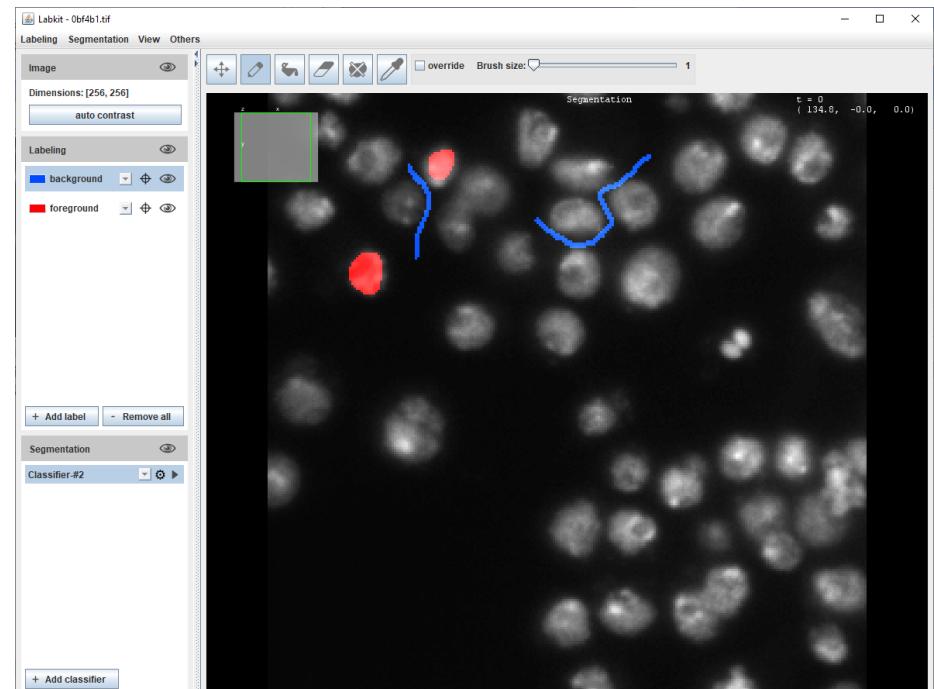
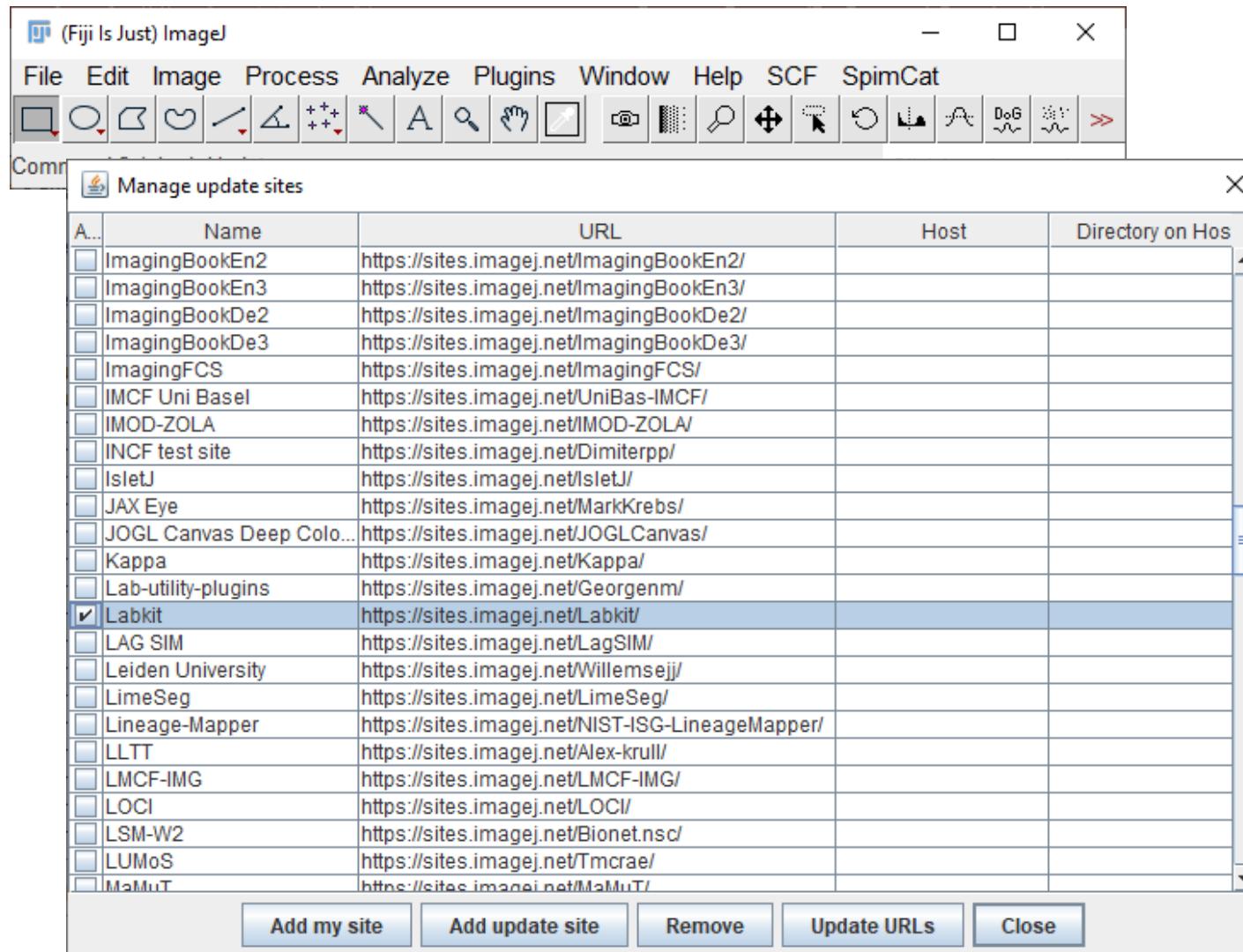


The screenshot shows the website for the Max Planck Institute of Molecular Cell Biology and Genetics (CBG). The header includes the CBG logo and the text "Max Planck Institute of Molecular Cell Biology and Genetics". There are links for Home, Research, Research Groups, Services & Facilities, Education & Careers, Institute, Press, and News & Events. On the right, there are language links EN DE and a search icon. The main content features a fluorescence microscopy image of cells with yellow outlines and red dots. Below the image, there is a navigation bar for the "FLORIAN JUG GROUP" with sections for Research Focus, Group Leader, Group Members, and Publications. Under "Research Focus", there are links for Fiji Tweets and DAIS Tweets.

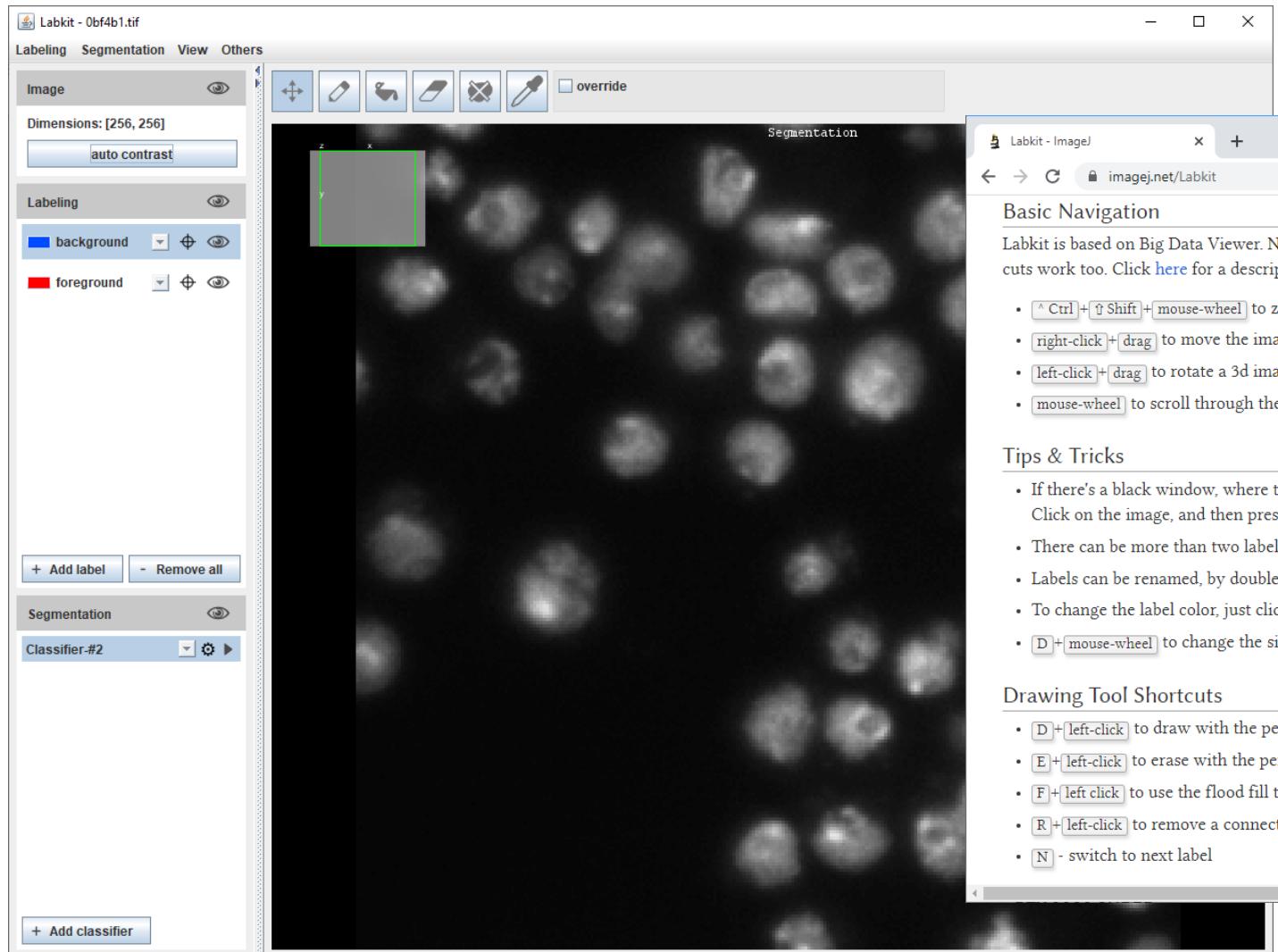


Two photographs of researchers. On the left, a man with glasses and a maroon sweater is smiling, leaning against a white desk. On the right, a man with dark hair and a black jacket is also smiling. The text "FLORIAN JUG" is above the first photo, and "Matthias Arzt, RSE" is above the second.

- Help > Update.... Manage update sites....



- Plugins > Segmentation > LabKit



- Help:
 - <https://imagej.net/Labkit>
 - <https://imagej.net/BigDataViewer>

Basic Navigation

Labkit is based on Big Data Viewer. Navigation cuts work too. Click [here](#) for a description.

- `Ctrl + Shift + mouse-wheel` to zoom
- right-click + drag to move the image
- left-click + drag to rotate a 3d image
- mouse-wheel to scroll through the z-axis

Tips & Tricks

- If there's a black window, where the image should be, Click on the image, and then press `Ctrl + S`.
- There can be more than two labels, just click on the image to add more.
- Labels can be renamed, by double clicking on them.
- To change the label color, just click on the label and then click on the color swatch.
- `D + mouse-wheel` to change the size of the selection box.

Drawing Tool Shortcuts

- `D + left-click` to draw with the pencil tool
- `E + left-click` to erase with the pencil tool
- `F + left click` to use the flood fill tool
- `R + left-click` to remove a connected component
- `N` - switch to next label

Navigation Commands

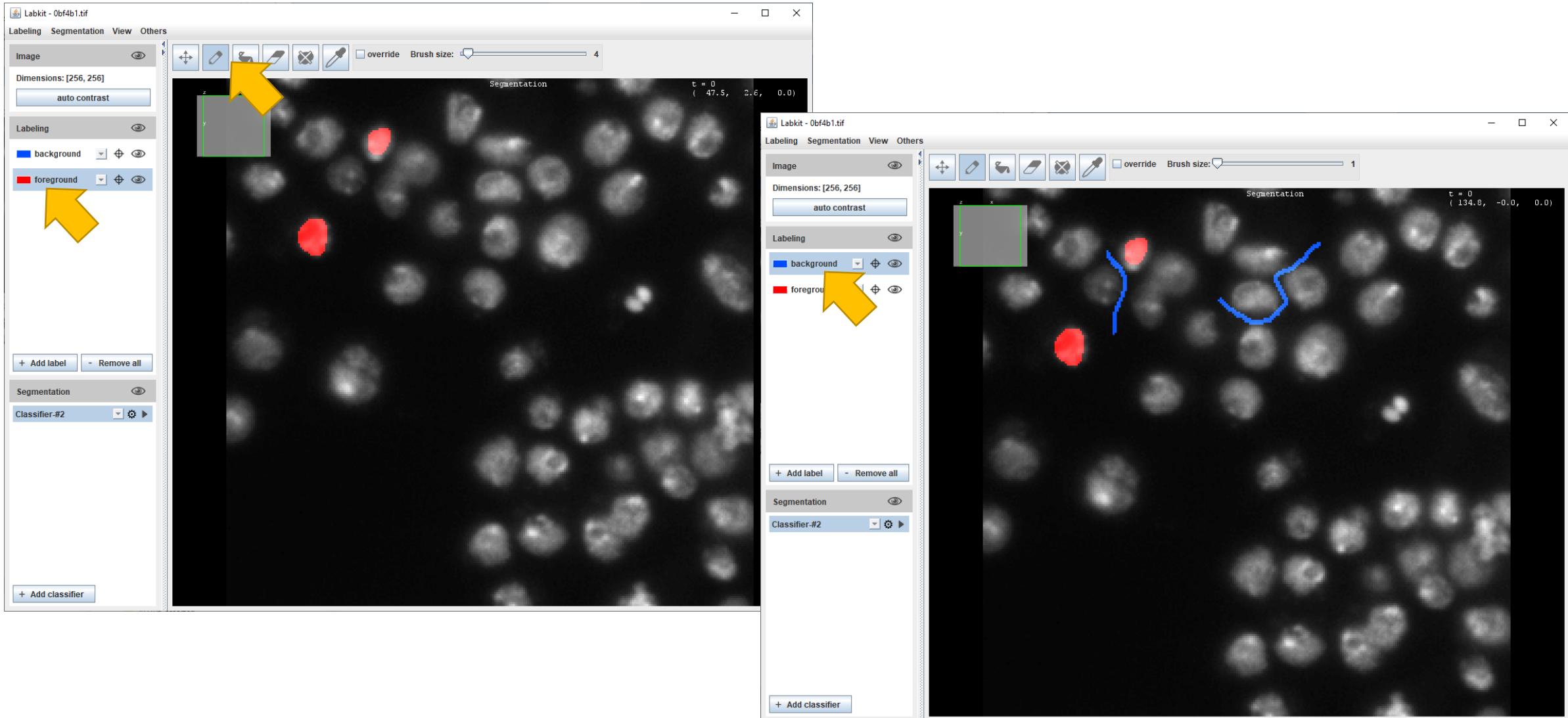
The following table shows the available navigation commands using the mouse:

<code>left-click + drag</code>	Rotate (pan and tilt) around the point where the mouse was clicked.
<code>right-click + drag</code> or <code>middle-click + drag</code>	Translate in the XY-plane.
<code>mouse-wheel</code>	Move along the z-axis.
<code>Cmd + mouse-wheel</code> or <code>Shift + Ctrl + mouse-wheel</code>	Zoom in and out.

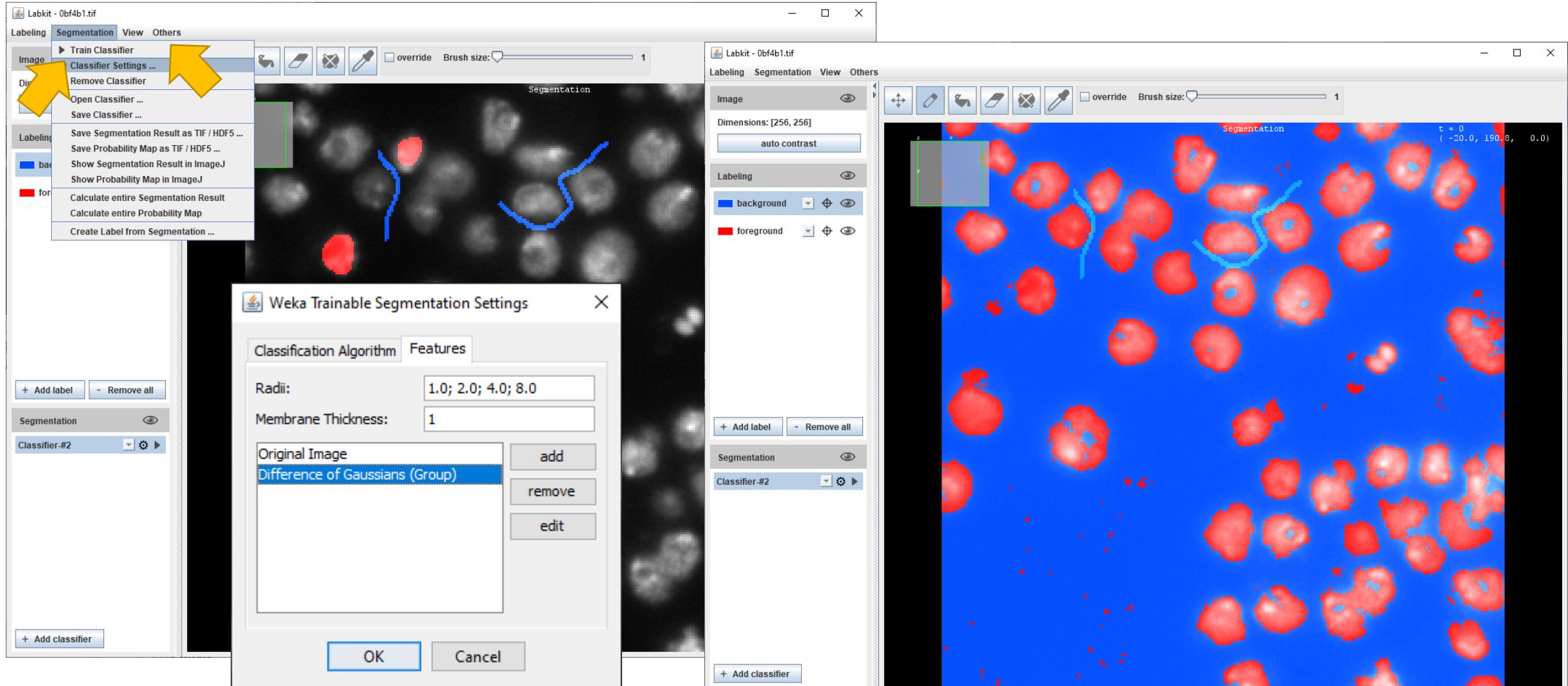
The following table shows the available navigation commands using keyboard shortcuts:

<code>X, Y, Z</code>	Select keyboard rotation axis.
<code>-, +</code>	Rotate clockwise or counter-clockwise around the chosen rotation axis.
<code>↑, ↓</code>	Zoom in or out.
<code>, .</code>	Move forward or backward along the Z-axis.
<code>Shift + X</code>	Rotate to the ZY-plane of the current source. (Look along the X-axis of the current source.)
<code>Shift + Y</code> or <code>Shift + A</code>	Rotate to the XZ-plane of the current source. (Look along the Y-axis of the current source.)
<code>Shift + Z</code>	Rotate to the XY-plane of the current source. (Look along the Z-axis of the current source.)
<code>Q or N</code>	Move to previous timepoint.

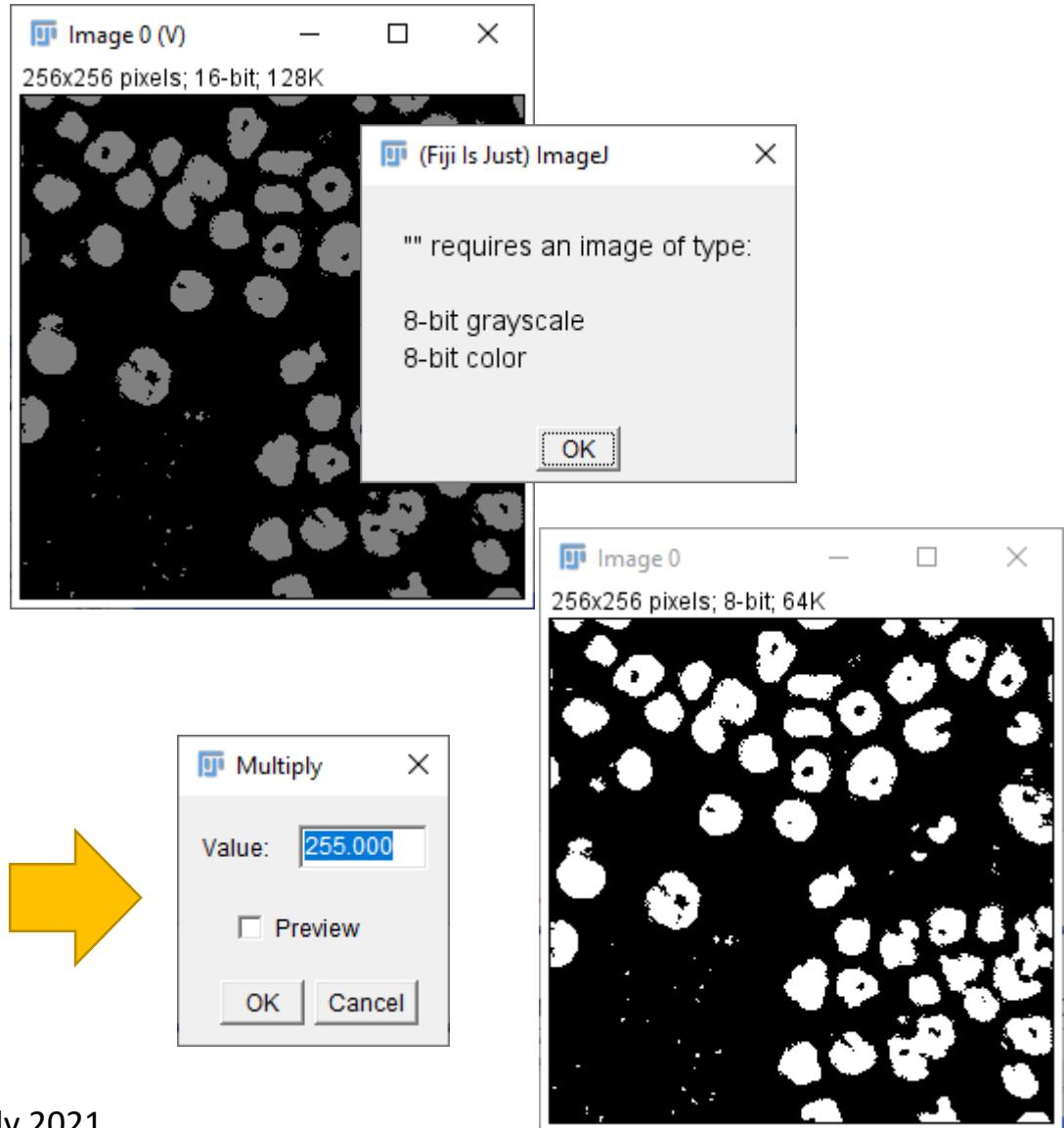
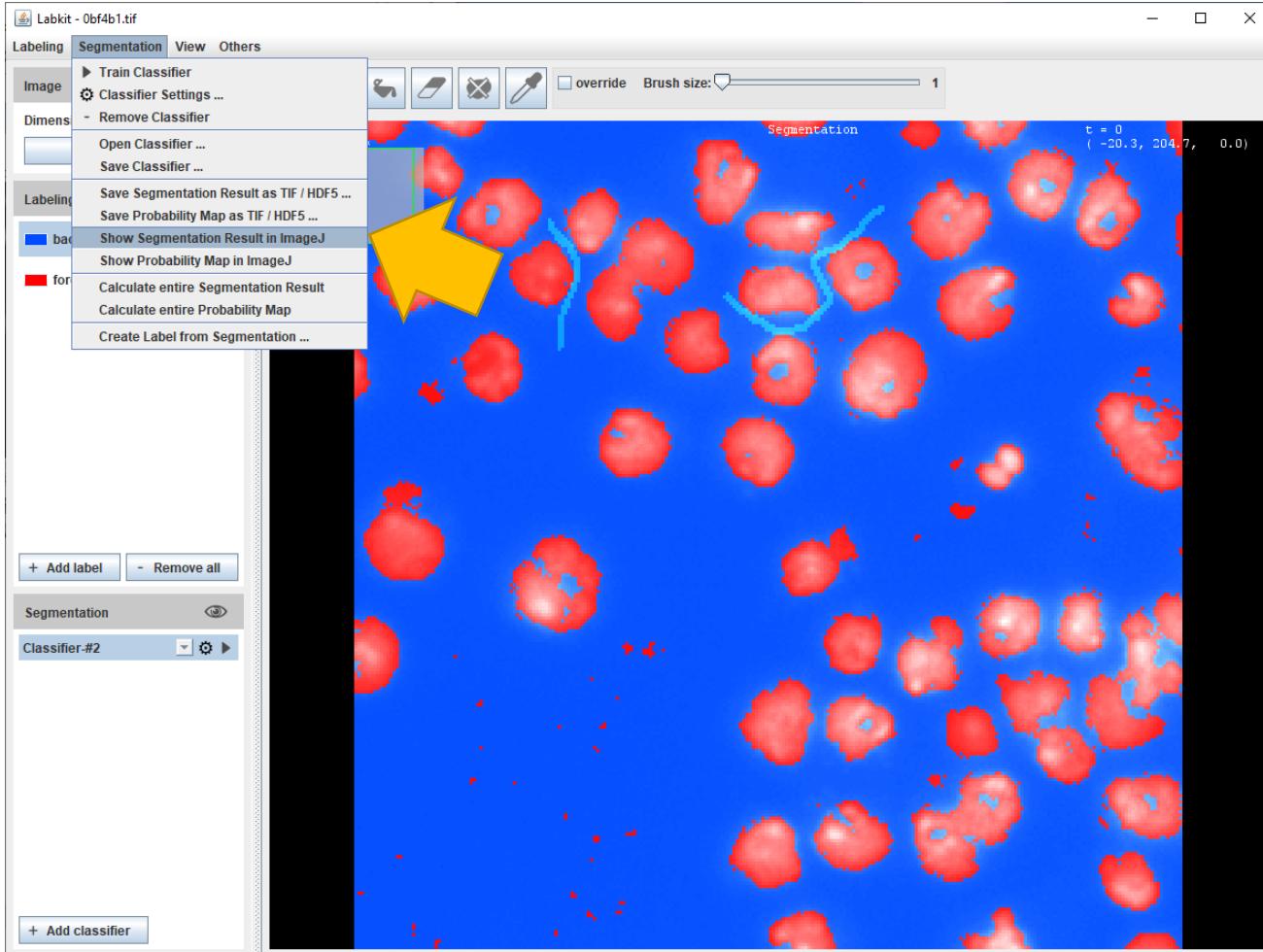
- Annotations: Foreground vs. background vs. ...



- Feature selection + Training

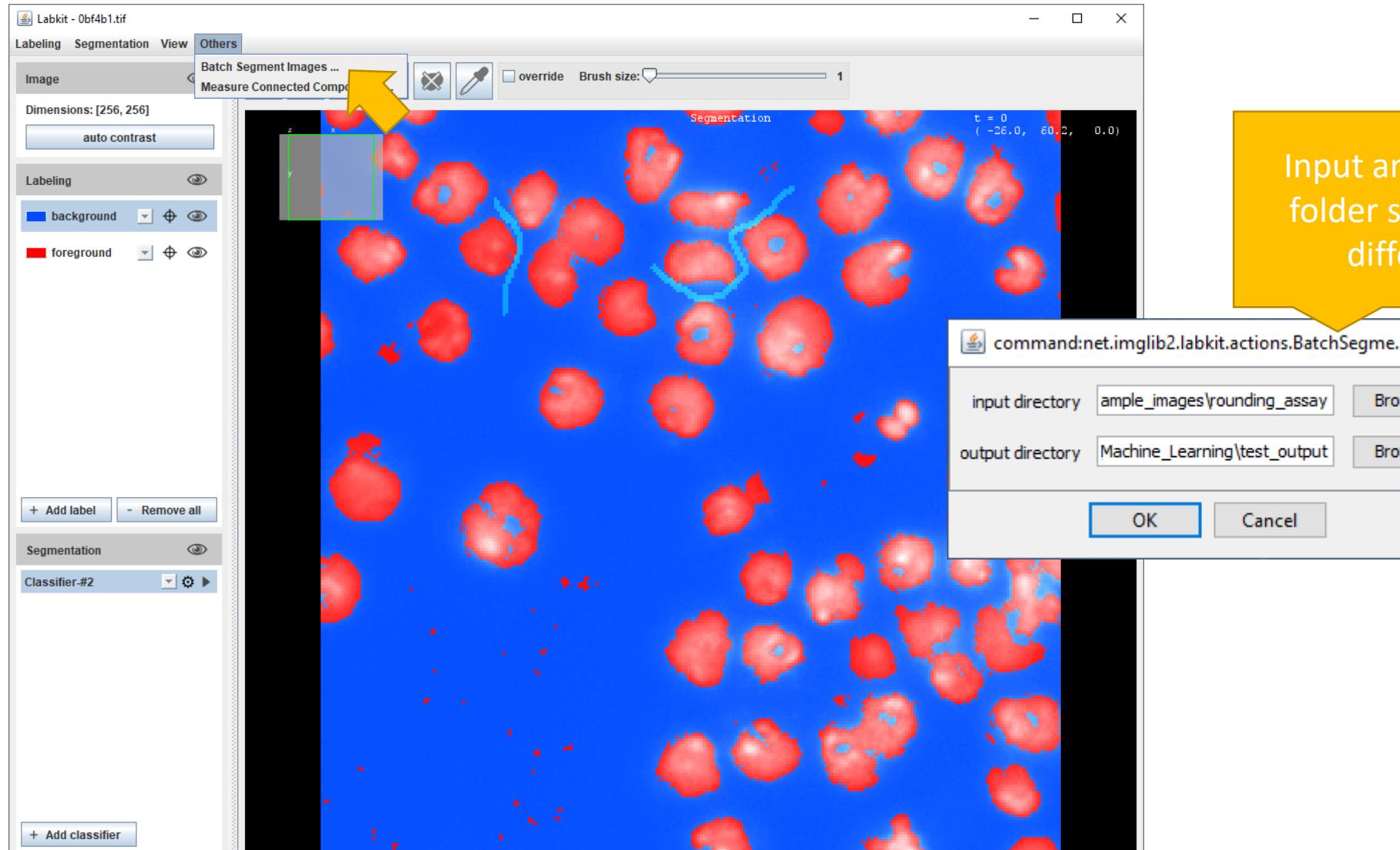


- Exporting and post-processing: Exported binary images need to be multiplied by 255 and converted to 8-bit





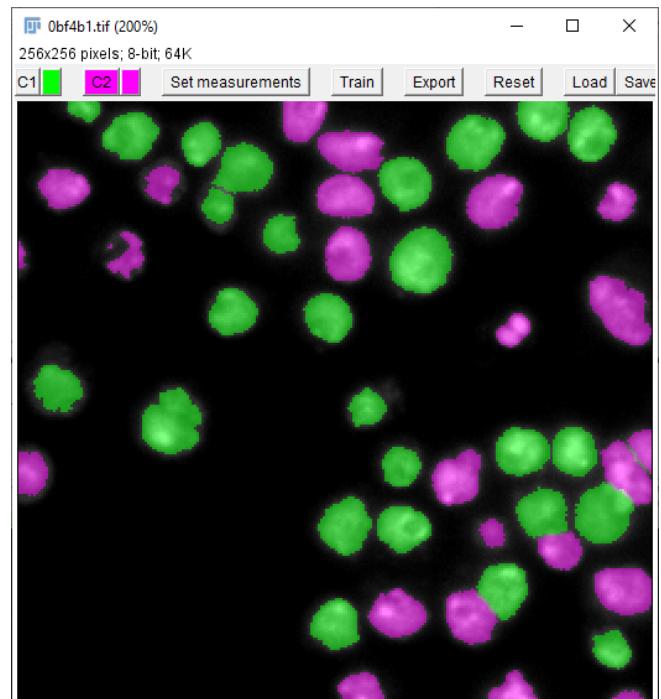
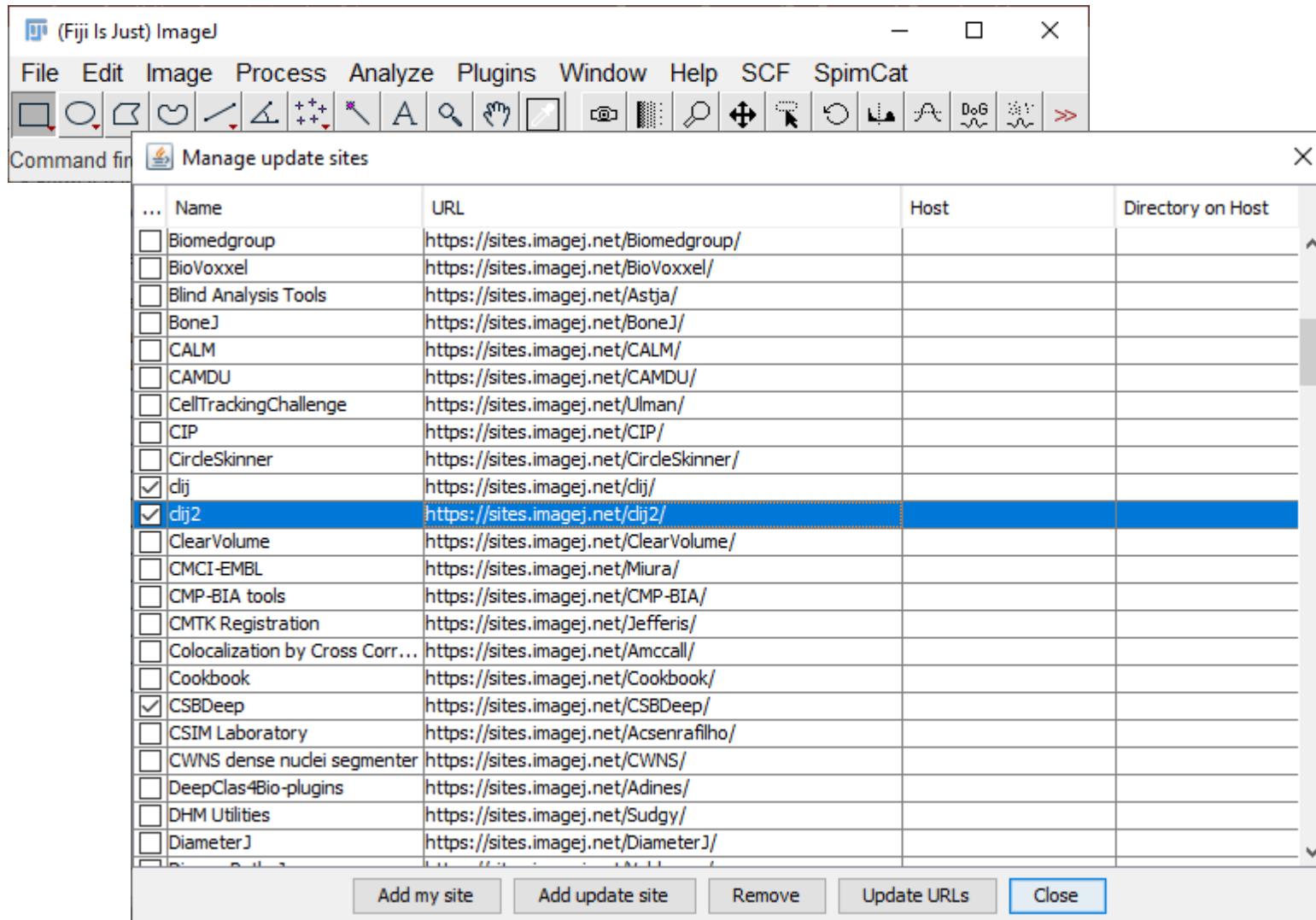
- Generate predictions of a folder of images



Input and output
folder should be
different

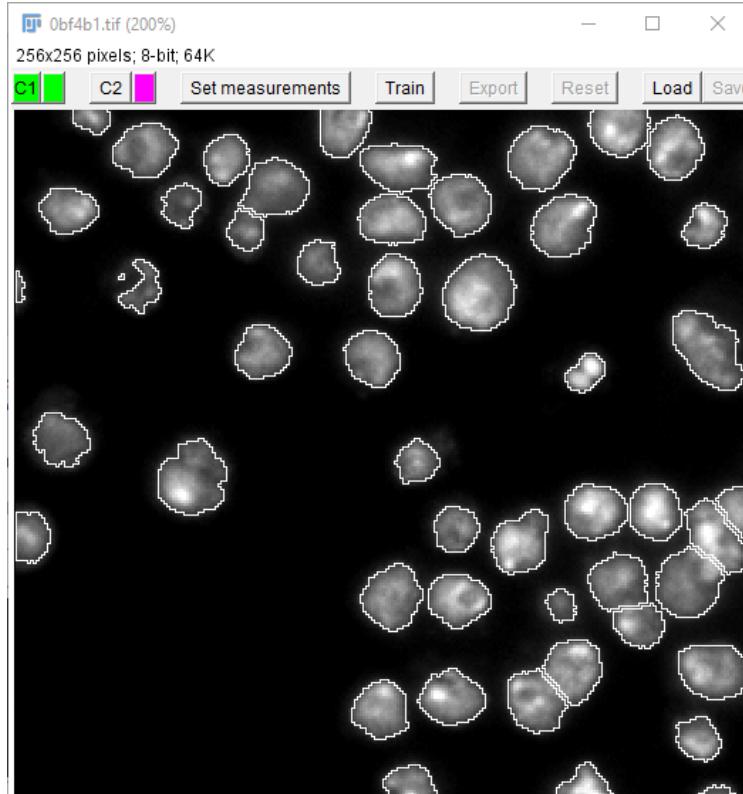
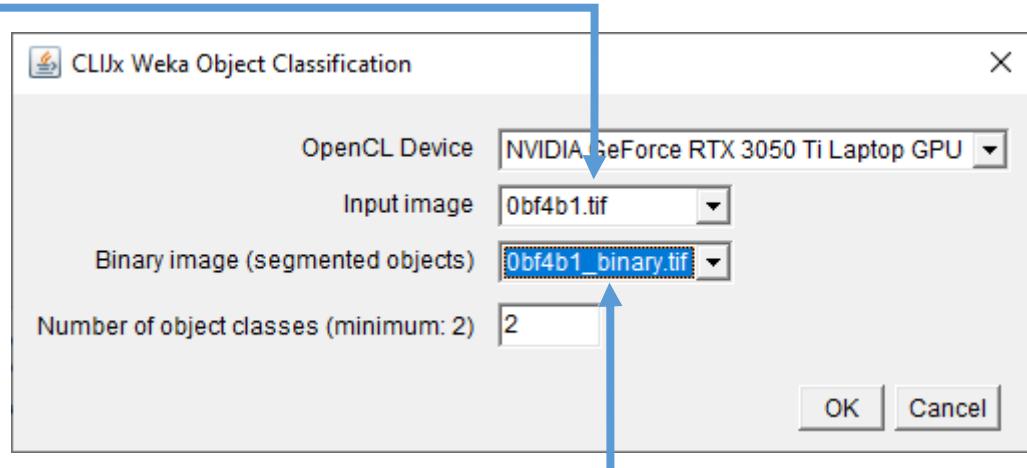
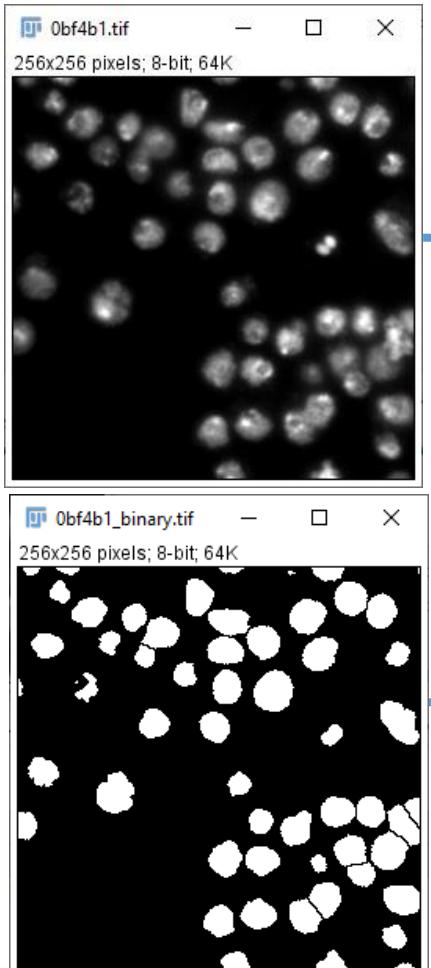
CLIJxWeka Object classification

- An academic machine learning tool for 2D Object classification in Fiji



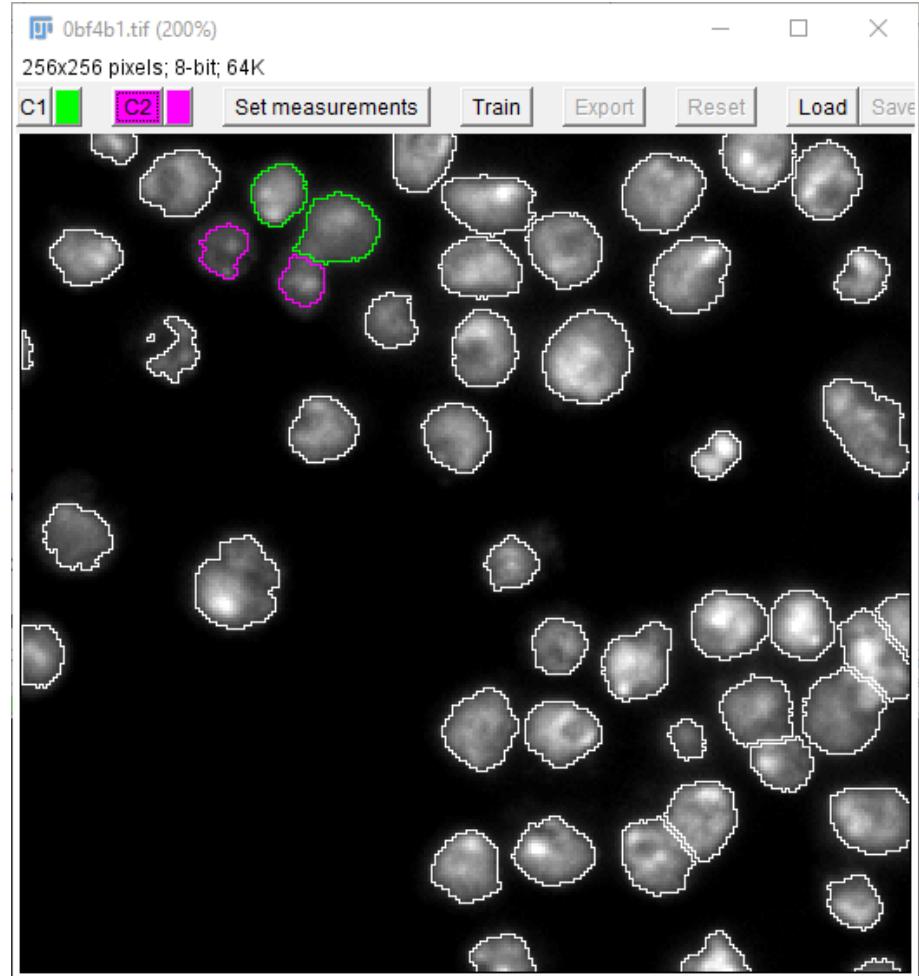
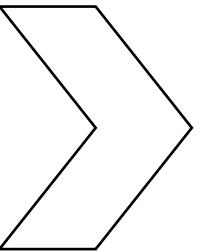
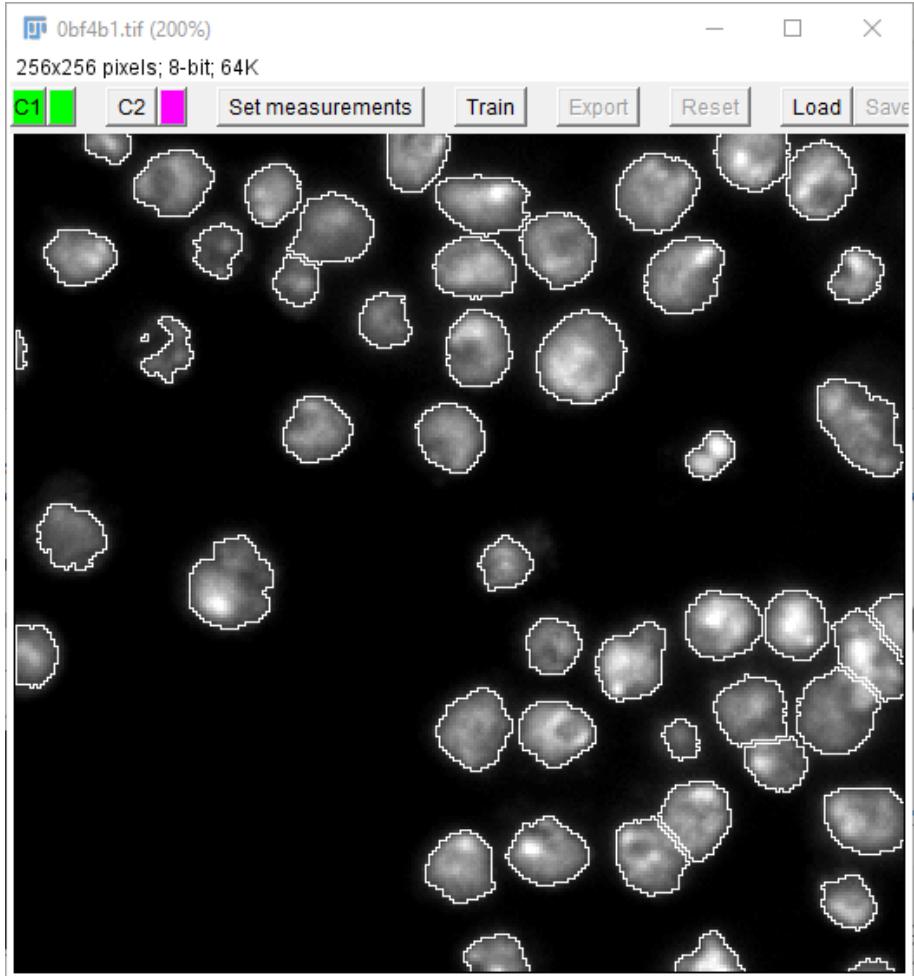
CLIJxWeka Object classification

- Input: Original image + binary image
- Plugins > Segmentation > CLIJx Weka Object Classification on GPU (CLIJx, experimental)



CLIJxWeka Object classification

- Add annotation: Click
- Remove annotation: Click again



CLIJxWeka Object classification

- Feature selection + training

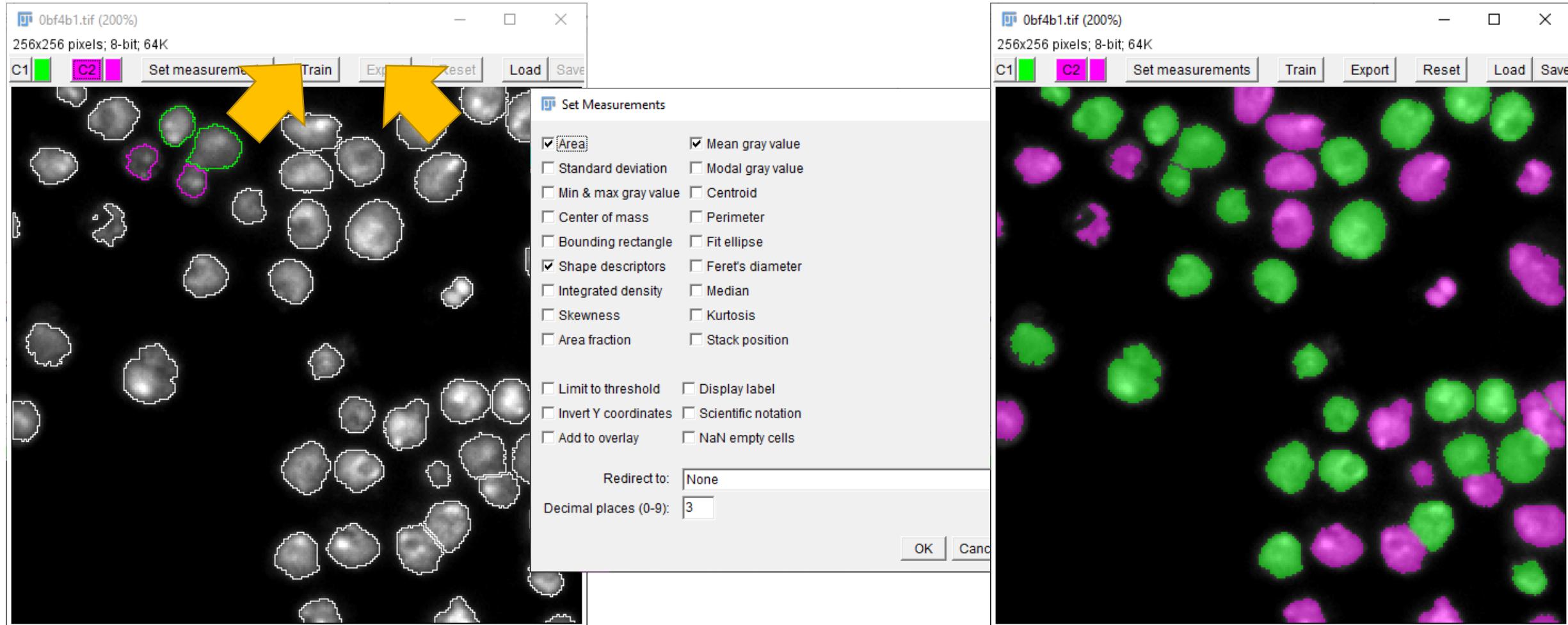


Image Segmentation in Python using scikit-learn

Robert Haase



May 2021

The scikit-learn logo is BSD3 licensed by the scikit-learn developers
https://commons.wikimedia.org/wiki/File:Scikit_learn_logo_small.svg

Interactive pixel classification

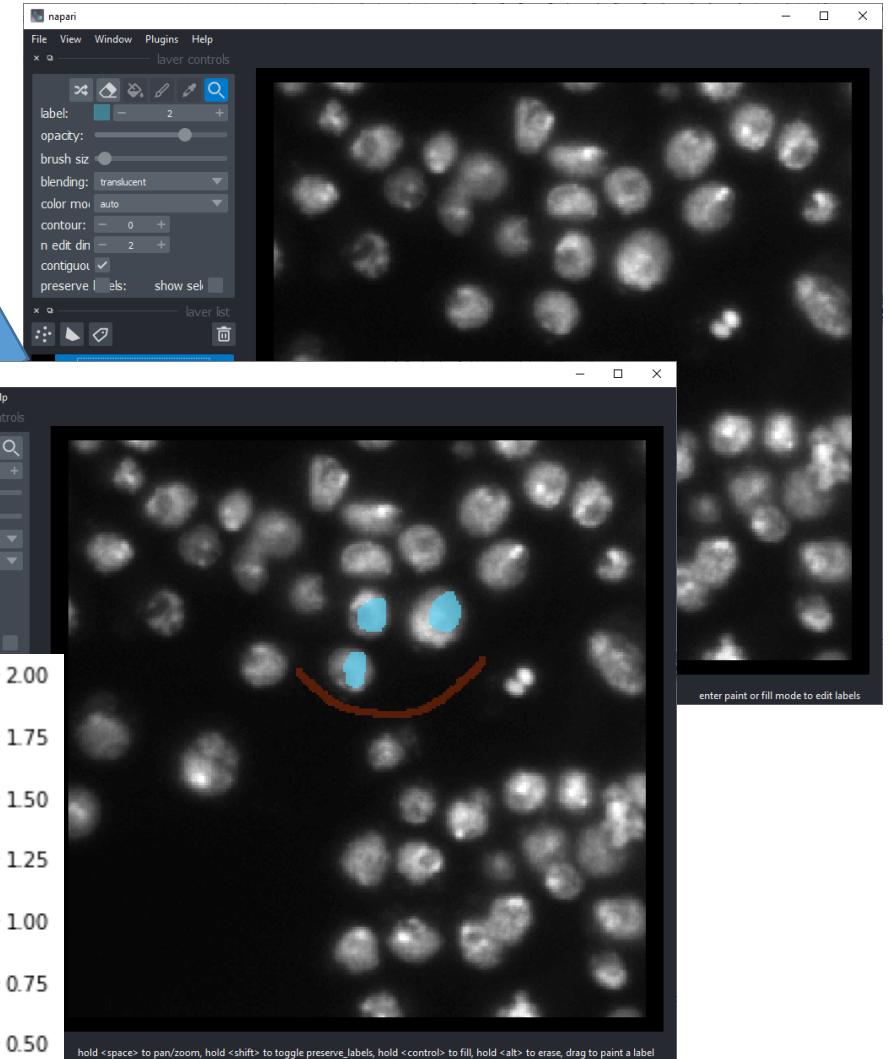
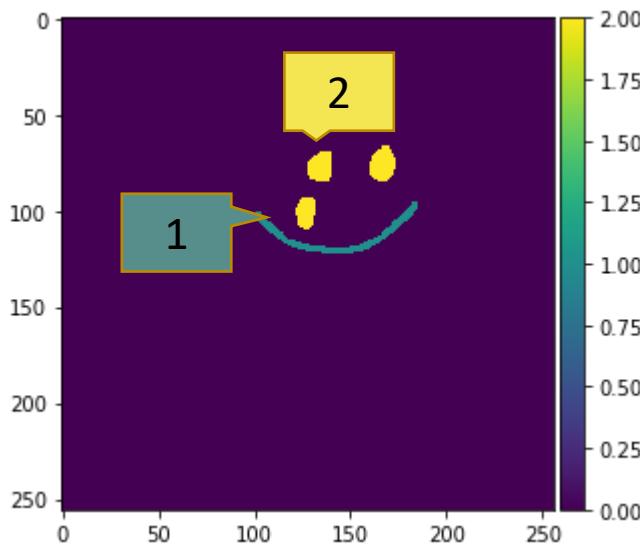
- Prepare an empty layer for annotations and keep a reference

```
labels = viewer.add_labels(  
    np.zeros(image.shape).astype(int))
```

- Read annotations

```
manual_annotations = labels.data
```

```
from skimage.io import imshow  
  
imshow(manual_annotations,  
       vmin=0, vmax=2)
```



- Pixel classification using scikit-learn
 - Expects one-dimensional arrays for
 - every feature individually
 - ground truth

```
# train classifier
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X, y)
```

Image data

Ground truth /
annotation

Image data

y_ = classifier.predict(X)

prediction

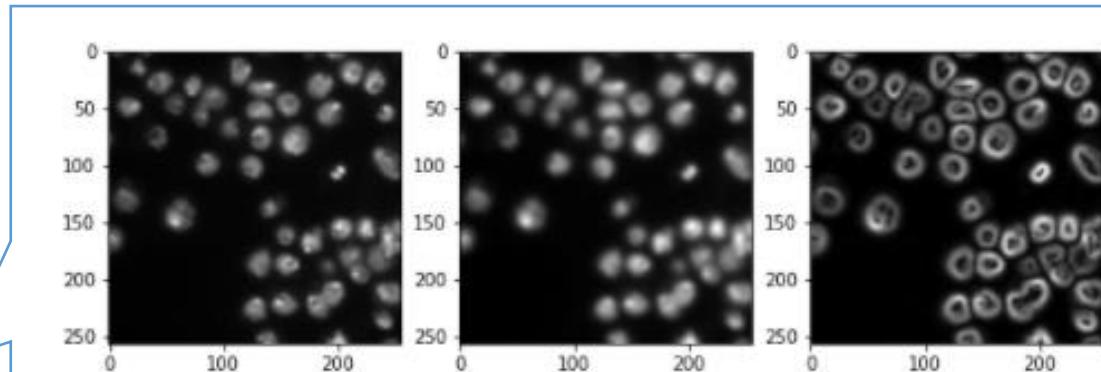
Interactive pixel classification

- Pixel classification using scikit-learn

- Expects one-dimensional arrays for
 - every feature individually
 - ground truth

```
# for training, we need to generate features
feature_stack = generate_feature_stack(image)
X, y = format_data(feature_stack, manual_annotations)

# train classifier
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X, y)
```

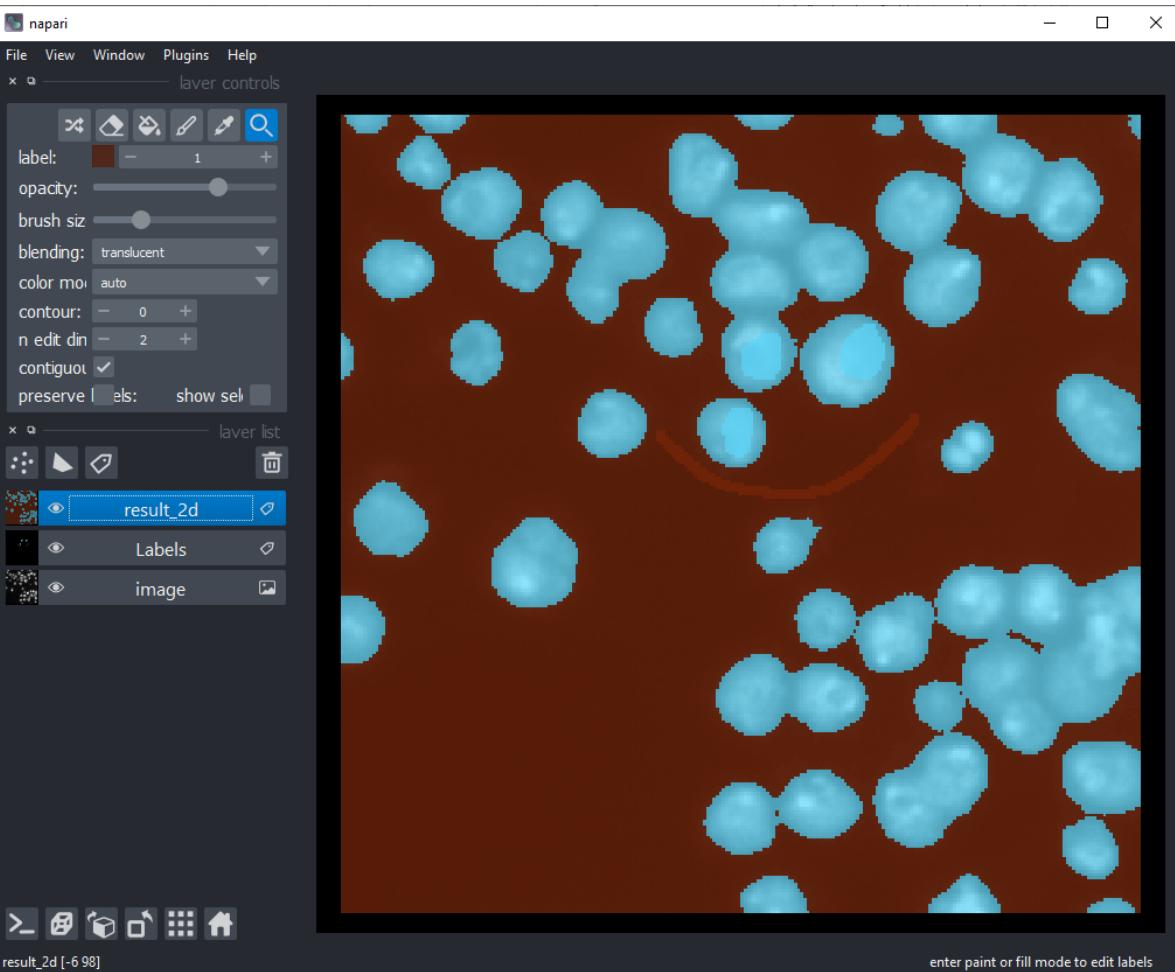


Interactive pixel classification

- Pixel classification using scikit-learn

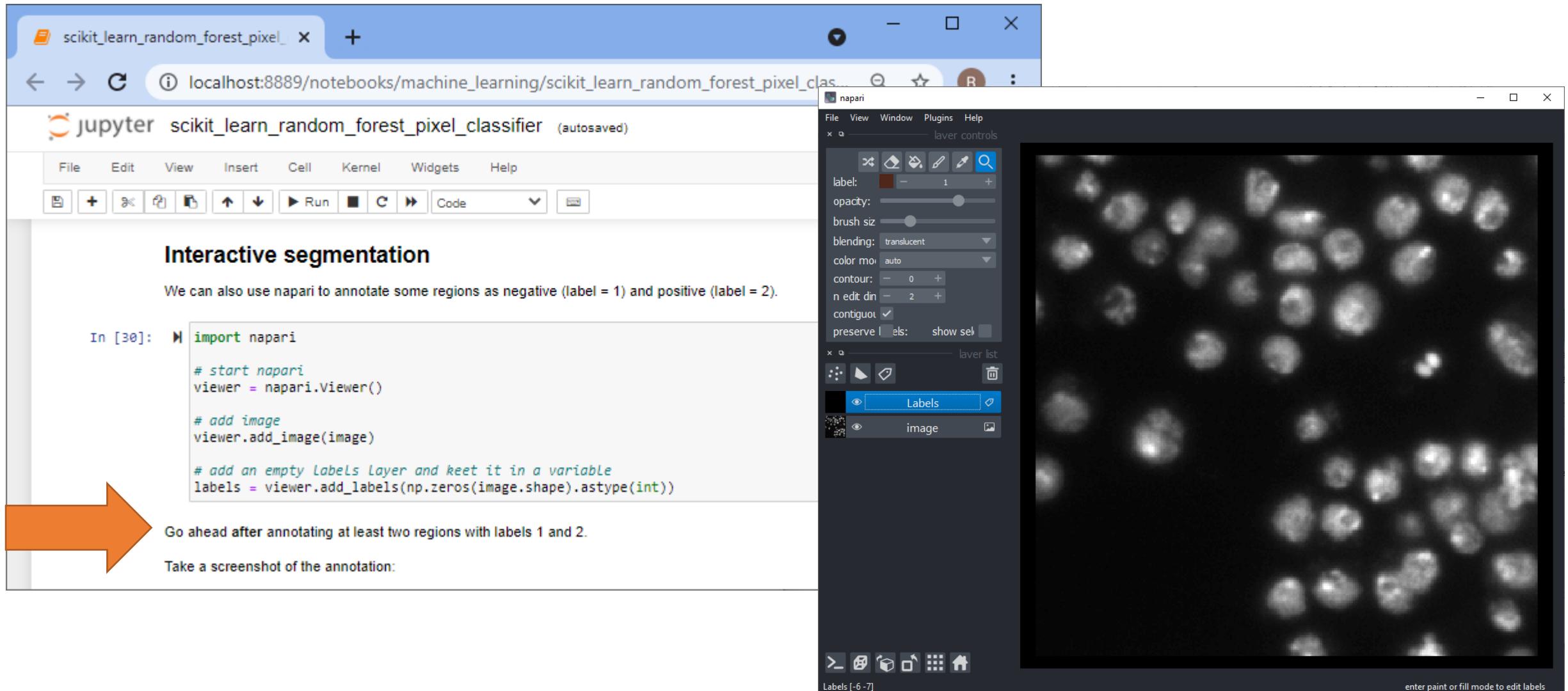
```
# process the whole image and show result
result_1d = classifier.predict(feature_stack.T)
result_2d = result_1d.reshape(image.shape)

viewer.add_labels(result_2d)
```



Interactive pixel classification

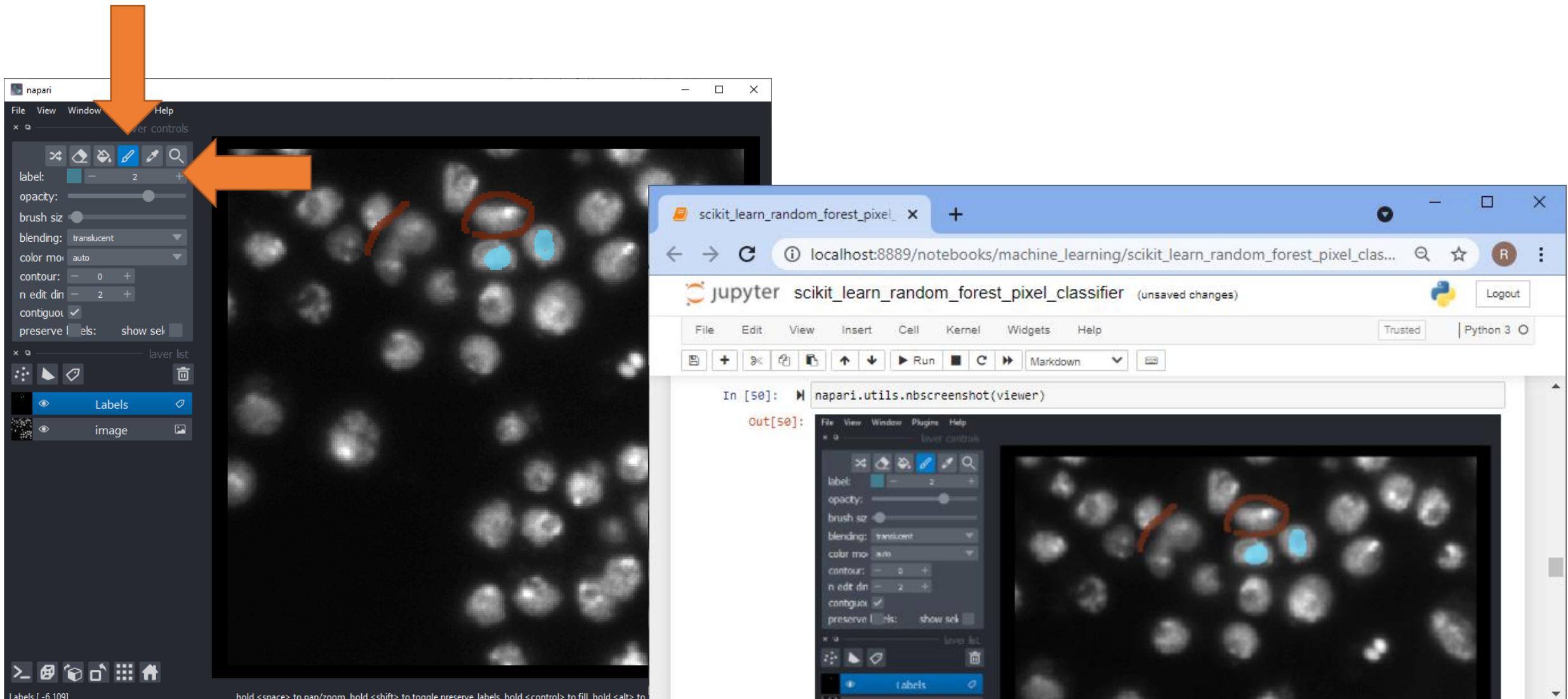
- Jupyter notebooks and napari side-by-side



The image shows a Jupyter notebook interface and the napari image viewer running side-by-side. On the left, the Jupyter notebook has a tab titled "scikit_learn_random_forest_pixel_" and displays Python code for importing napari and creating a viewer. An orange arrow points from the text "Go ahead after annotating at least two regions with labels 1 and 2." to the napari interface. On the right, the napari window shows a grayscale image of cells with a labels layer overlaid. A legend on the left of the napari interface shows a brown square labeled "1" and a white square labeled "2". The napari interface includes various segmentation tools like brush size, opacity, and blending mode.

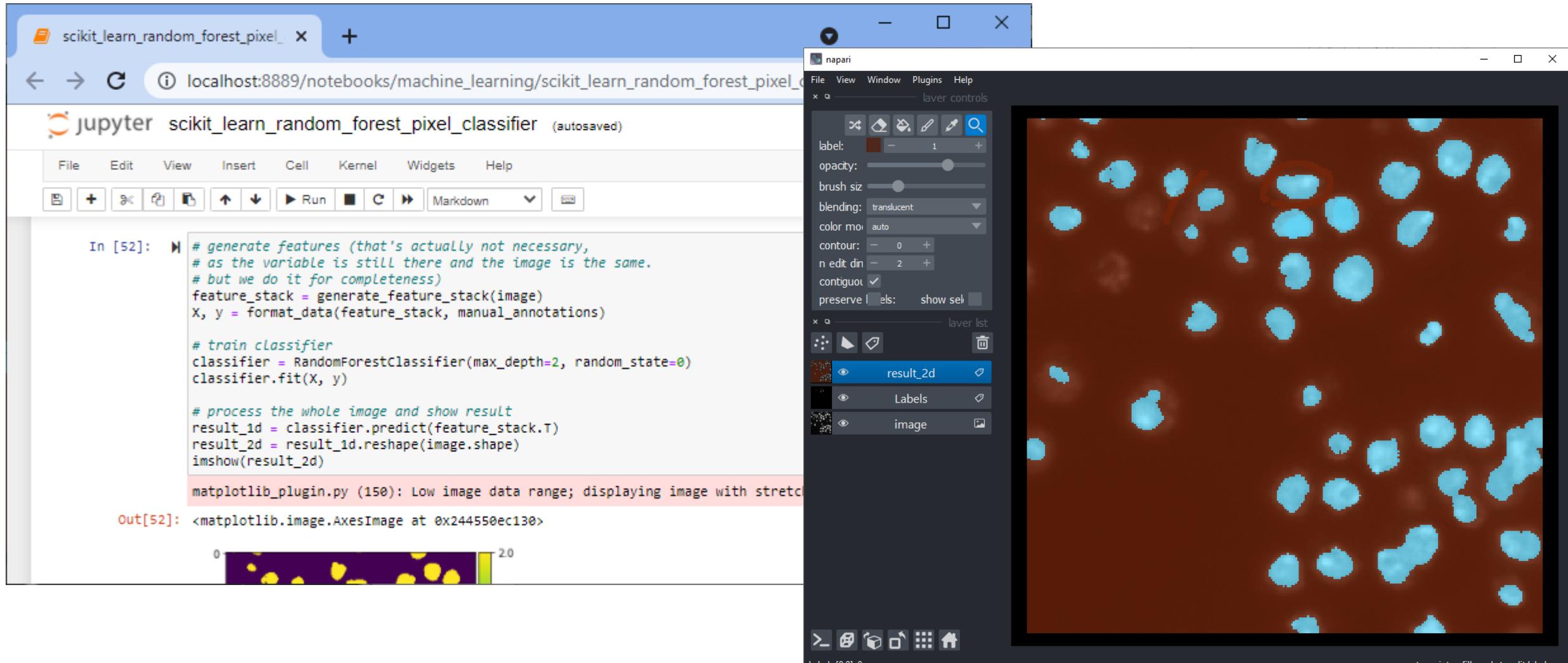
Interactive pixel classification

- Jupyter notebooks and napari side-by-side



Interactive pixel classification

- Jupyter notebooks and napari side-by-side



The figure displays a dual-interface setup for bio-image analysis. On the left, a Jupyter notebook cell contains the following Python code:

```
# generate features (that's actually not necessary,
# as the variable is still there and the image is the same.
# but we do it for completeness)
feature_stack = generate_feature_stack(image)
X, y = format_data(feature_stack, manual_annotations)

# train classifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X, y)

# process the whole image and show result
result_1d = classifier.predict(feature_stack.T)
result_2d = result_1d.reshape(image.shape)
imshow(result_2d)
```

A warning message from `matplotlib_plugin.py` is visible at the bottom of the cell: `(150): Low image data range; displaying image with stretch`.

The output of the cell is:

```
<matplotlib.image.AxesImage at 0x244550ec130>
```

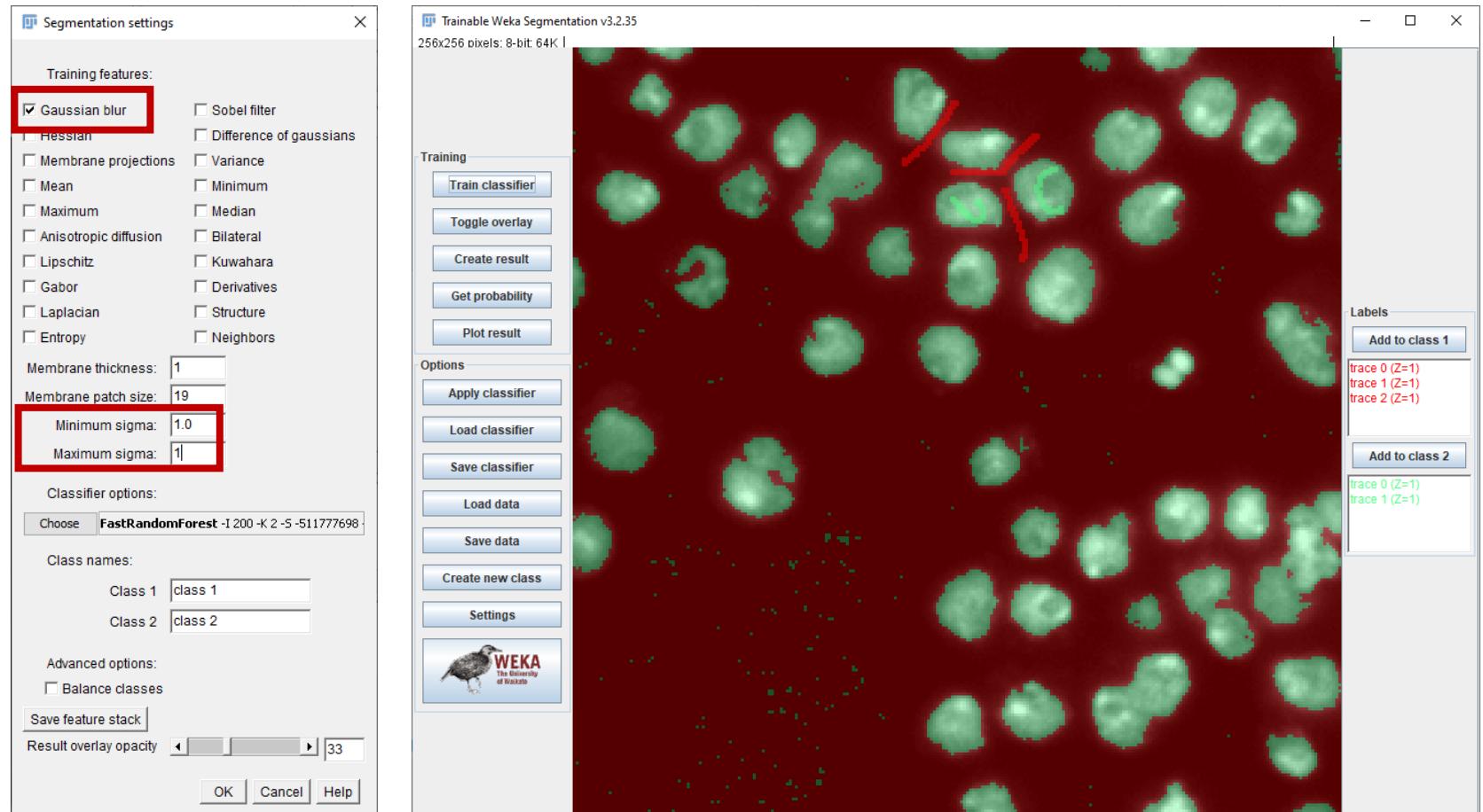
Below the code cell is a small color bar with a scale from 0 to 2.0.

On the right, the napari interface shows a 2D image of a brown background with numerous blue, irregularly shaped objects. A red brush tool is used to draw a selection around one of these blue objects. The napari control panel on the left includes sliders for `label`, `opacity`, `brush size`, and `blending`. The `layer controls` section lists three layers: `result_2d`, `Labels`, and `image`. The status bar at the bottom of the napari window shows `Labels [0 0]: 0` and `enter paint or fill mode to edit labels`.

Exercises

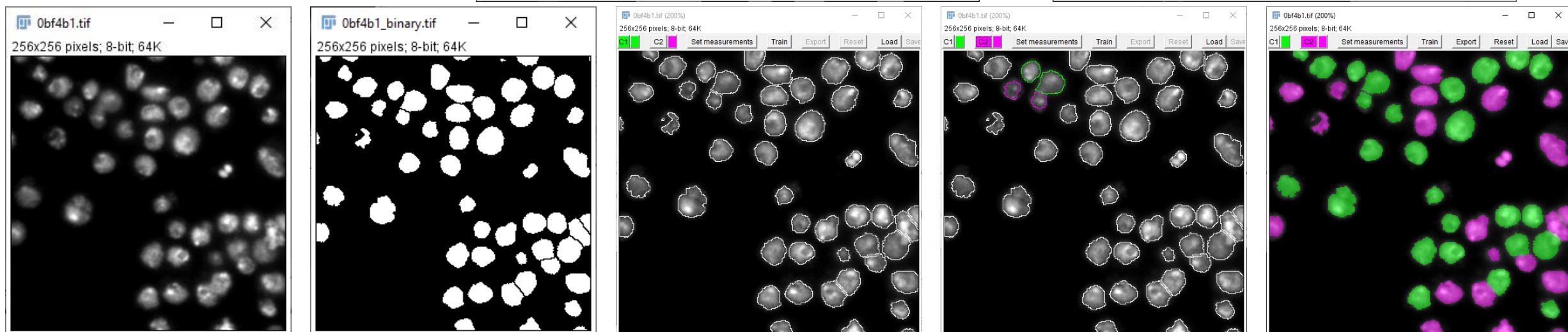
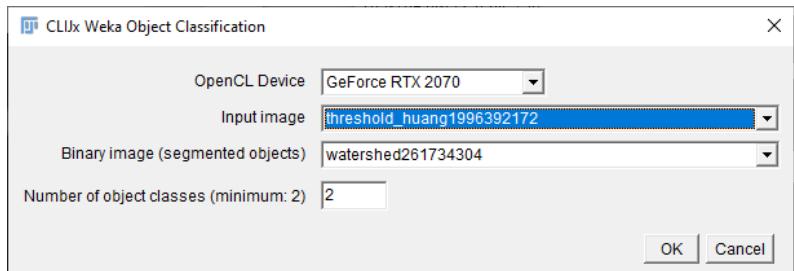
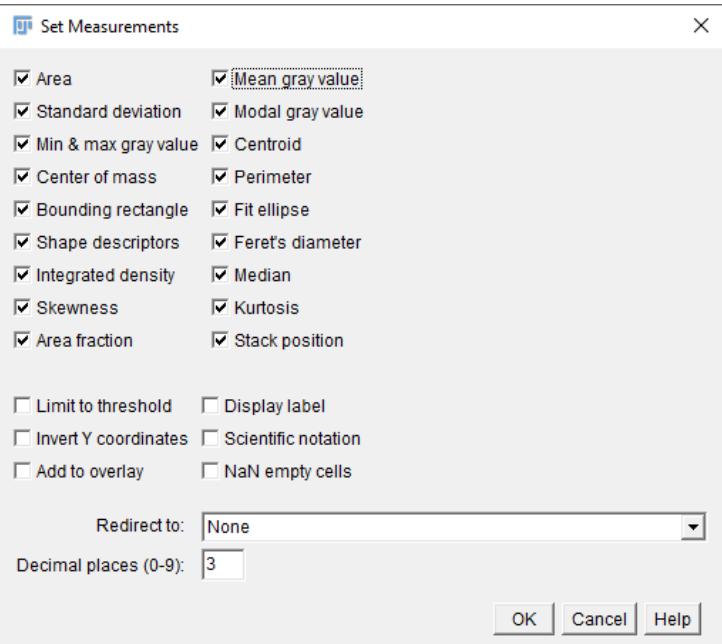
Exercises: Pixel classification using Trainable Weka / Labkit

- Use Trainable Weka Segmentation and / or Labkit
- Download an example image from the Broad BioImage Benchmark collection:
 - <https://bbbc.broadinstitute.org/BBBC038>
 - Use image 0bf4b144167694b6846d584cf52c458f34f28fcae75328a2a096c8214e01c0d0
- Train a random forest classifier using Weka Trainable Segmentation or LabKit in Fiji
- Can a classifier with just a single Gaussian blur be trained successfully?
- If yes, what does that imply?
- Is machine learning necessary to solve this segmentation problem?



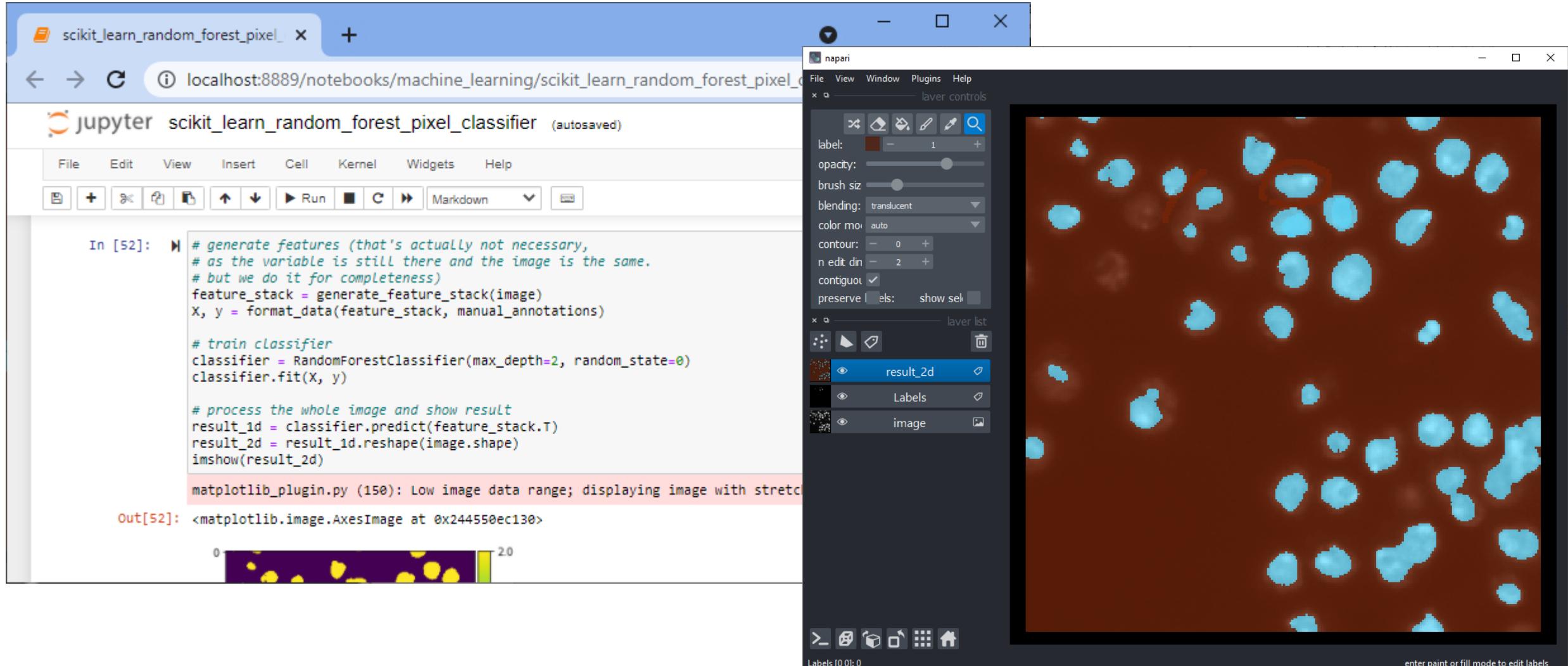
Exercise: Object classification

- Test CLIJx Weka Object classification to separate objects according to
 - Position
 - Shape
 - Size
- Which measurements do you have to set?
- Optional: Can you classify three classes?
 - Small, large, non-round



Exercise: scikit-learn / napari

- In the jupyter notebook: Which code changes are necessary to classify three classes of pixels?



The image shows a Jupyter Notebook interface and a napari application window side-by-side.

Jupyter Notebook:

- Title bar: scikit_learn_random_forest_pixel_
- Toolbar: File, Edit, View, Insert, Cell, Kernel, Widgets, Help.
- Code cell (In [52]):

```
# generate features (that's actually not necessary,
# as the variable is still there and the image is the same)
# but we do it for completeness)
feature_stack = generate_feature_stack(image)
X, y = format_data(feature_stack, manual_annotations)

# train classifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X, y)

# process the whole image and show result
result_1d = classifier.predict(feature_stack.T)
result_2d = result_1d.reshape(image.shape)
imshow(result_2d)
```
- Output cell (Out[52]):

```
matplotlib_plugin.py (150): Low image data range; displaying image with stretch
```

napari Application:

- Title bar: napari
- Toolbar: File, View, Window, Plugins, Help.
- Panel: layer controls (label, opacity, brush size, blending, color mode, contour, n edit dir, contiguous, preserve labels, show sel).
- Panel: layer list (result_2d, Labels, image).
- Image view: A 2D image of blue, irregularly shaped objects on a brown background. One object is highlighted with a red outline.
- Status bar: Labels [0 0]: 0, enter paint or fill mode to edit labels.

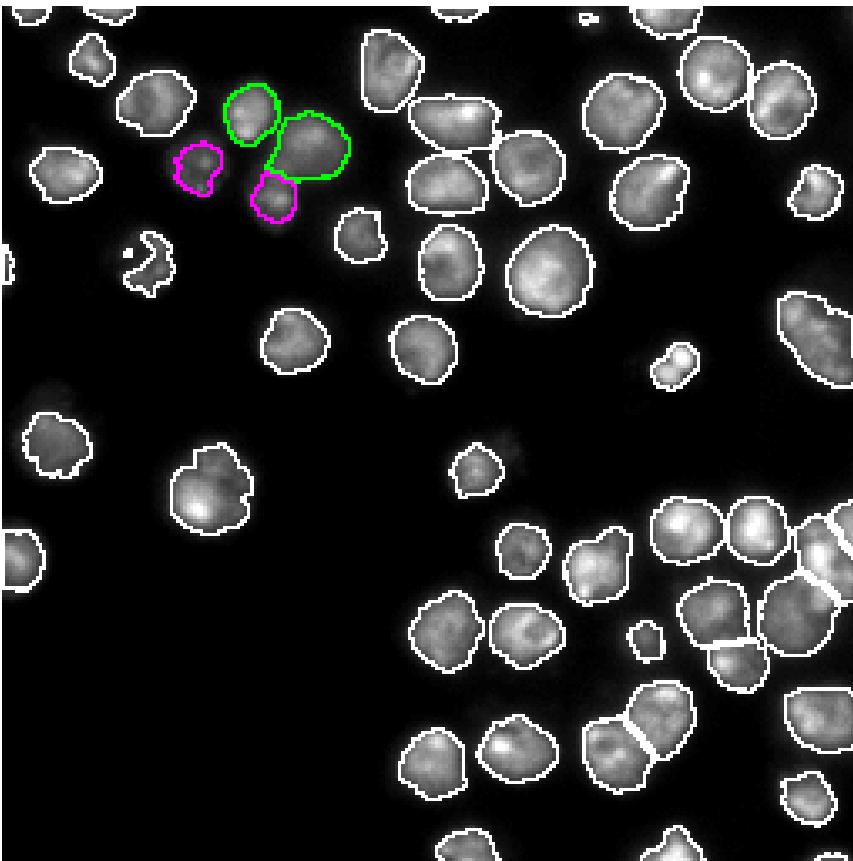
Exercise: Precision / recall



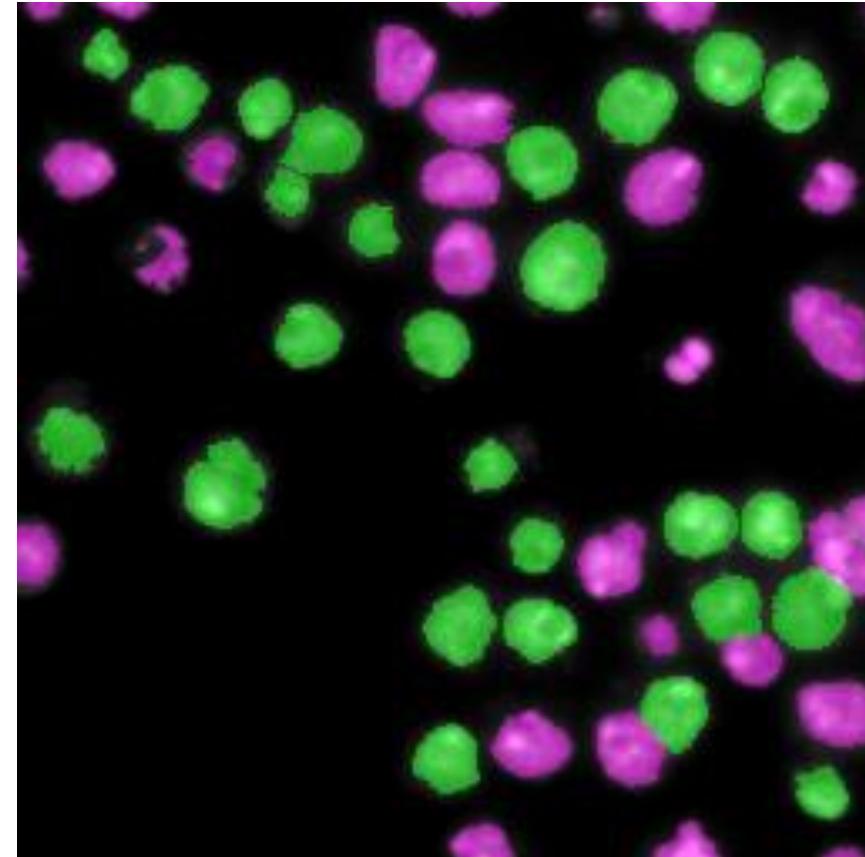
PoL
Physics of Life
TU Dresden



- Calculate precision and recall for the algorithm which created this prediction



Ground truth annotation



Prediction

Today, you learned

- Machine learning for Pixel and Object classification
- Segmentation / prediction quality measurements
 - Precision
 - Recall
 - Jaccard Index
- Fiji
 - Trainable Weka Segmentation
 - Labkit
- Python
 - Scikit-learn / napari

Coming up next:

- Deep learning
 - Convolutional neural networks

