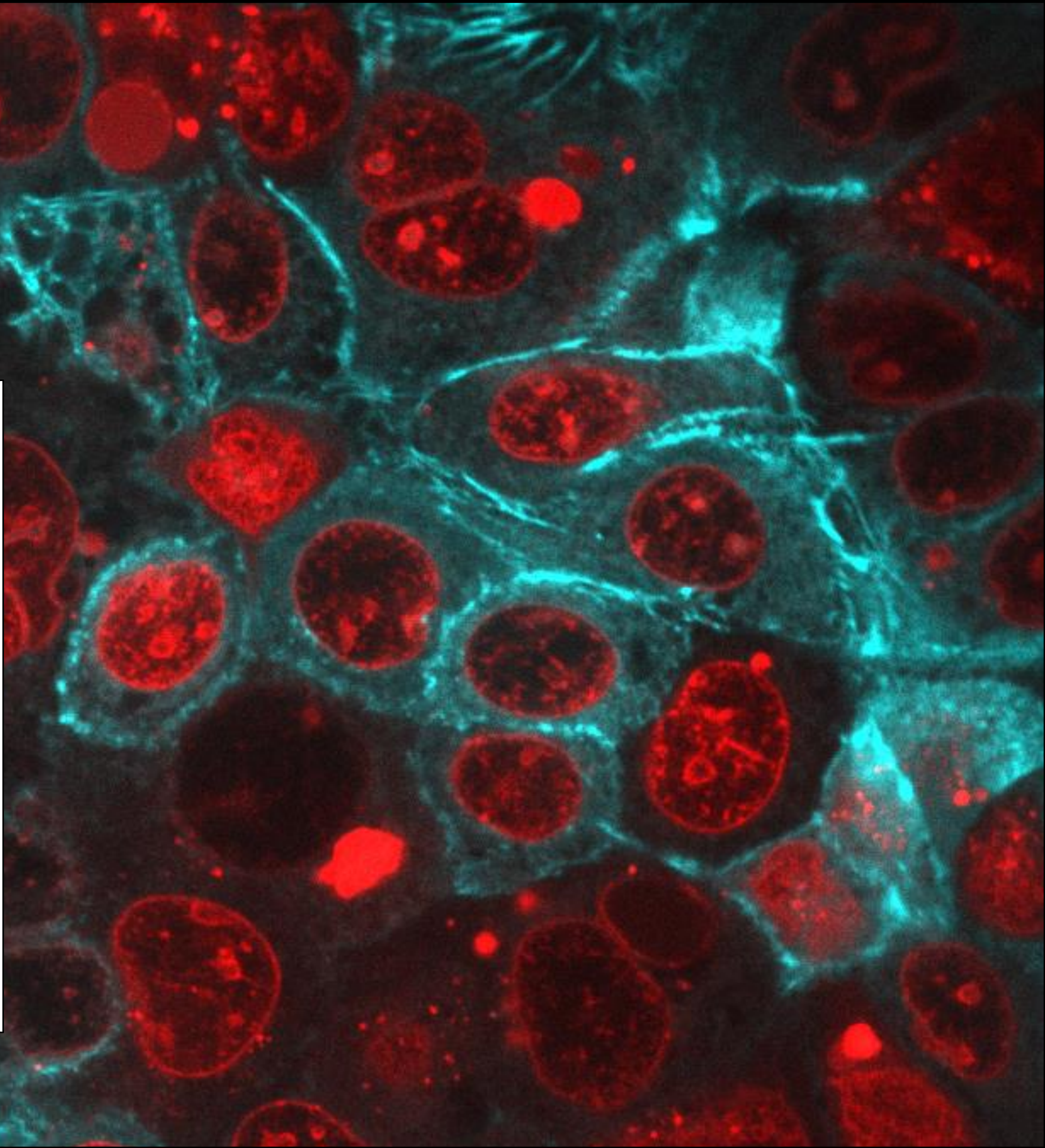
Damaged Nuclei Identification

António Maschio, Dimitrios Anastasiou,
Georgios Sevastakis, Liam de Búrca

UNIVERSITY OF COPENHAGEN

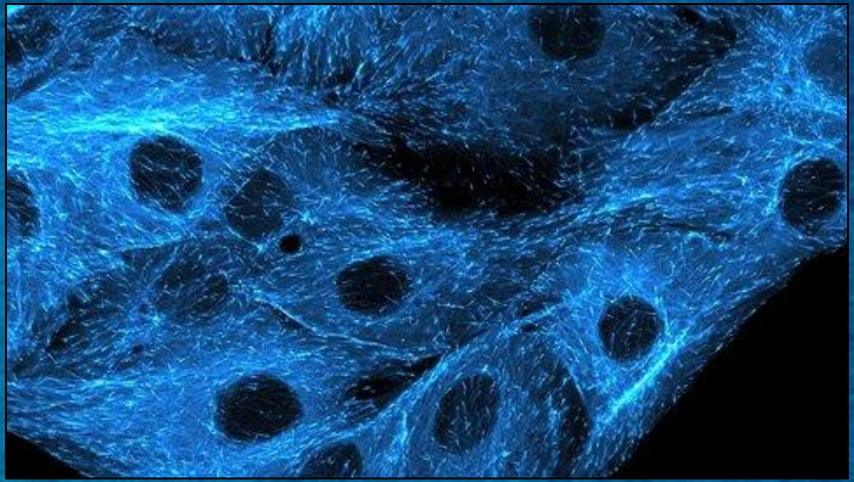


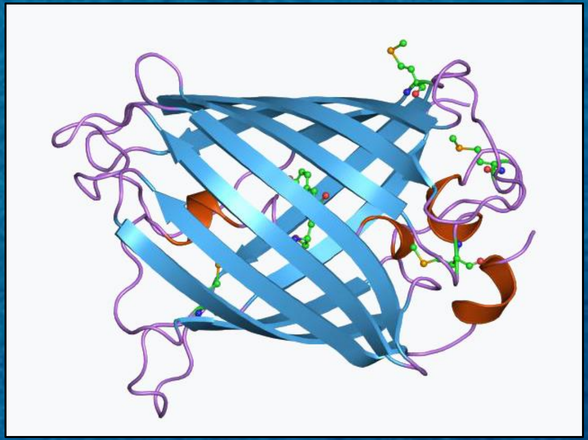
Overview

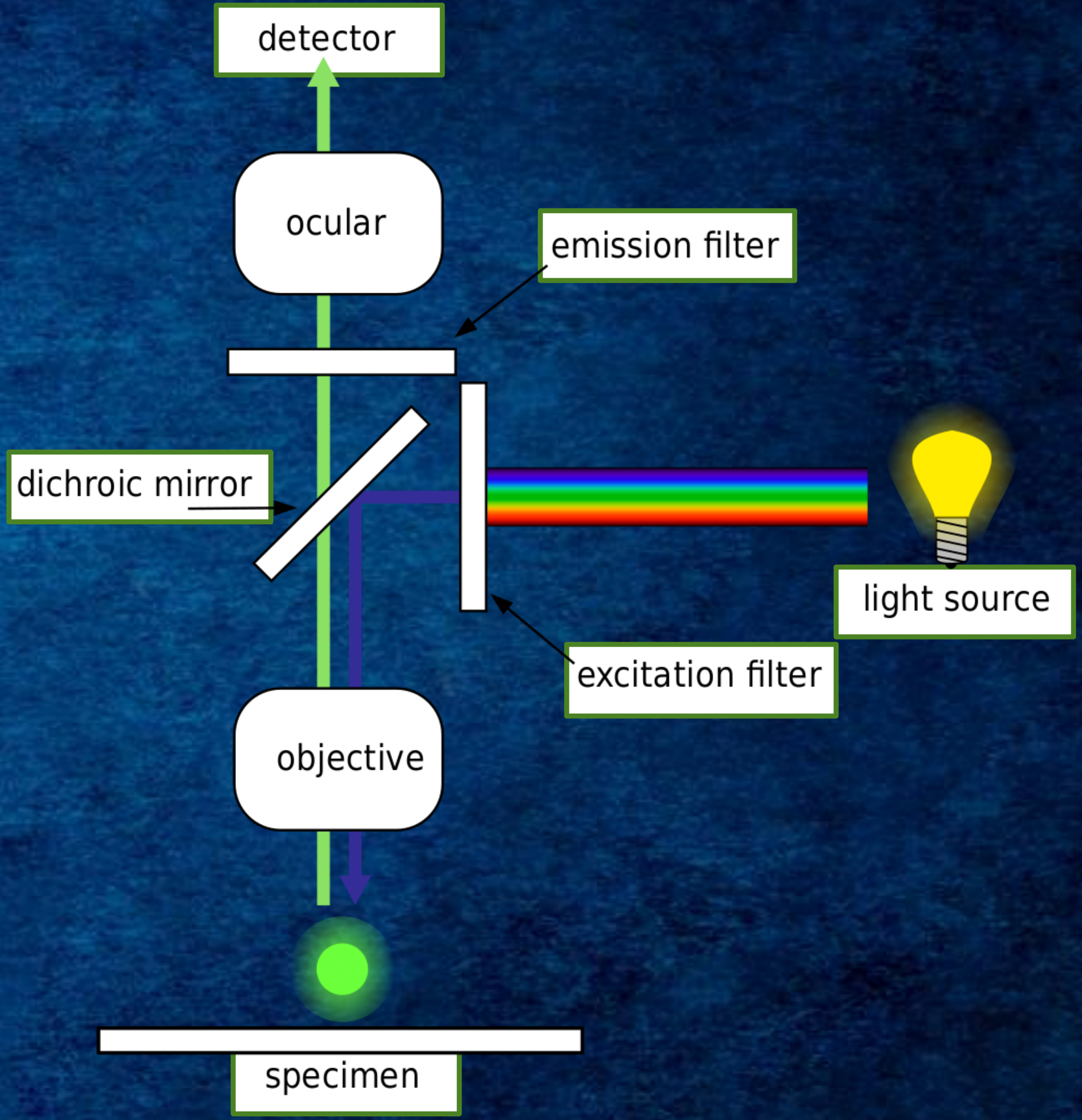
- Introduction
  - Goal & Description
- Data acquisition & Inspection
- Preprocessing
  - Segmentation
  - Image Processing
  - Unsupervised Approach
- Implementation
  - Random Forest & XGBoost
  - CNN
- Remarks
- Failed Attempts & Future Work

Introduction

Goal & Description

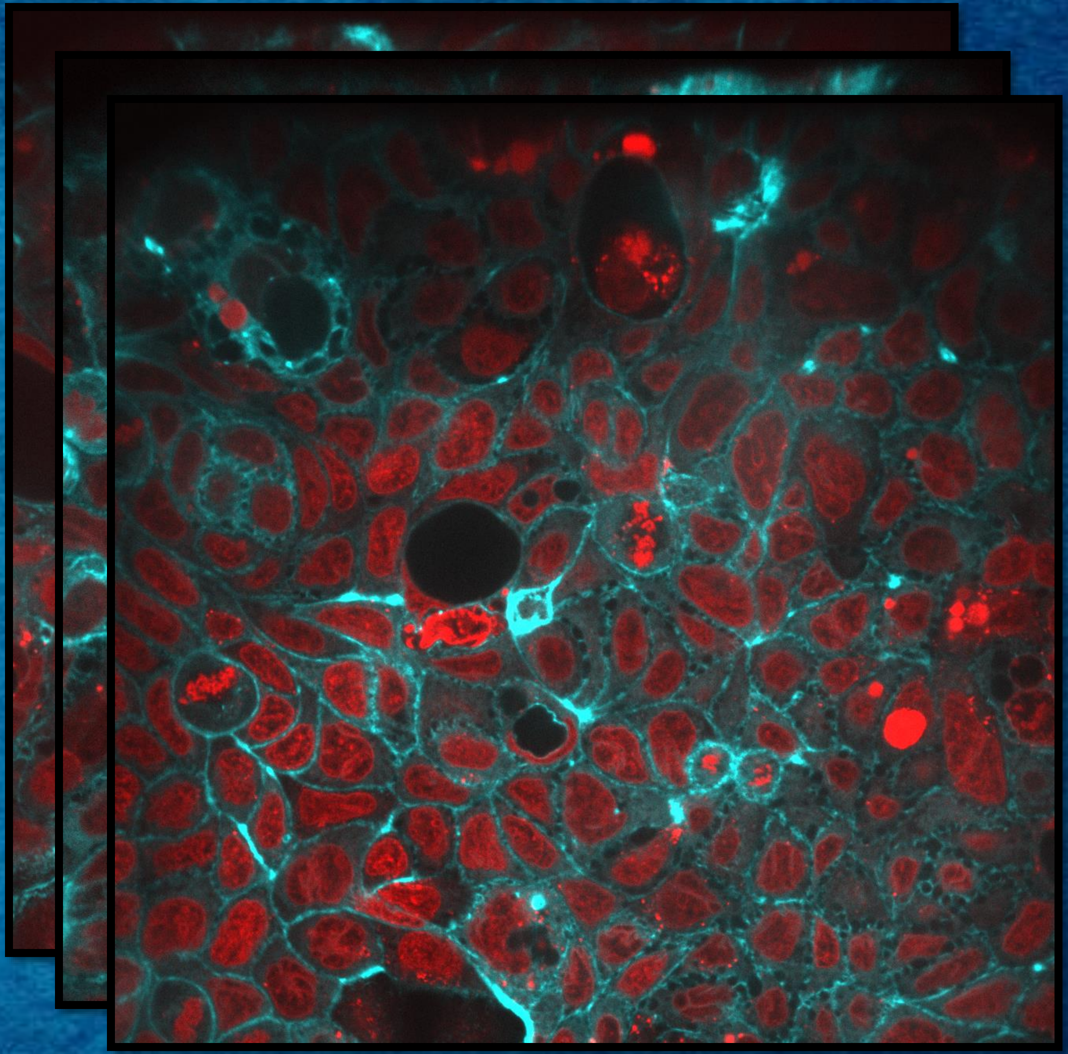




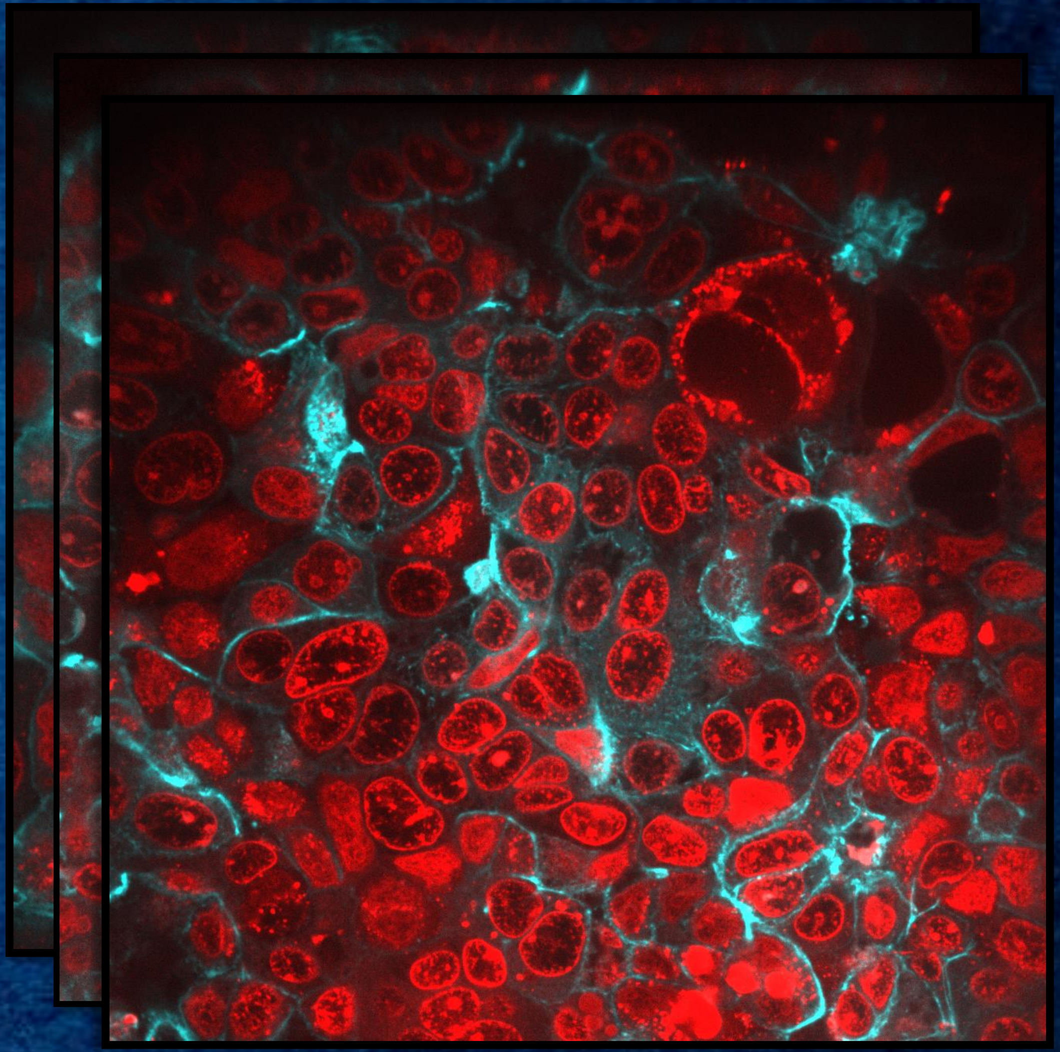


Data Acquisition & Inspection

Control



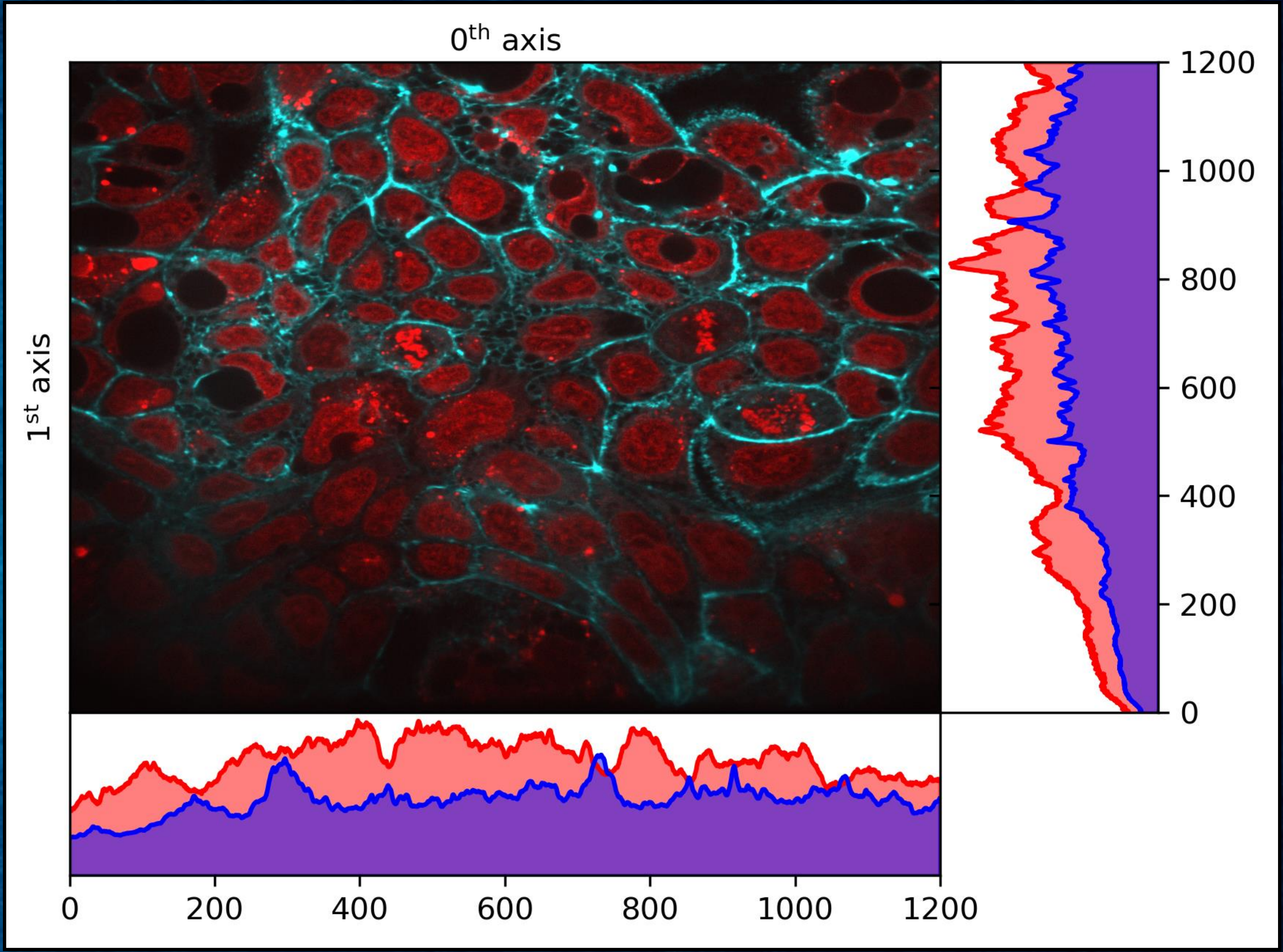
Drug



Preprocessing
Image Processing

Clear shadowing (lack of flux) in the lower half of the image.

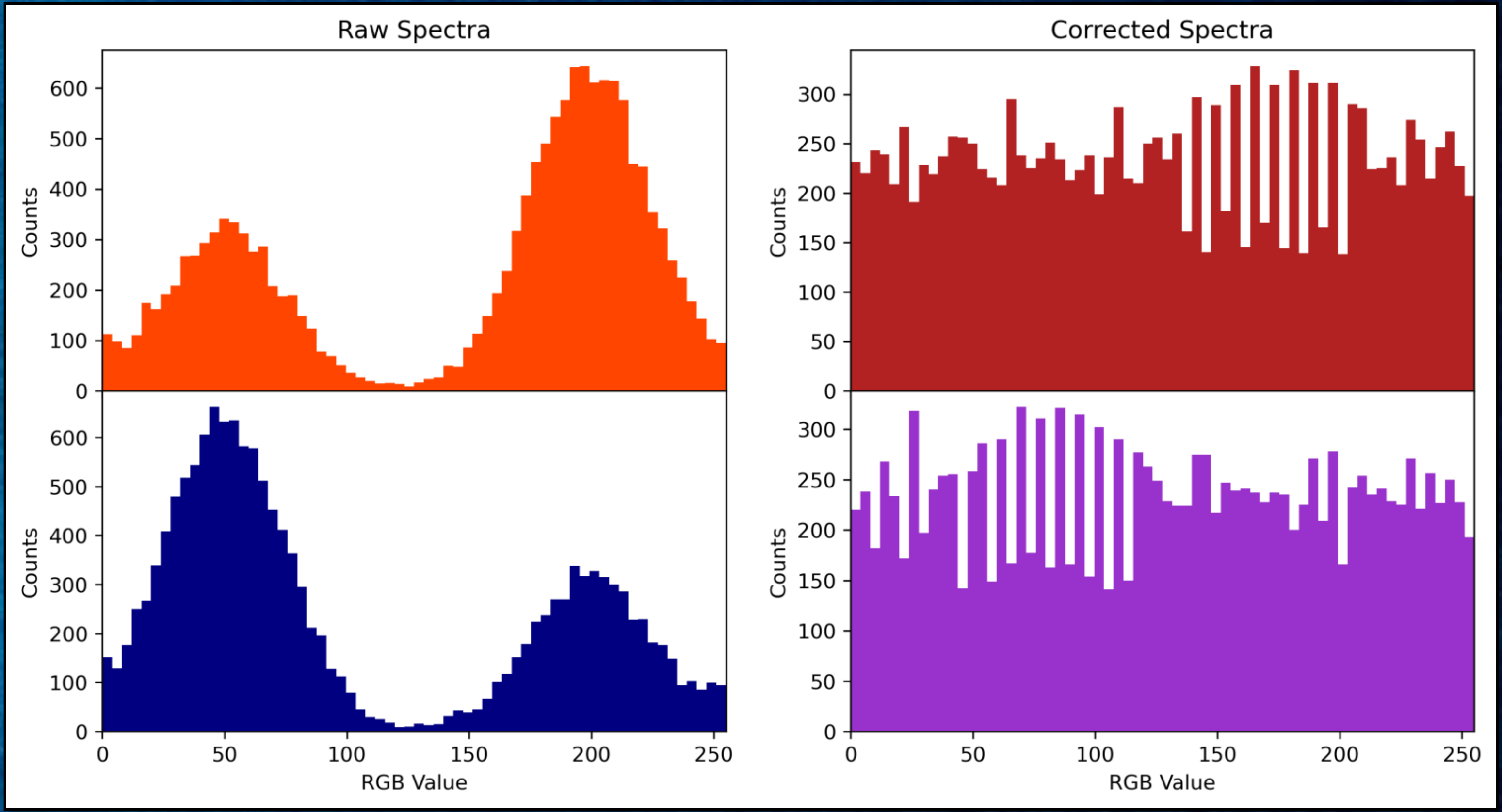
We could throw away part of the image, or use colour correction techniques...



Preprocessing
Image Processing

Histogram equalisation boosts areas with low flux, and softens areas with high.

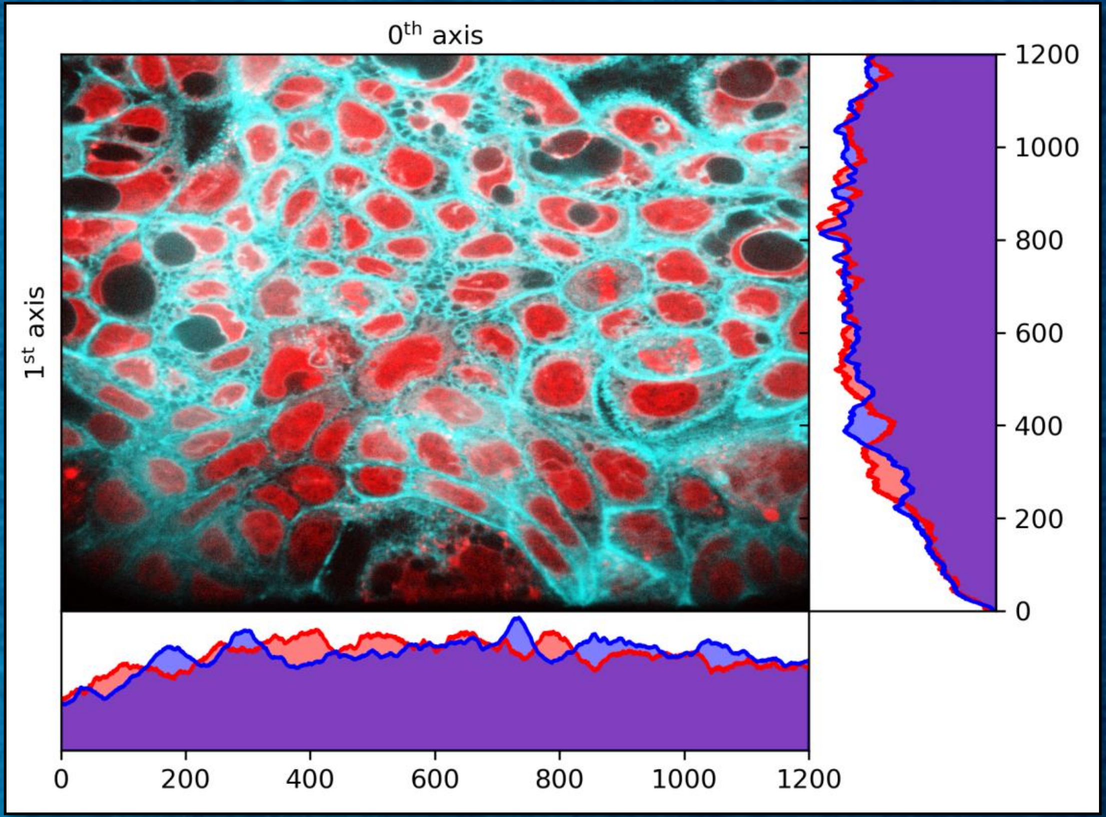
Resultant spectra are roughly uniform.



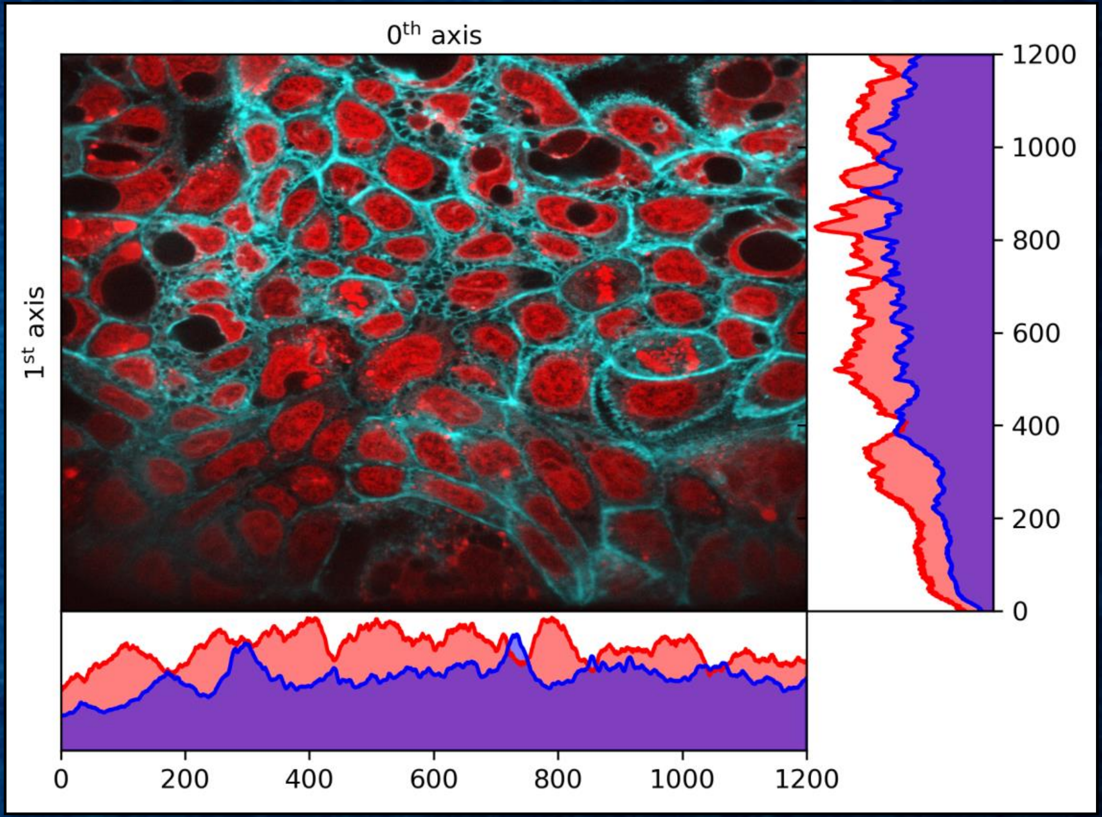
Preprocessing

Image Processing

Global (Basic)



Local (Adaptive)

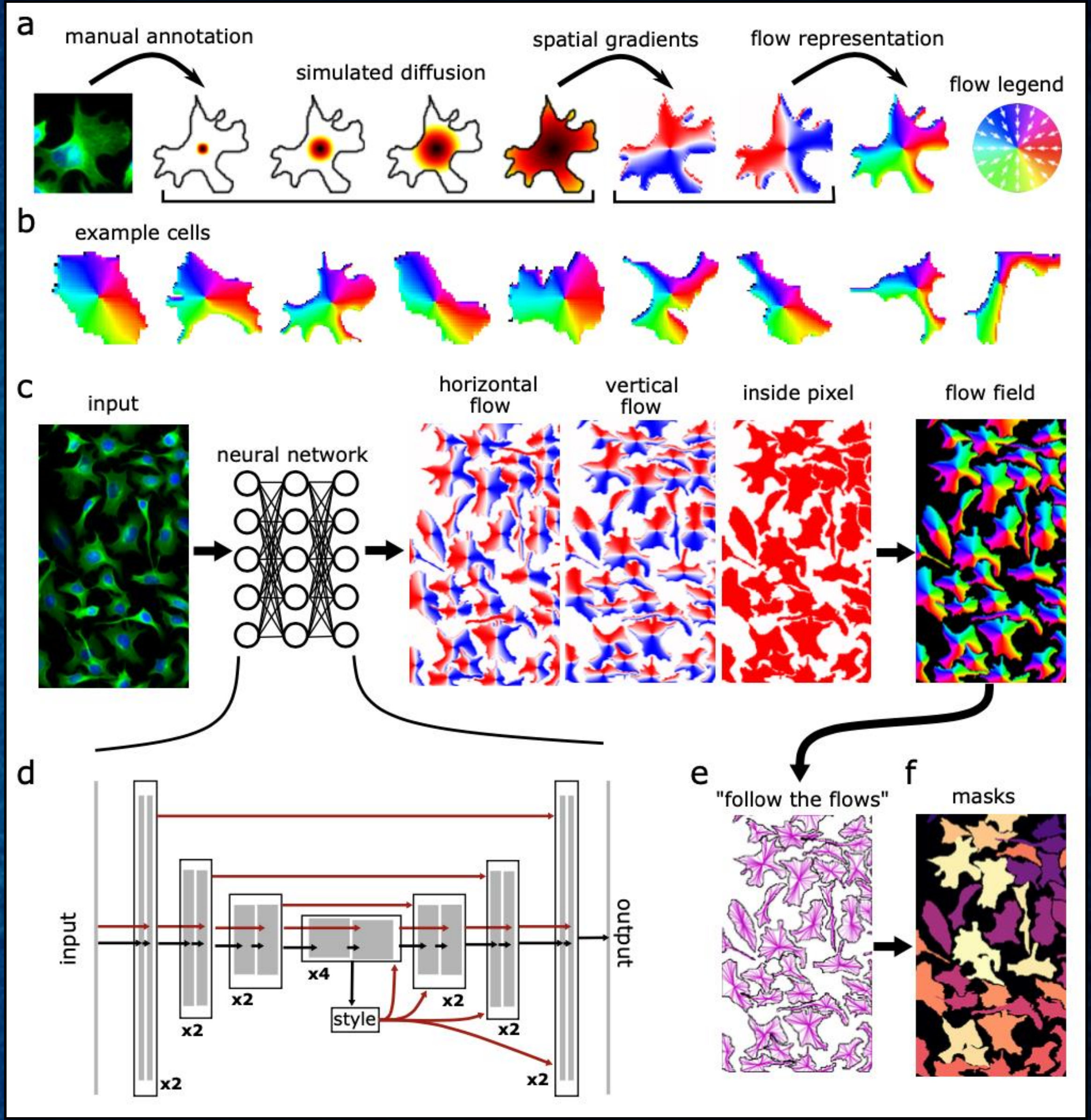


Preprocessing
Segmentation

To segment the nuclei (and cells), we used **Cellpose**.

Cellpose works by applying a CNN to predict "flows" (imagine flow of heat away from the nucleus centre), and then discerning which pixels have flow paths toward the centre.

Originally made for cell detection.



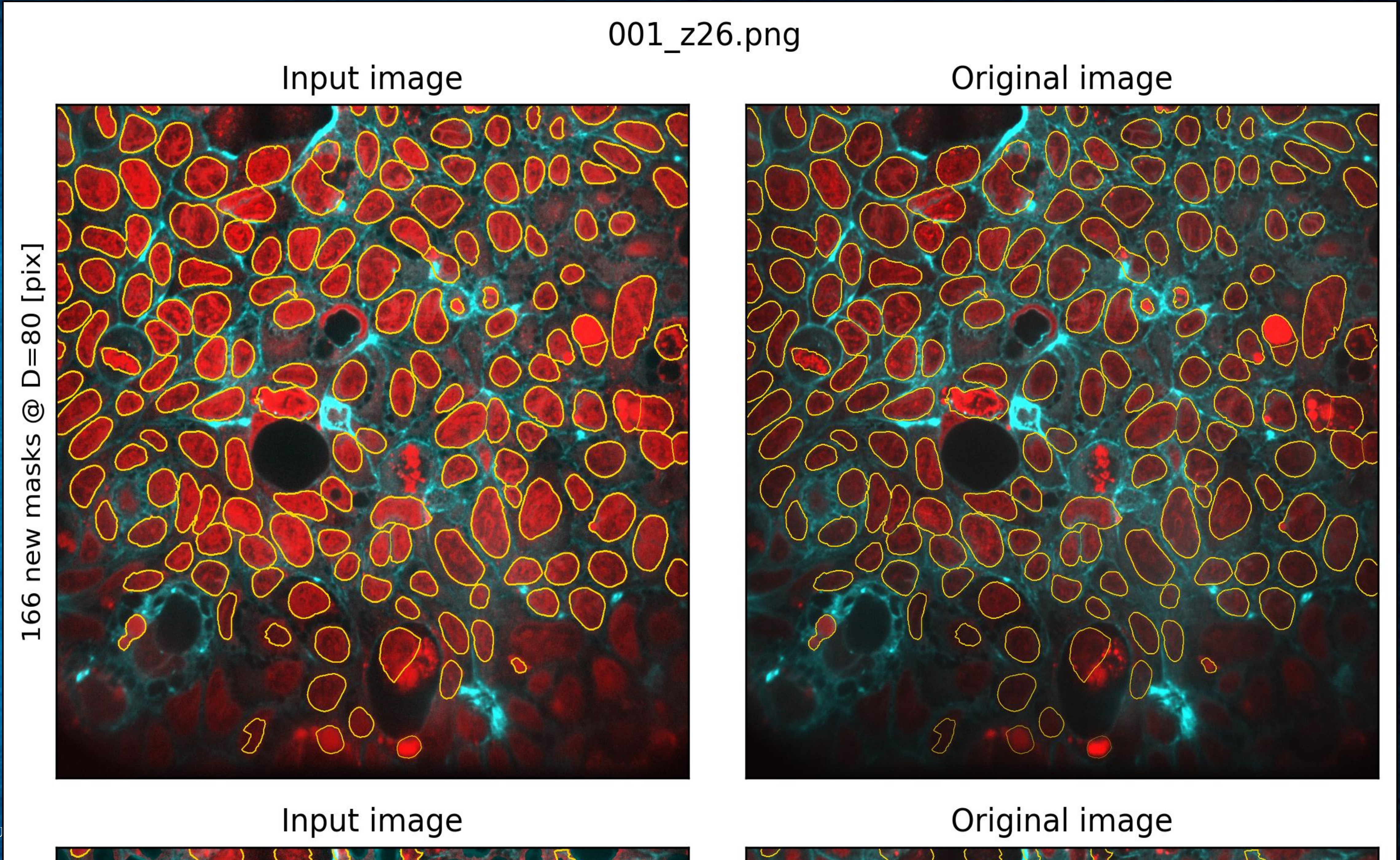
https://www.biorxiv.org/content/10.1101/2020.02.02.931238v1

Preprocessing
Segmentation

By repeatedly masking out previous segments and colour correcting, we can capture nuclei and cells in darker areas of images.

… but we also capture more trash!

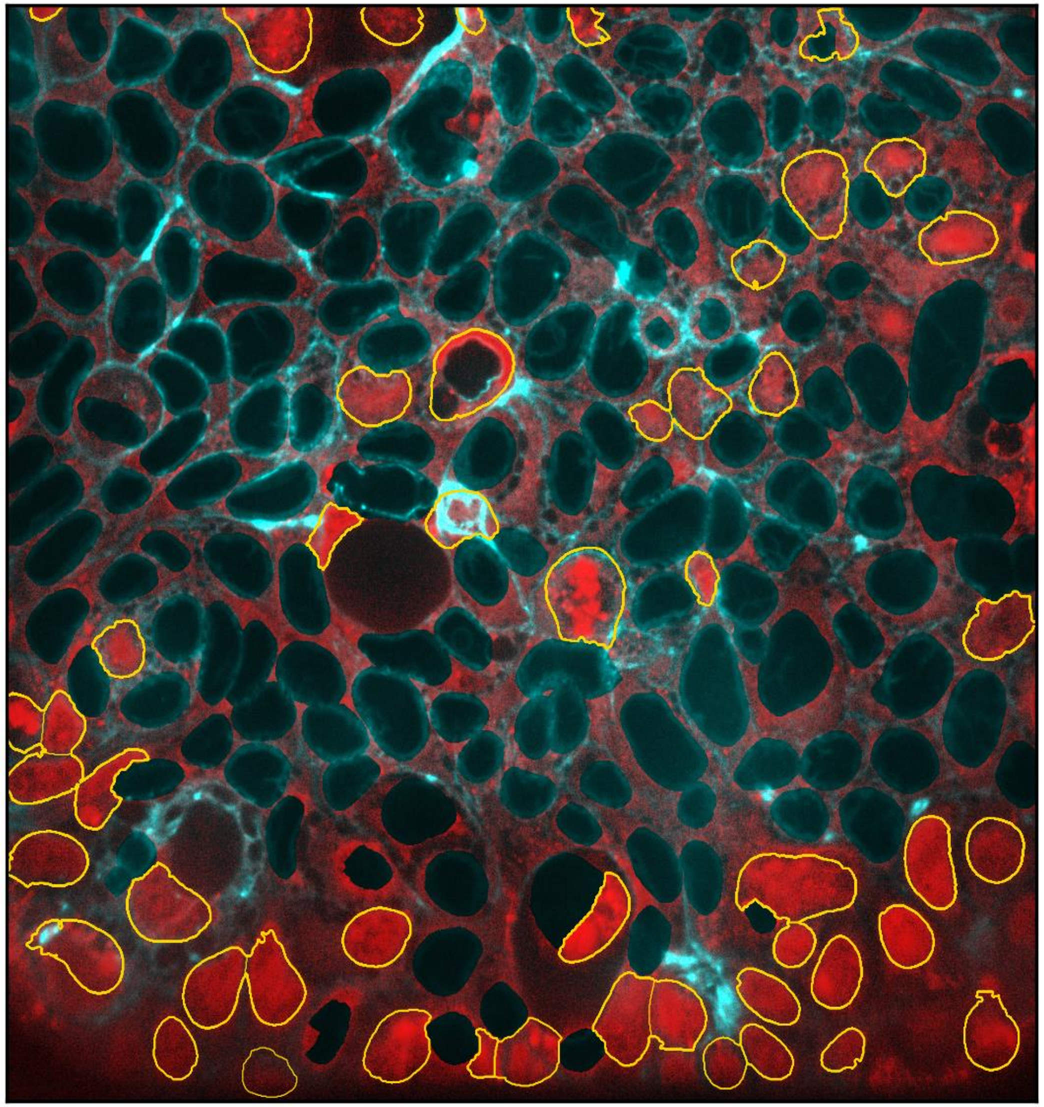
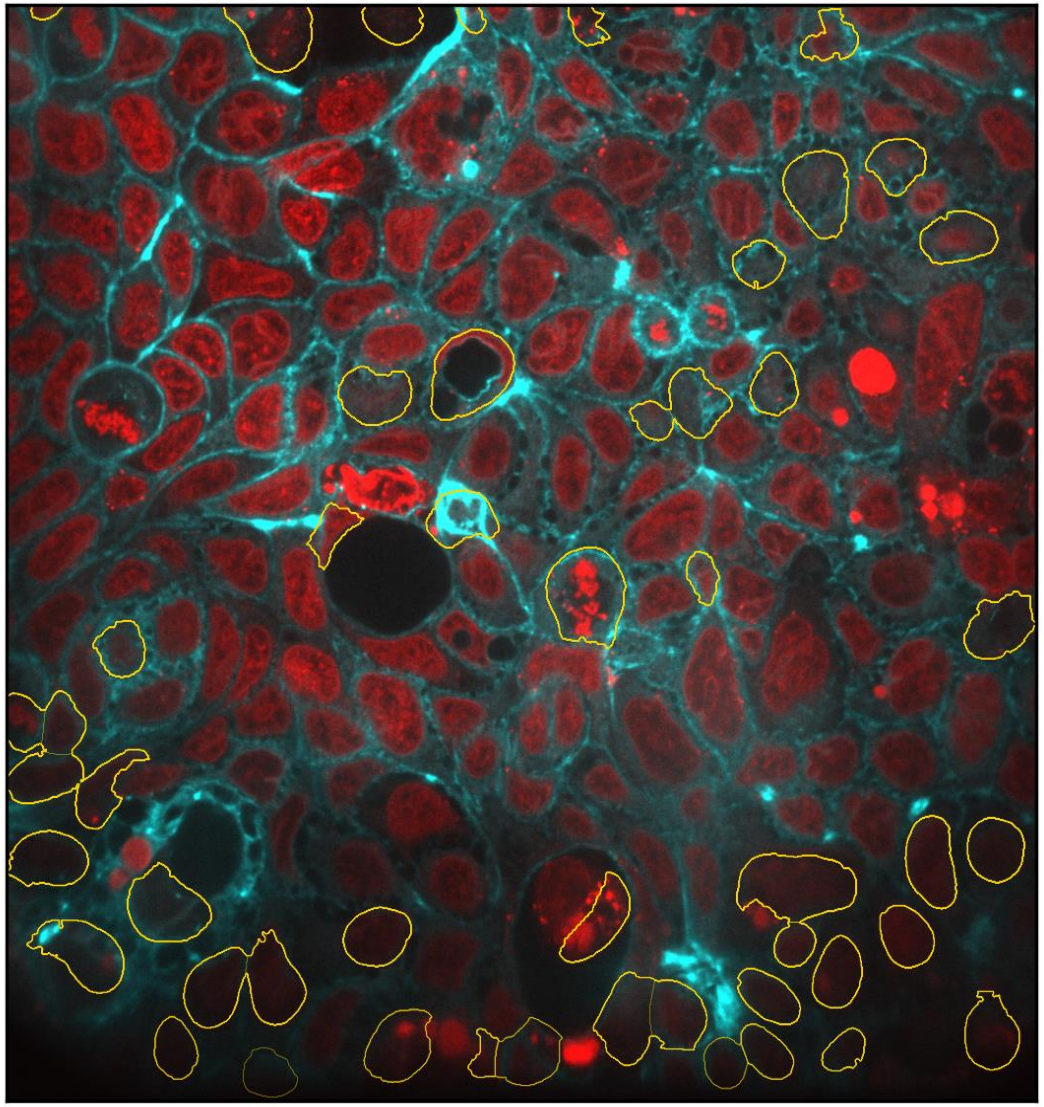
Solution: segment cells and nuclei at the same time, and filter pairs with the most overlap (agreement).


001_z26.png
Input image
Original image
166 new masks @ D=80 [pix]
Input image
Original image


Input image

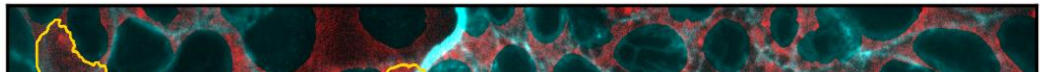
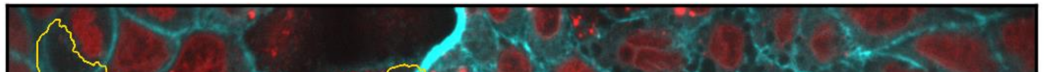
Original image

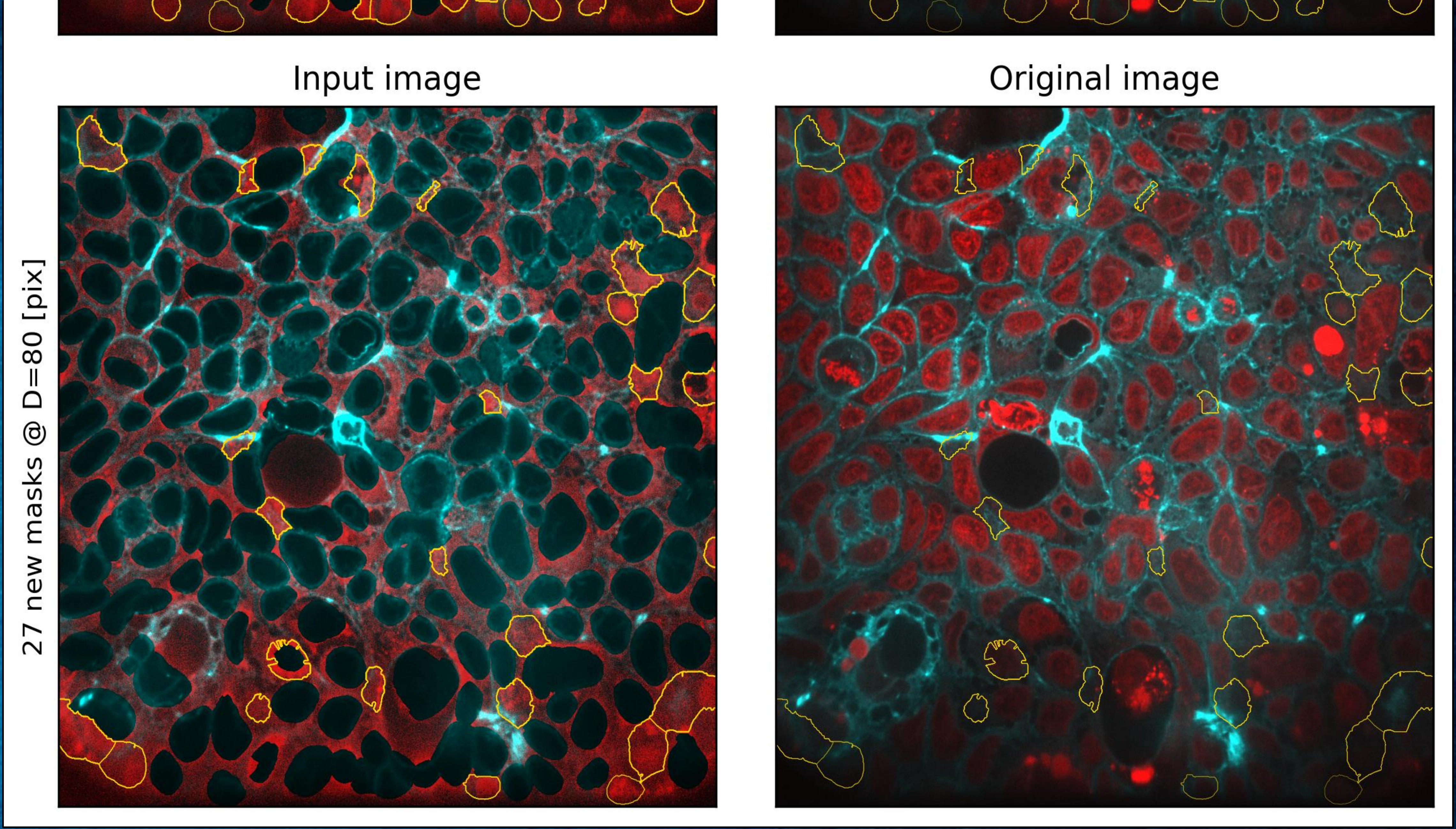
50 new masks @ D=80 [pix]

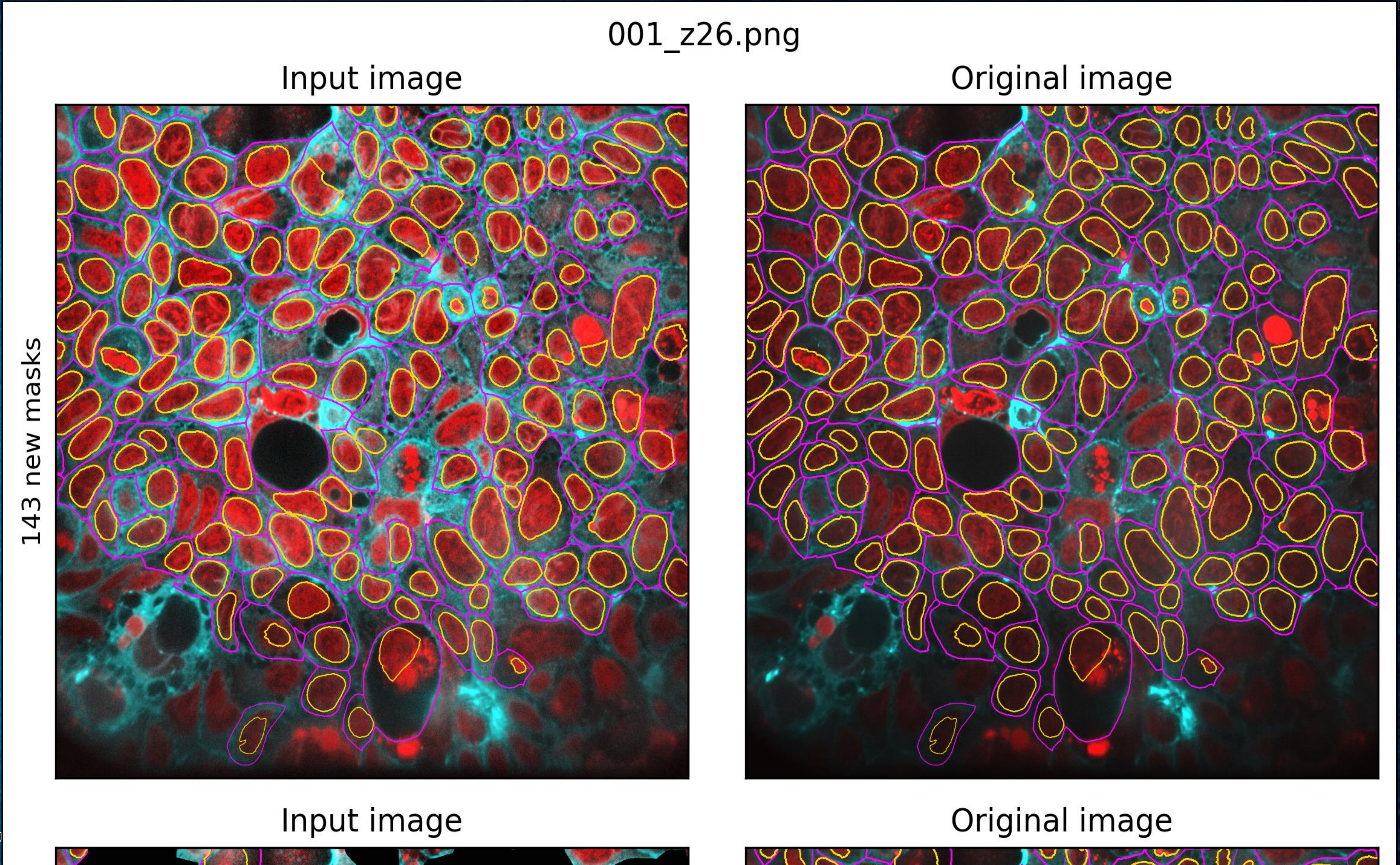



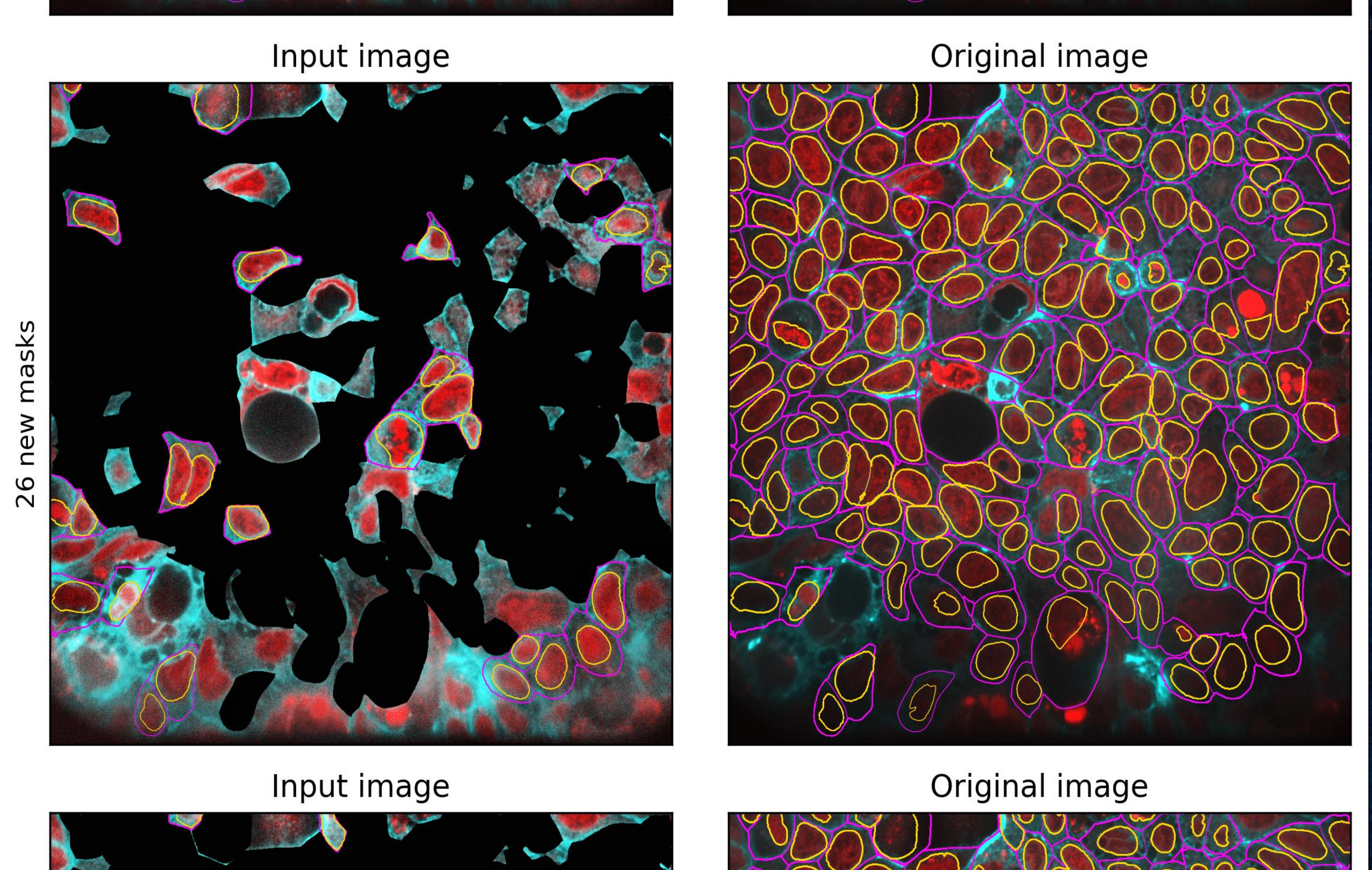
Input image

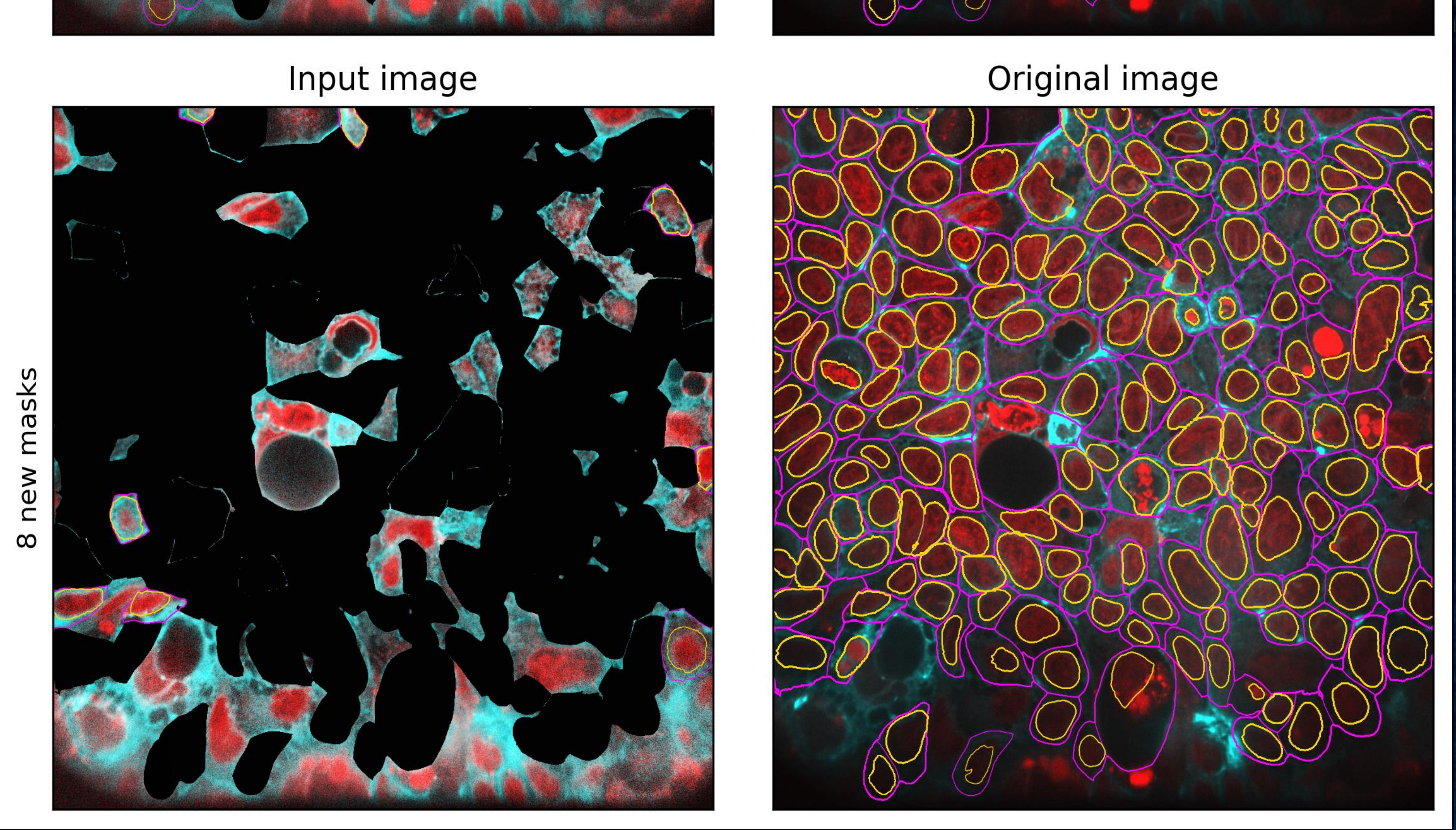
Original image


Input image
Original image
27 new masks @ D=80 [pix]


001_z26.png
Input image
Original image
143 new masks
Input image
Original image

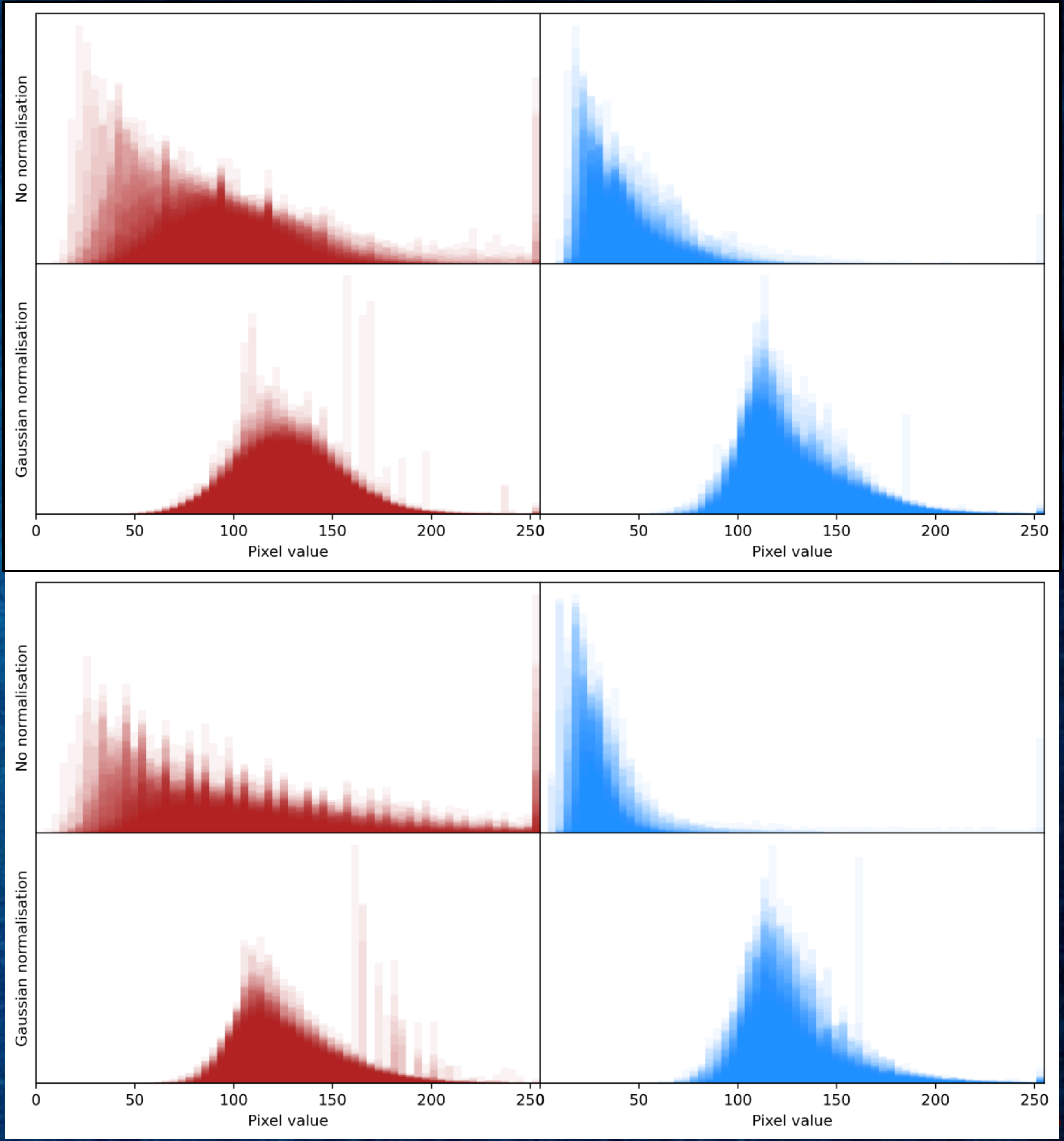

Input image
Original image
26 new masks
Input image
Original image


Input image
Original image
8 new masks

Preprocessing
Segmentation

We want to make nuclei in shaded areas comparable to nuclei in lighter areas.

By performing a Gaussian normalisation (RGB to Z), and mapping back to RGB we collect all spectra.



Preprocessing
Unsupervised Approach

Which features do we extract?

We extracted 76 features from each nucleus and cell using blue, red, and binary images, e.g.:

- Regionprops: eccentricity, solidity, diameter...
- Haralick features (texture).
- Custom: roundness, edge flux, Shannon entropy...

Many features were highly correlated.

https://www.researchgate.net/publication/302341151_Textural_Features_for_Image_Classification

Preprocessing
Unsupervised Approach

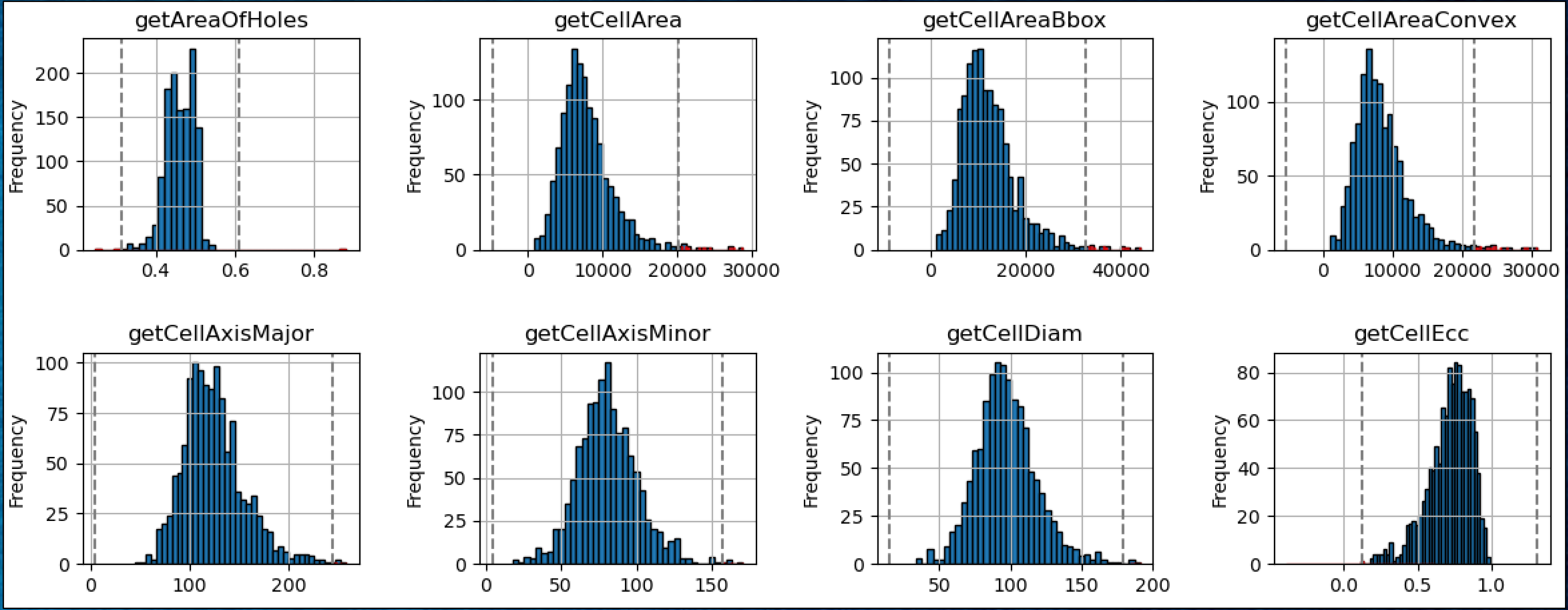
How can we detect faulty ROIs (segments) without having to look through 2000+ images?

We could apply dimensionality reduction and clustering algorithms, and hope that the data magically falls into large groups with a few segments placed randomly around them…

How do quantify what an outlier is?

- Thresholds from histograms
- Kernel density estimator
- DBSCAN

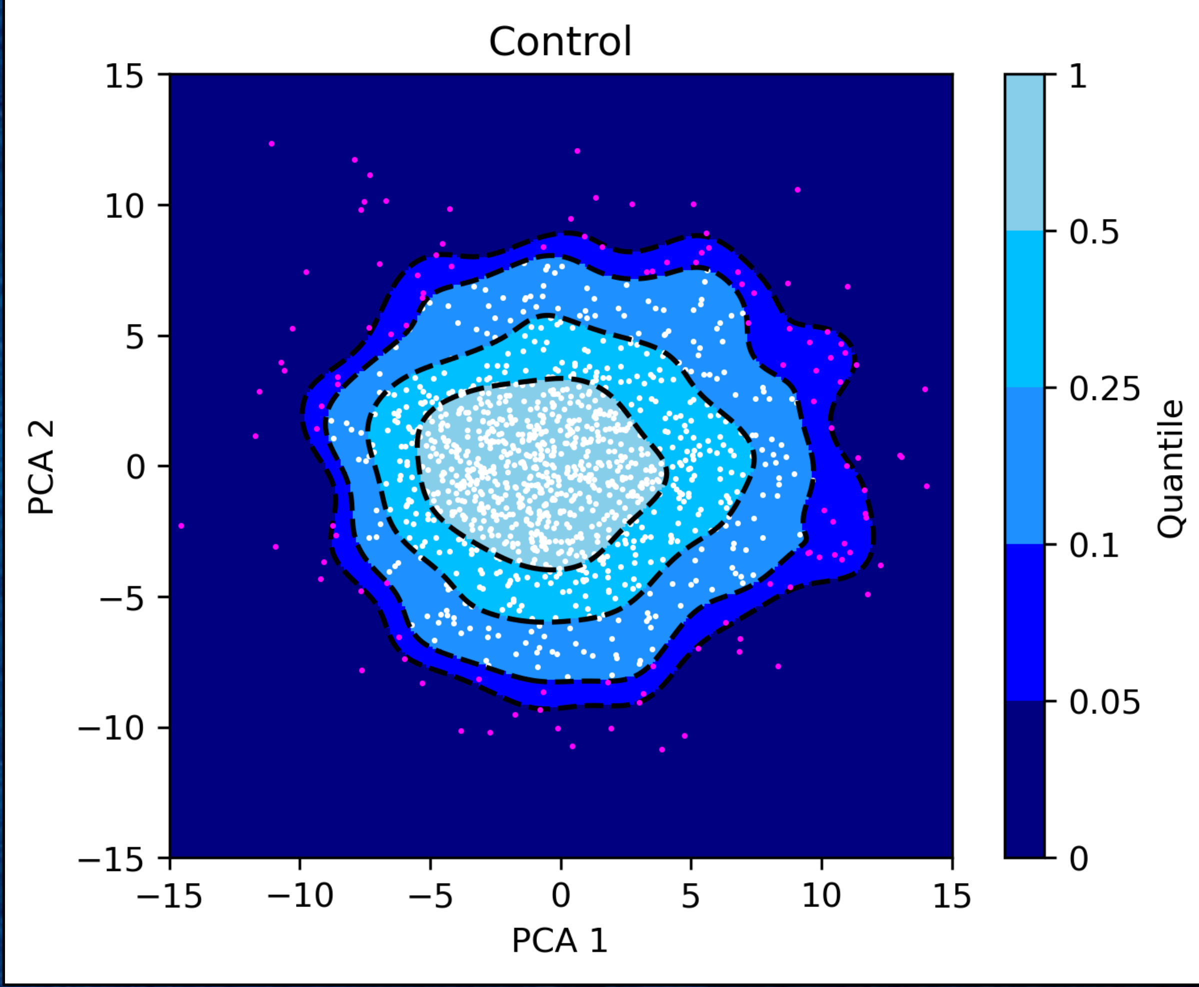
Preprocessing
Unsupervised Approach



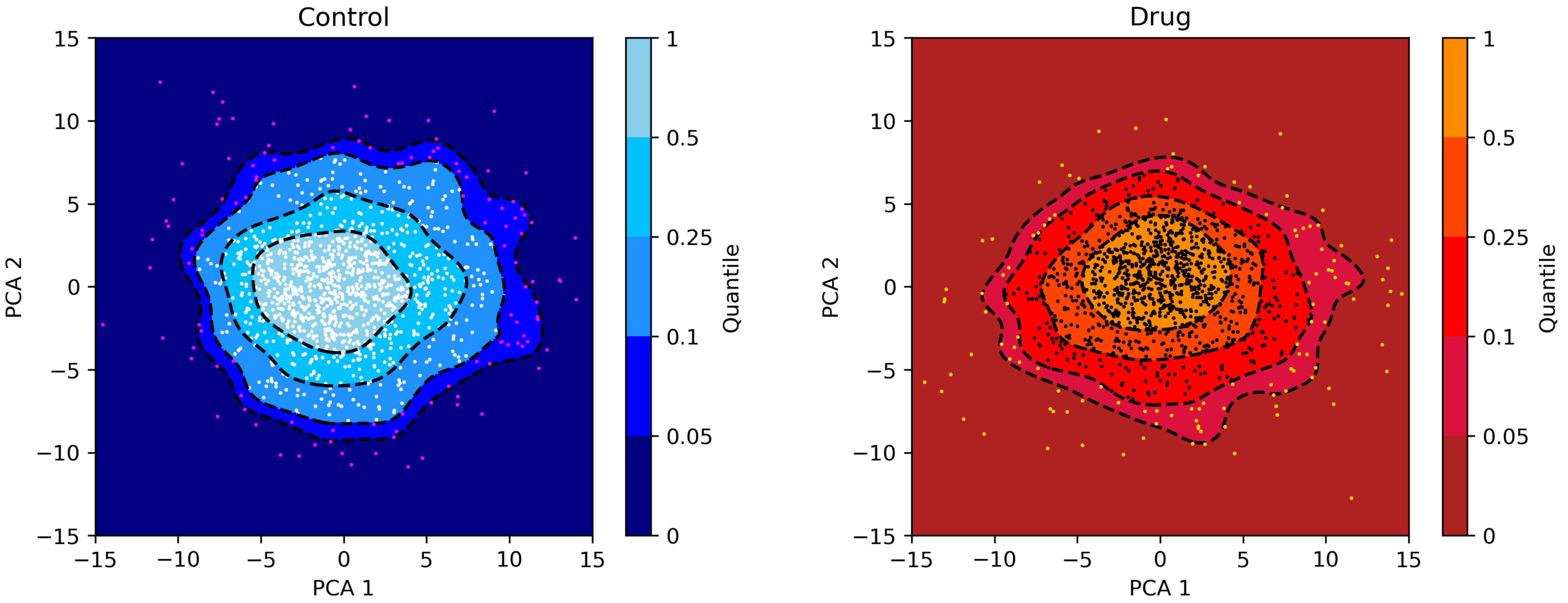
Preprocessing
Unsupervised Approach

Create KDEs in PCA space to estimate the distribution of samples.

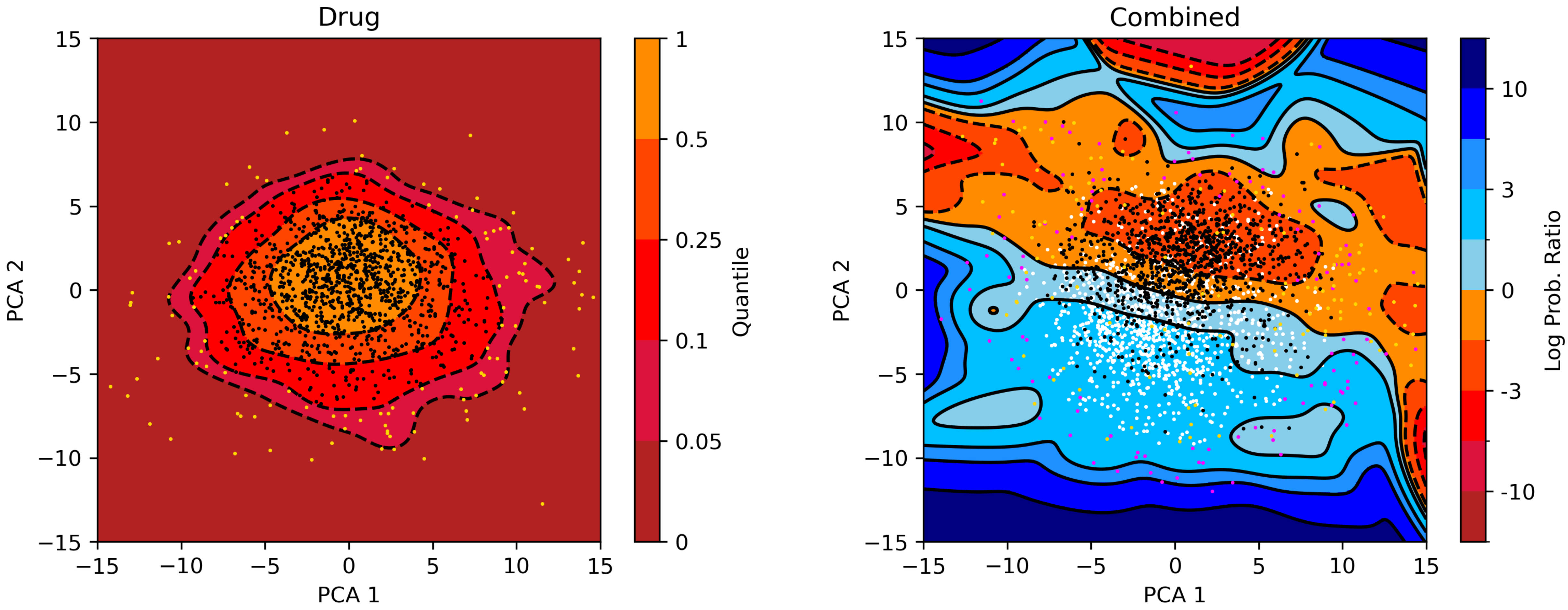
Samples with high likelihood are close to similar samples, while samples with low likelihoods aren't.



Preprocessing
Unsupervised Approach

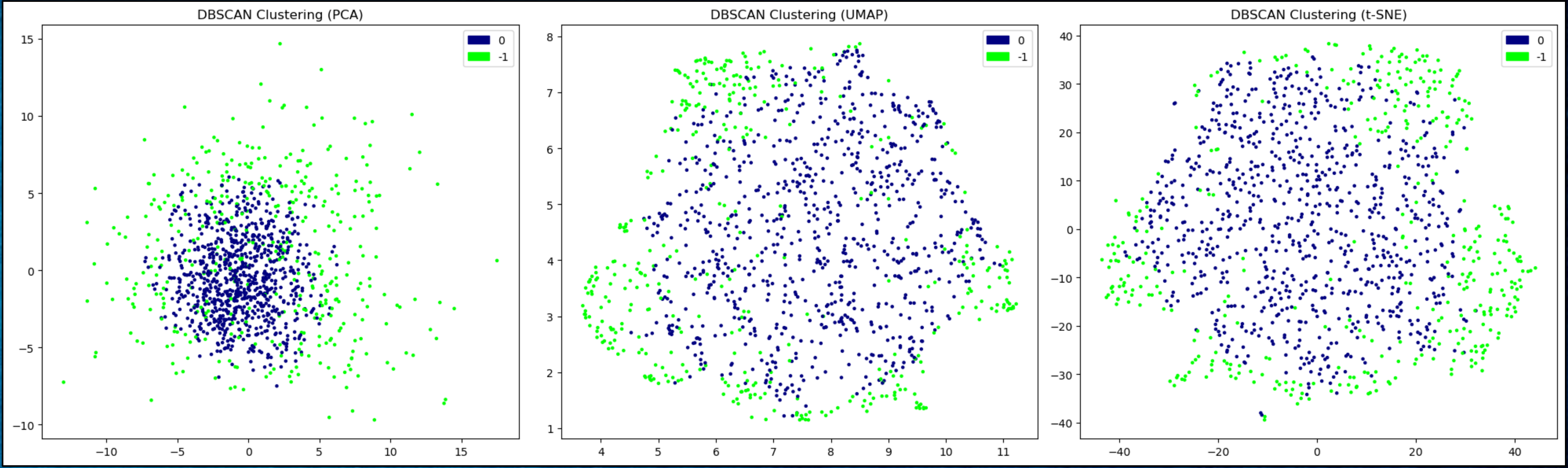


Preprocessing
Unsupervised Approach



Preprocessing

Unsupervised Approach

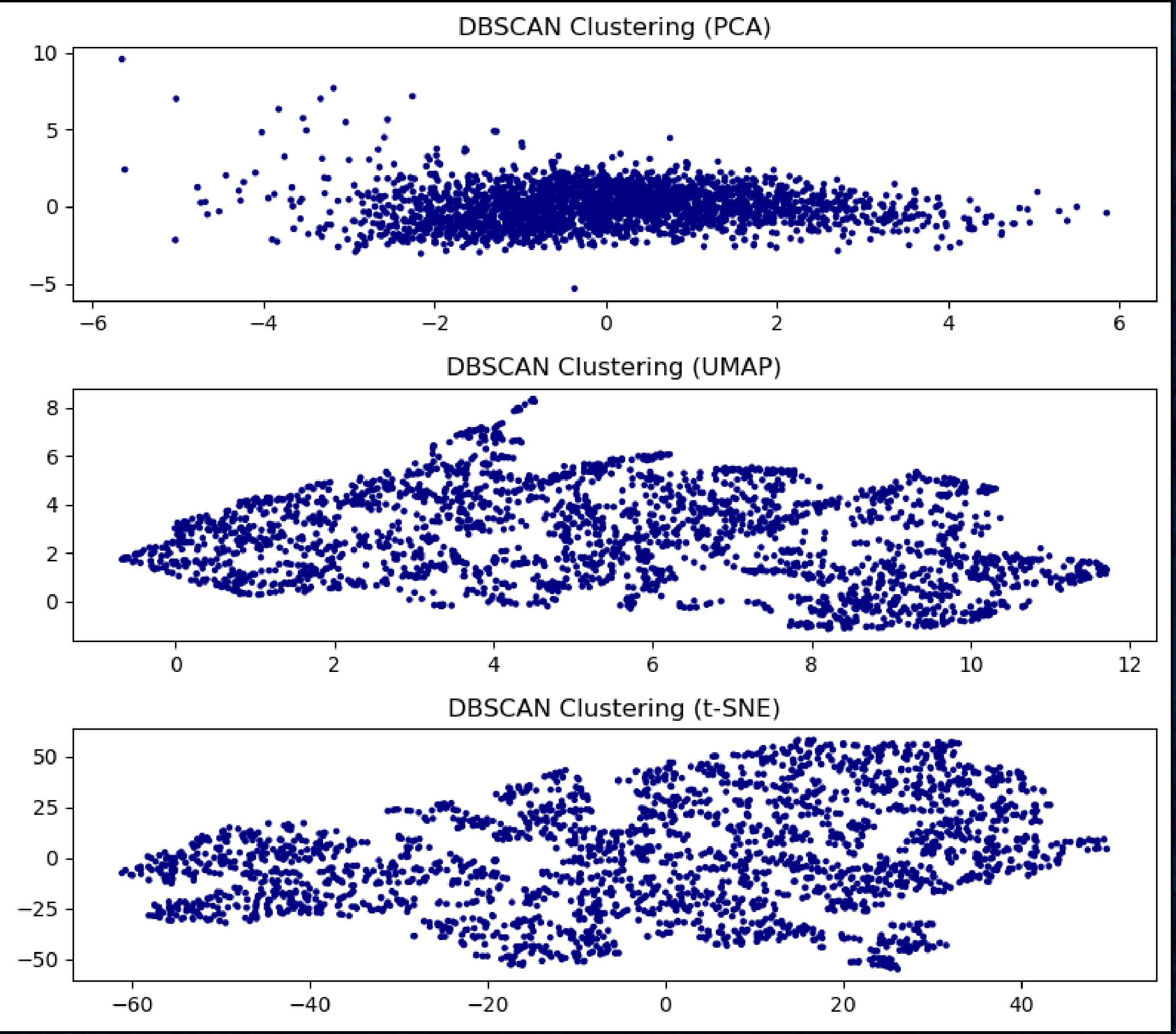


Preprocessing

Unsupervised Approach

What information can we get by applying unsupervised methods to all data?

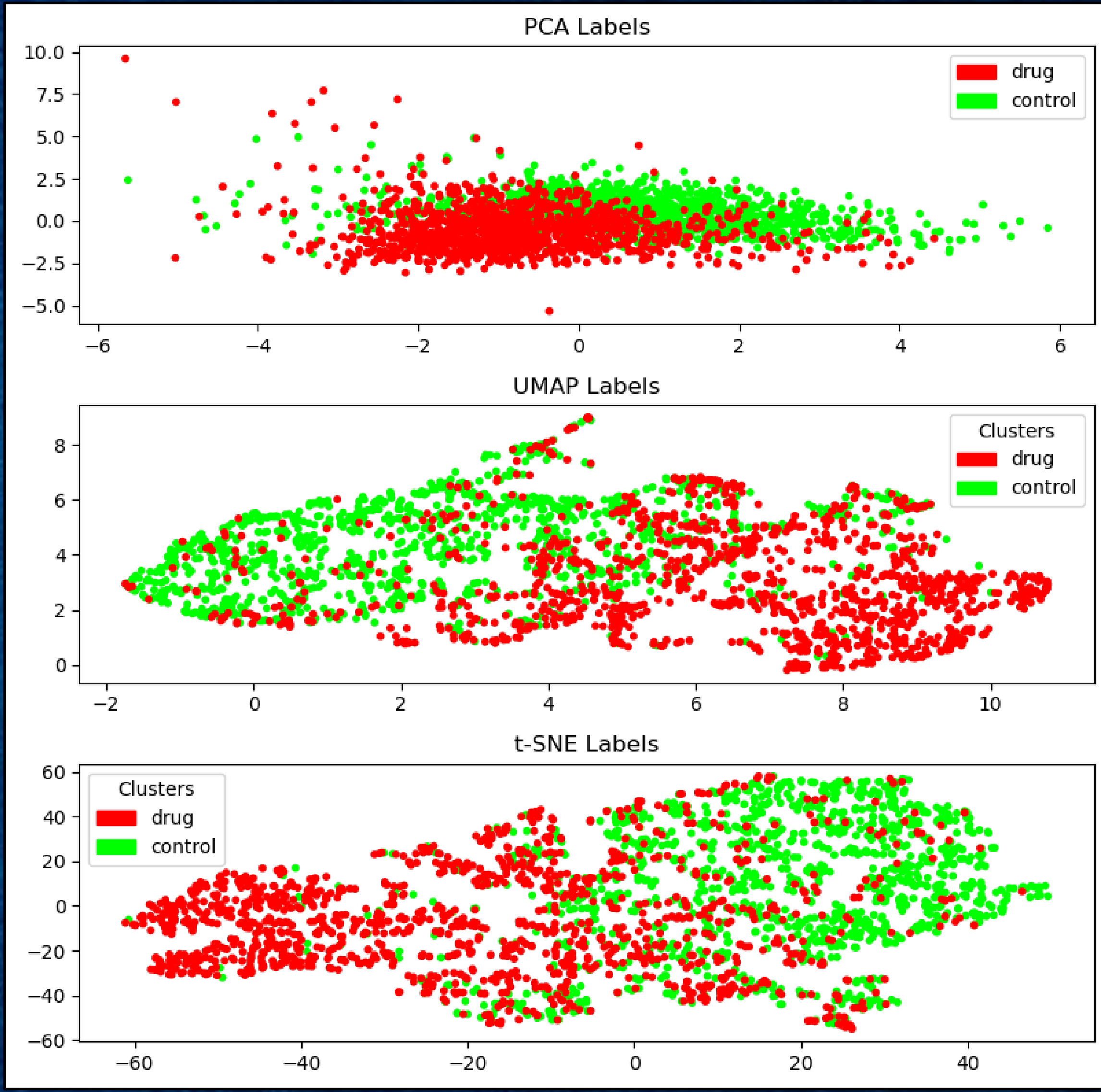
Dimensionality reduction algorithm show some separation but we don't get two distinct groups



Preprocessing
Unsupervised Approach

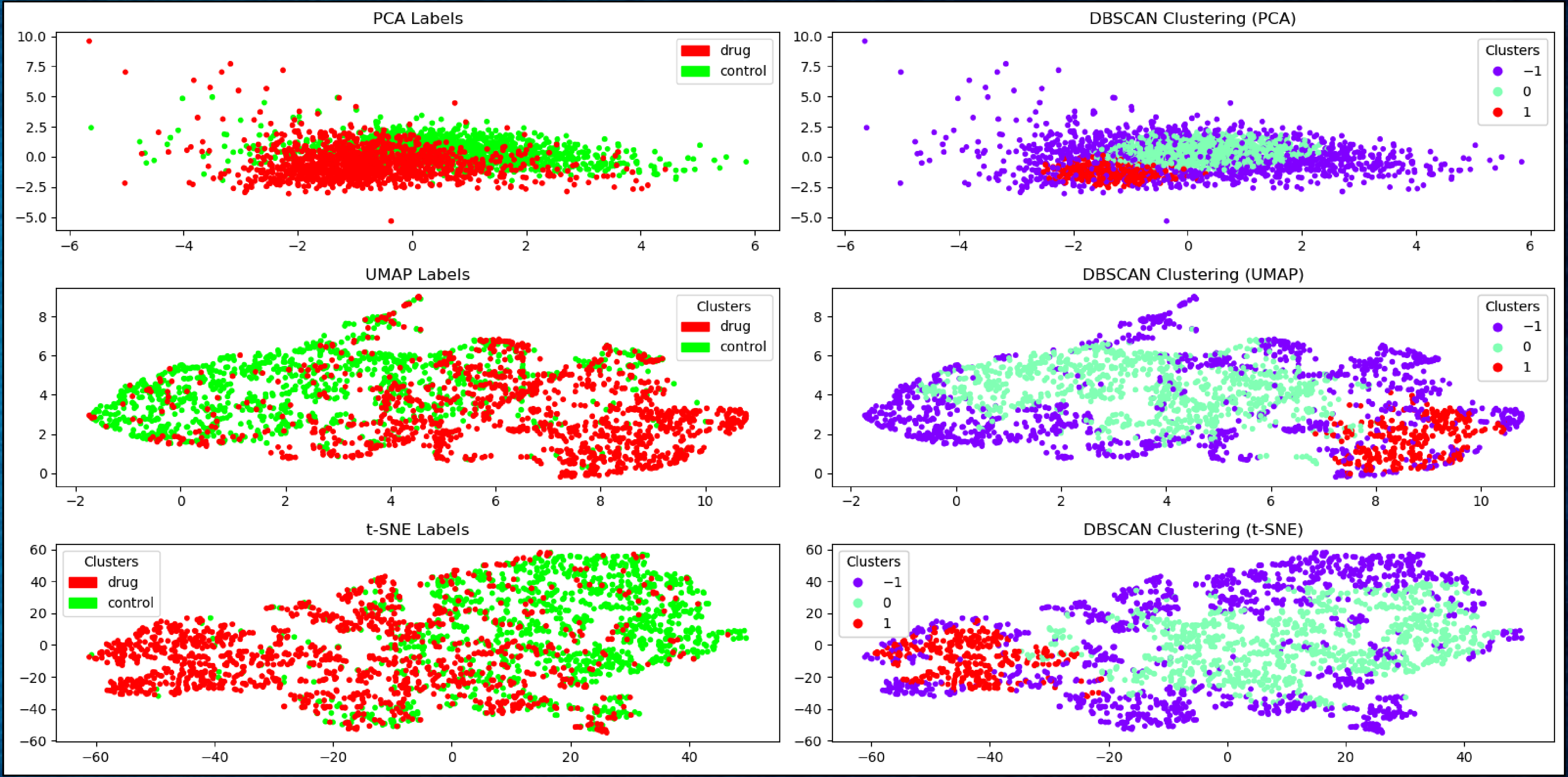
What information can we get by applying unsupervised methods to all data?

Dimensionality reduction algorithm show some separation but we don't get two distinct groups



Preprocessing

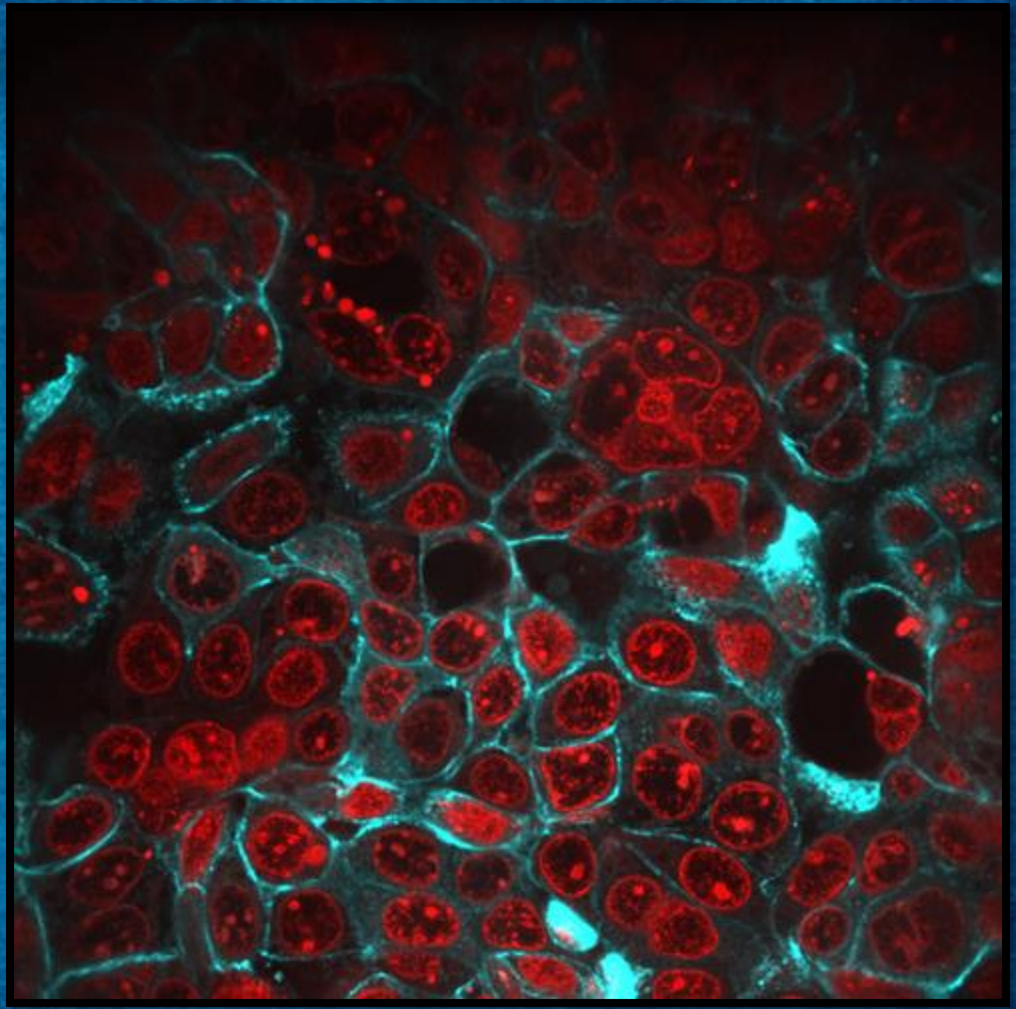
Unsupervised Approach



Implementation

Random Forest & XGBoost

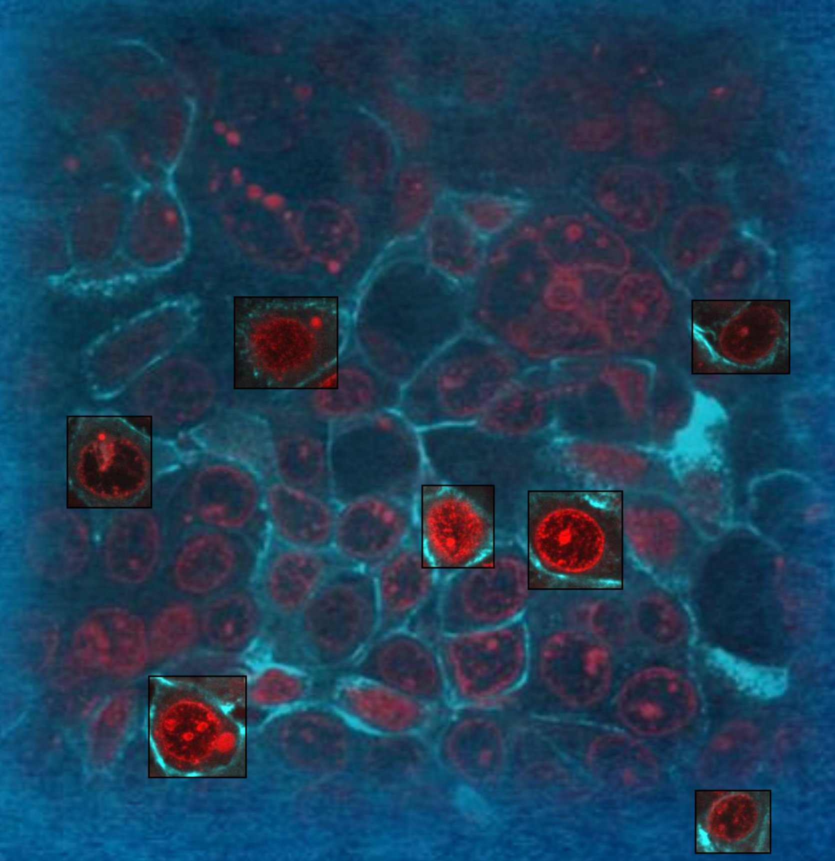
Original Image



Implementation

Random Forest & XGBoost

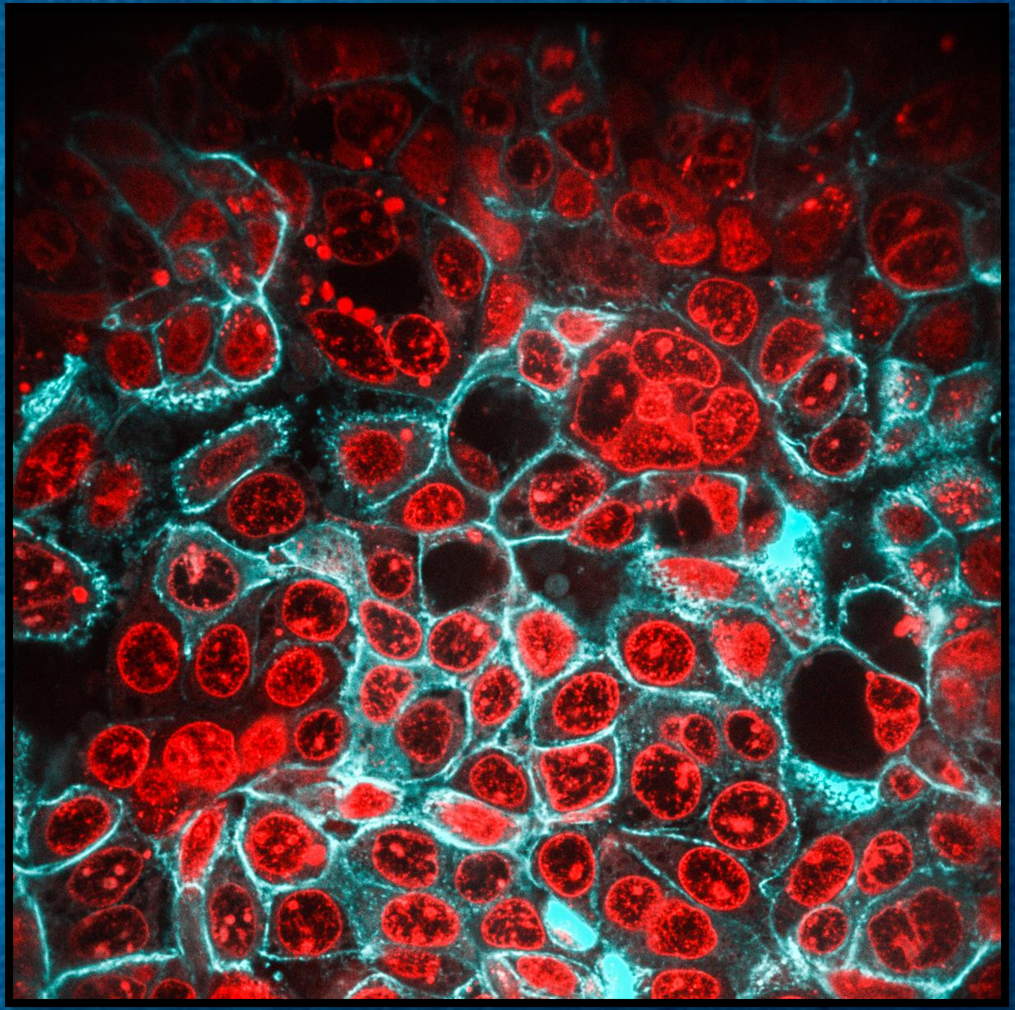
First Segmentation
Round



Implementation

Random Forest & XGBoost

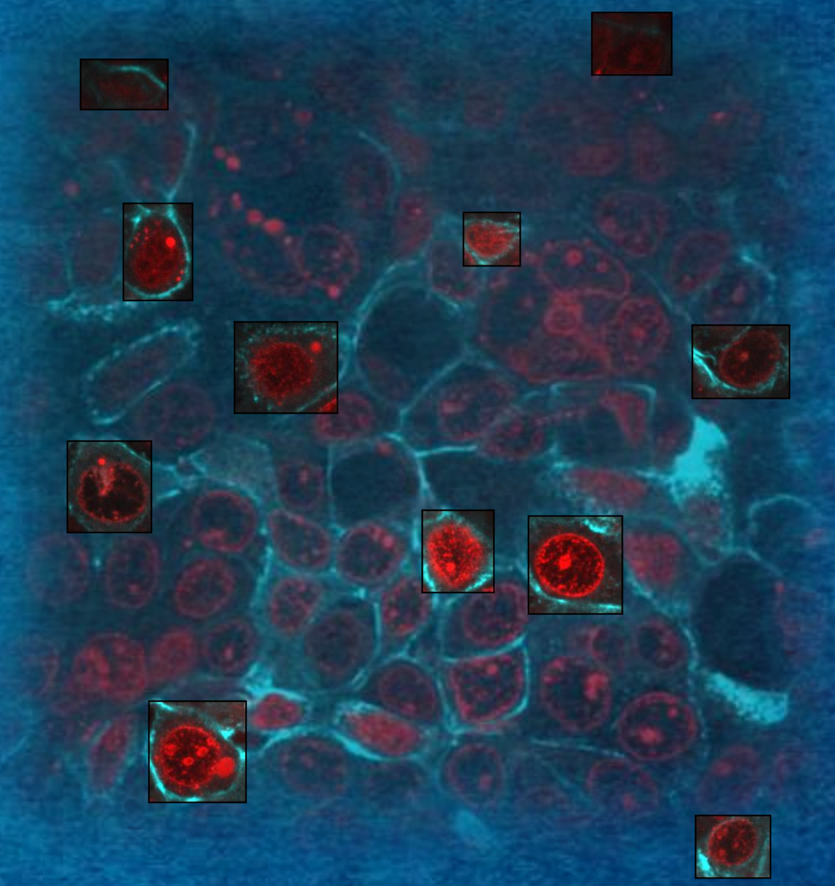
Brightened Image



Implementation

Random Forest & XGBoost

Second Segmentation Round



Implementation

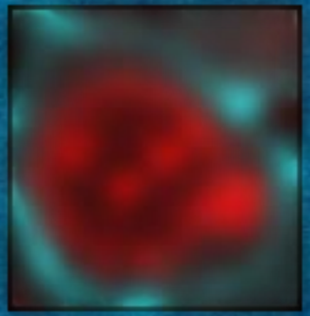
Random Forest & XGBoost

Color Correction &
Blurring

Implementation

Random Forest & XGBoost

Color Correction & Blurring

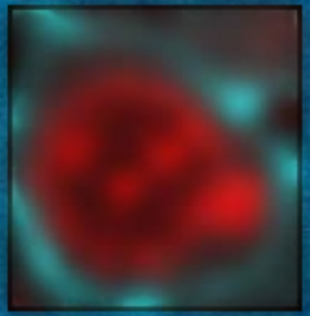


Features & DATA

Implementation

Random Forest & XGBoost

Color Correction & Blurring



Features & DATA

Tree Classifier

Implementation

Random Forest & XGBoost

	Features (raw)	Features (5% filter)	Features (15% filter)	Features (DBSCAN)	Features (clipped)
N features	76	76	76	76	76
Entries	2417	2297	2056	2193	2040
Val. Accuracy	**82.6 ± 0.7**	**81.3 ± 0.6**	**85.5 ± 0.6**	**85.4 ± 1.1**	**80.0 ± 0.8**
Val. Log Loss	0.39 ± 0.01	0.40 ± 0.01	0.34 ± 0.01	0.33 ± 0.01	0.43 ± 0.01
CV Folds	15	13	10	12	12
Test Accuracy	82.6	85.2	83.0	85.9	81.9
Test Log Loss	0.36	0.38	0.35	0.33	0.43
Elapsed time	58 min.	44 min	38 min.	50 min.	57 min.

Best Hyperparameter values using hyperopt: {'max_depth': 13, 'max_features': None, 'max_samples': 0.5182730111149145, 'min_samples_leaf': 0.001717253590231625, 'min_samples_split': 0.004193785663888986, 'n_estimators': 2850}

Implementation

Random Forest & XGBoost

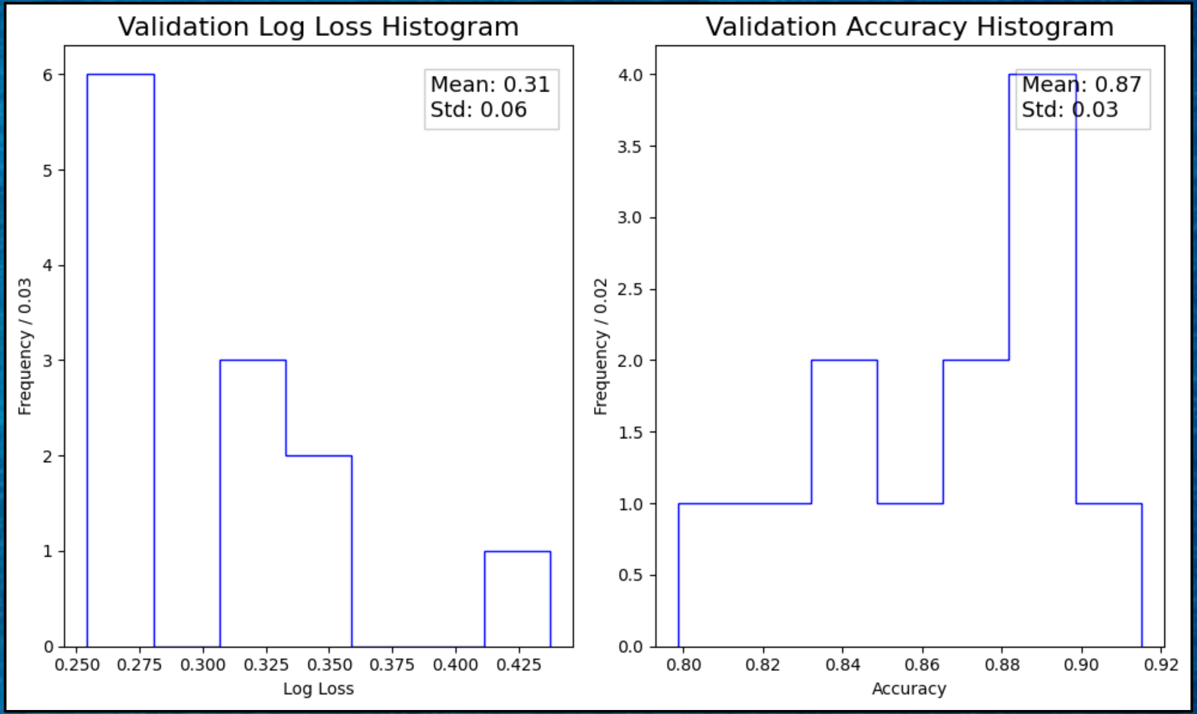
	Features (raw)	Features (5% filter)	Features (15% filter)	Features (DBSCAN)	Features (clipped)
N features	76	76	76	76	76
Entries	2417	2297	2056	2193	2040
Val. Accuracy	**85.4 ± 0.7**	**85.1 ± 0.8**	**86.6 ± 1.0**	**86.6 ± 1.0**	**84.3 ± 0.5**
Val. Log Loss	0.33 ± 0.01	0.33 ± 0.01	0.32 ± 0.02	0.31 ± 0.01	0.34 ± 0.01
CV Folds	15	13	10	12	12
Test Accuracy	85.1	85.2	85.9	87.3	82.4
Test Log Loss	0.29	0.33	0.33	0.29	0.39
Elapsed time	206 min.	141 min.	91 min.	101 min.	126 min.

Best Hyperparameter values using hyperopt: {'colsample_bytree' 0.2046844879644485, 'dropout' 4.0352406651943293e-07, 'learning_rate' 0.005490611984145916, 'lr_decay' 0.0021099261128270153, 'max_depth' 5, 'min_child_weight' 2.486367120206436e-08, 'n_estimators' 1950, 'reg_lambda' 0.00030284912898972 63, 'subsample' 0.3901849425407928 3}

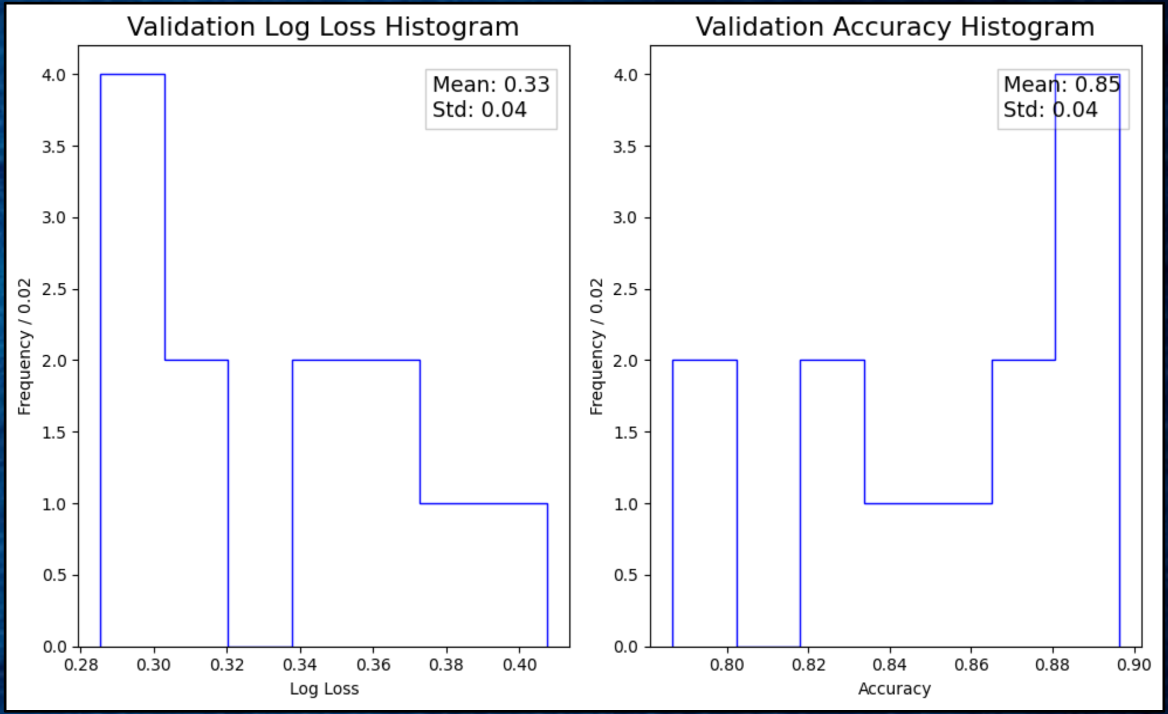
Implementation

Random Forest & XGBoost

XGBoost

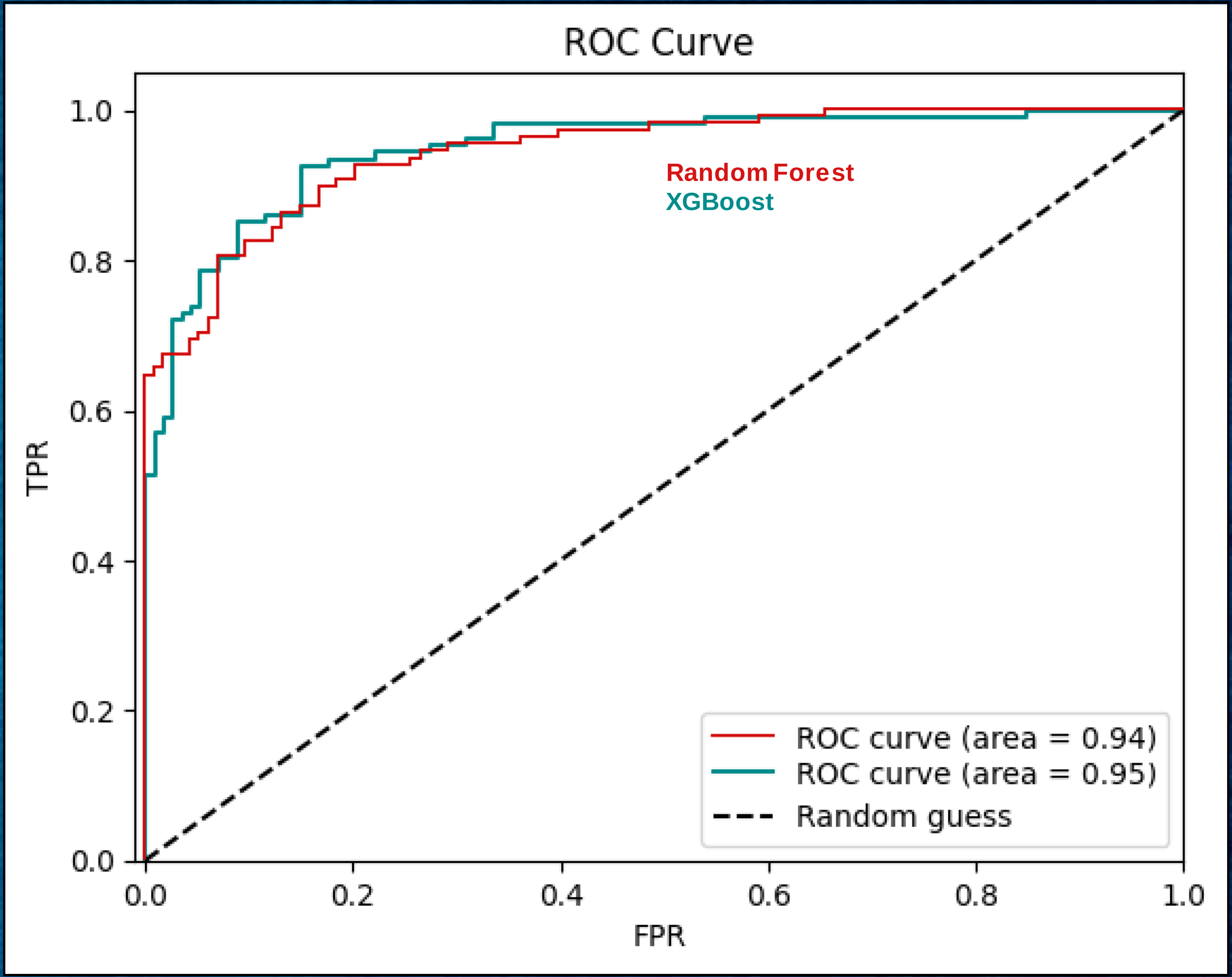


Random Forest



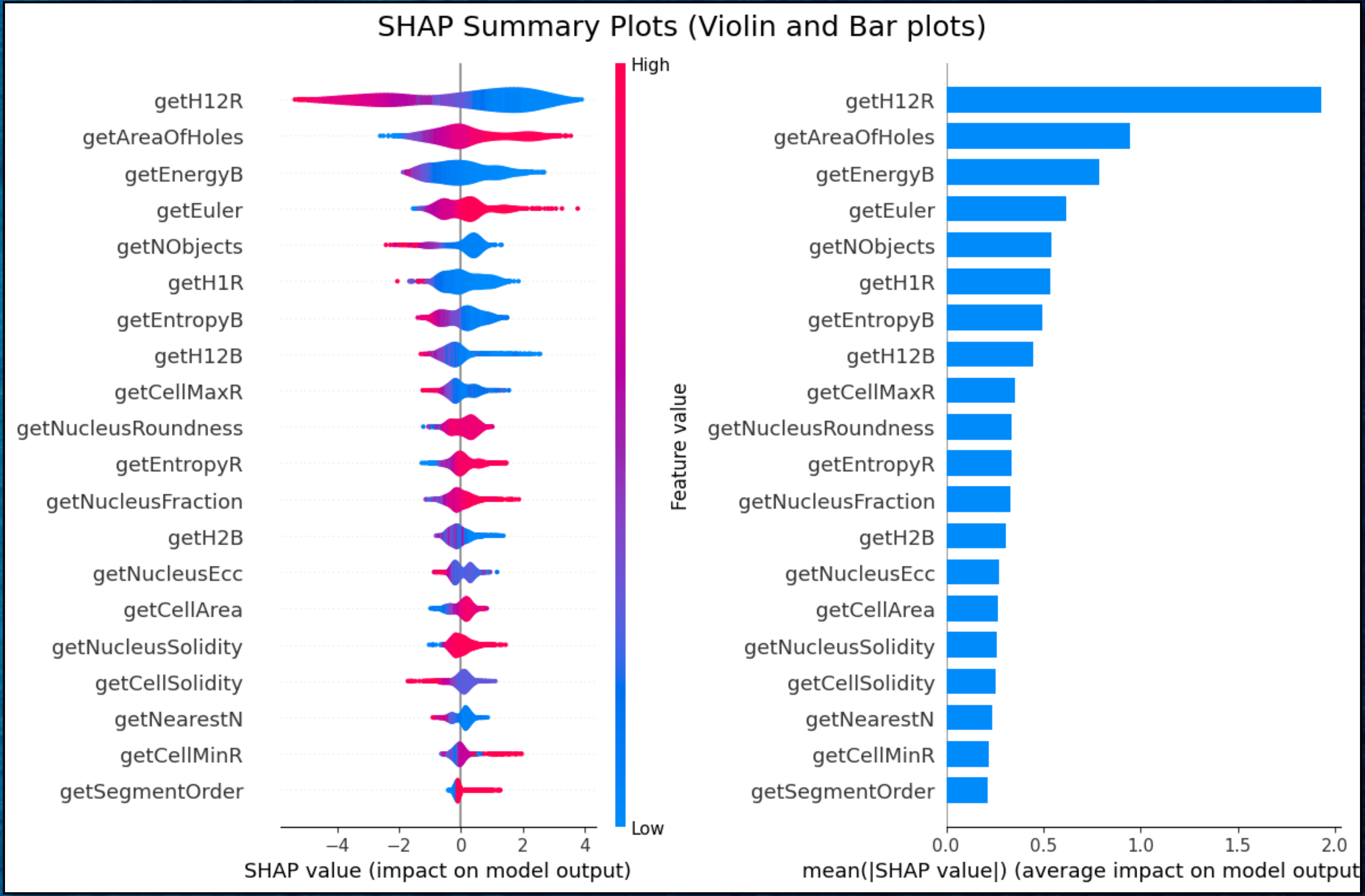
Implementation

Random Forest & XGBoost



Implementation

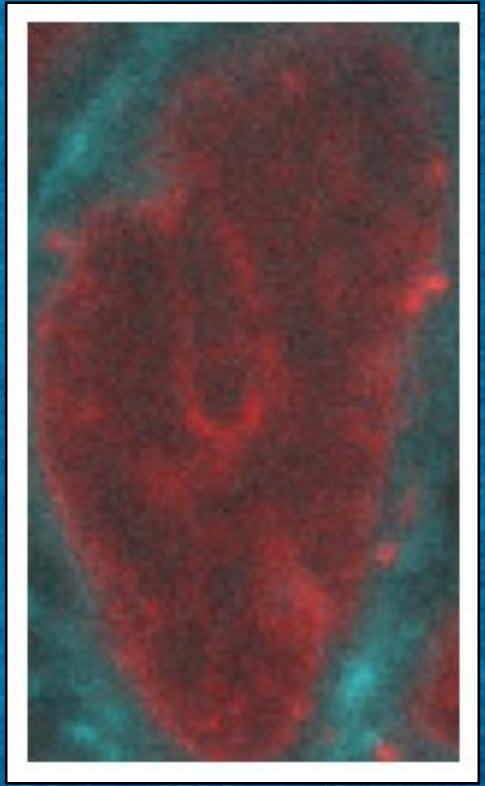
Random Forest & XGBoost

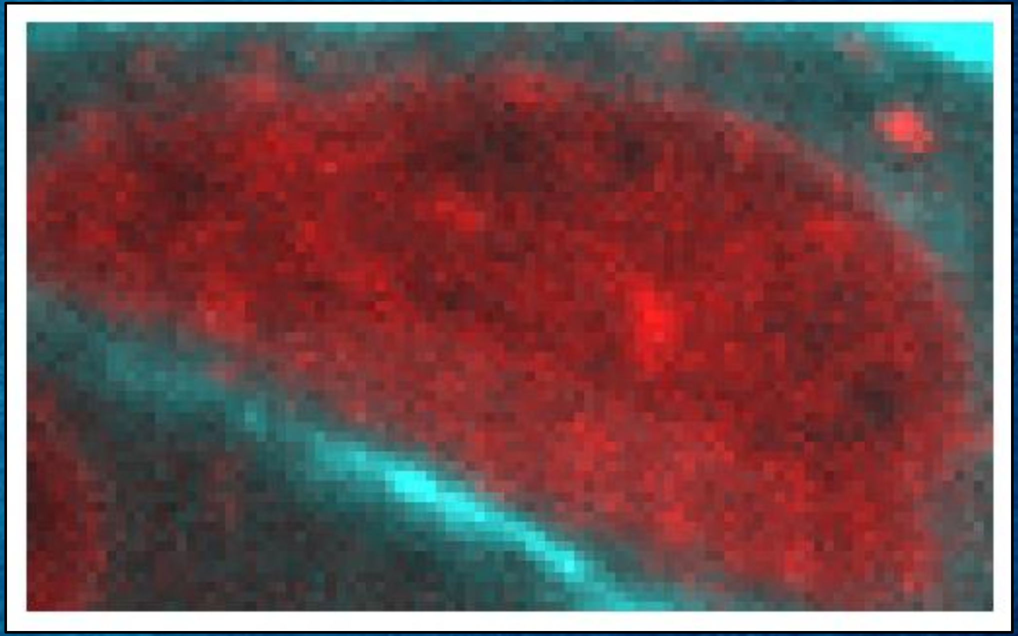


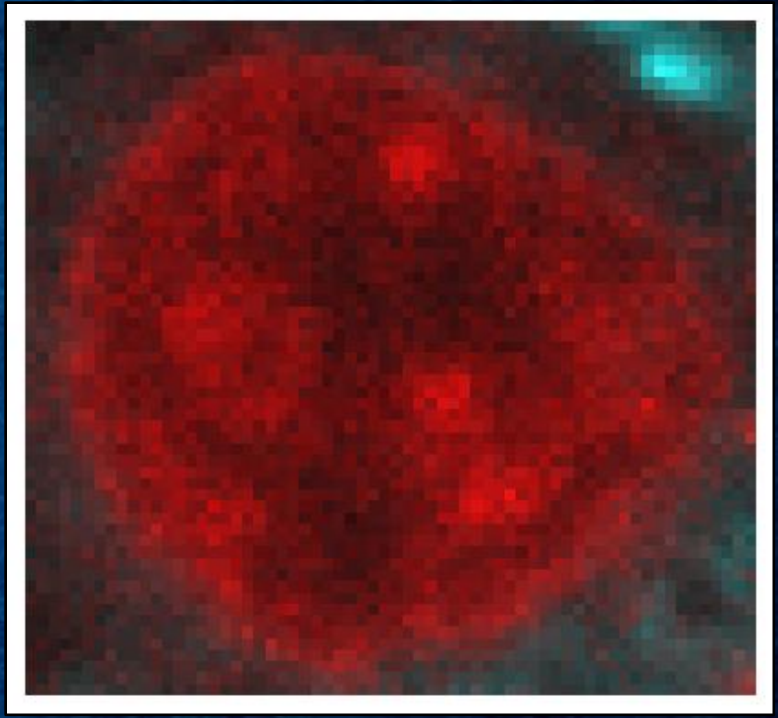
Implementation
CNN

Data overview

- Padding images to square and rescale to median





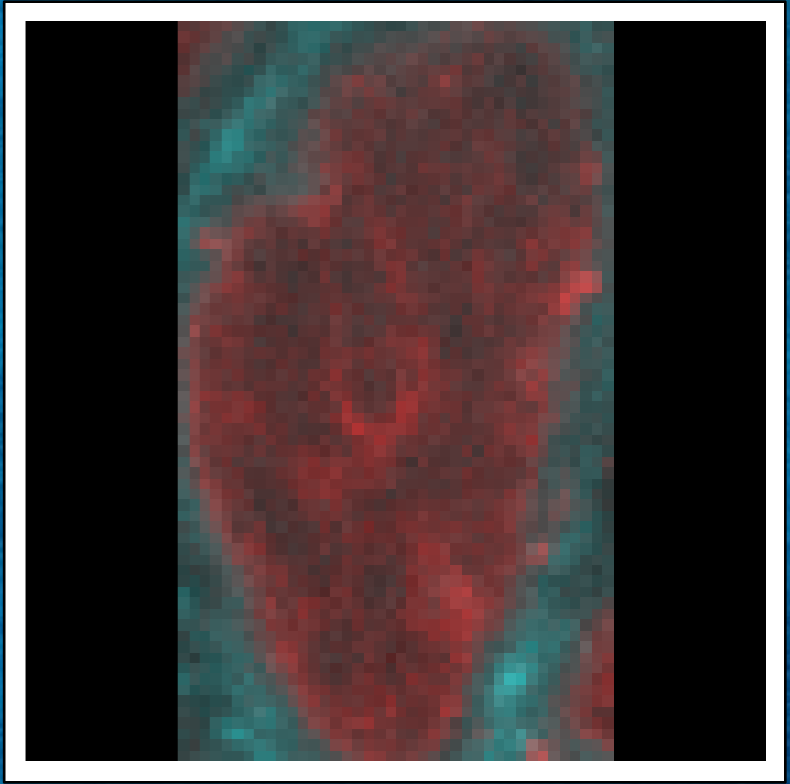
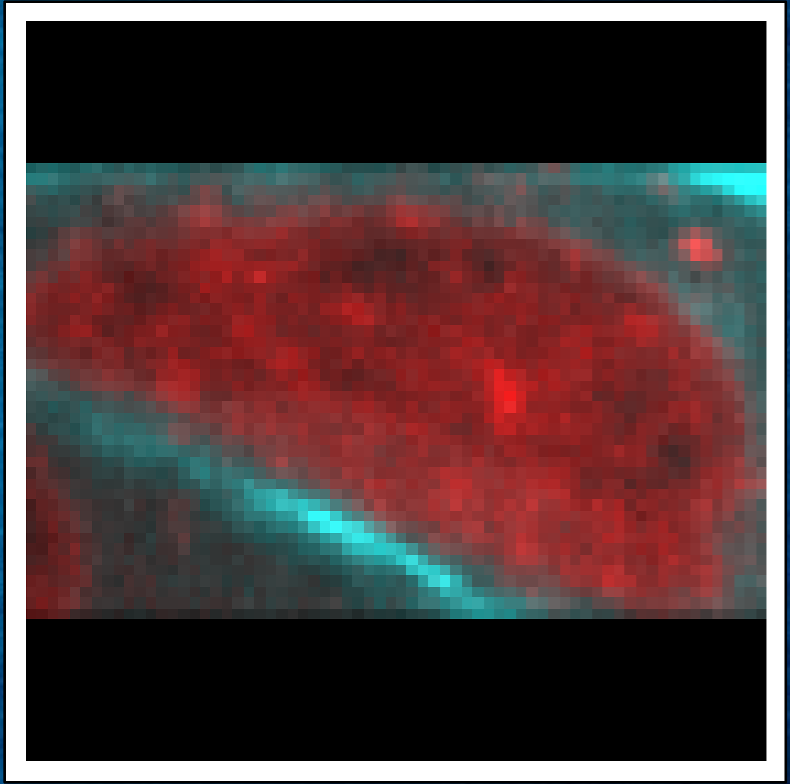
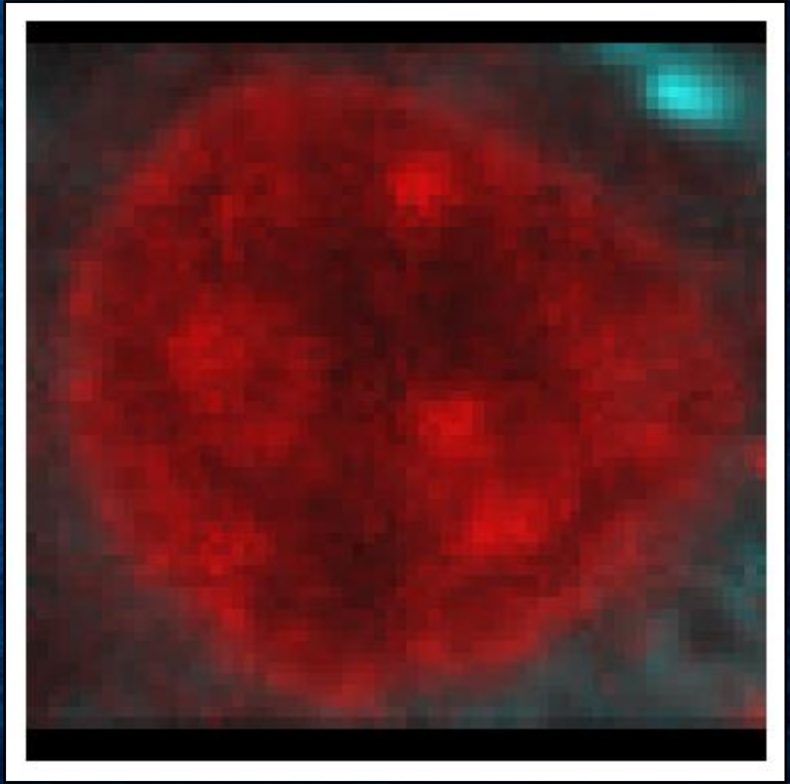


Implementation

CNN

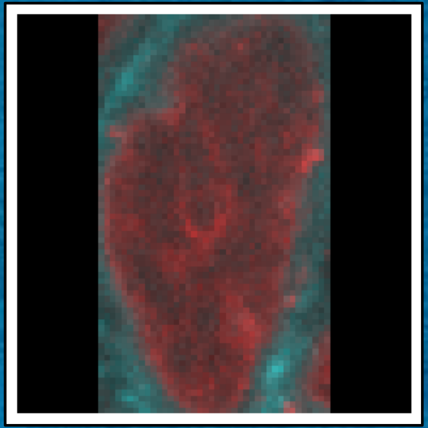
Data overview

- Padding images to square and rescale to median

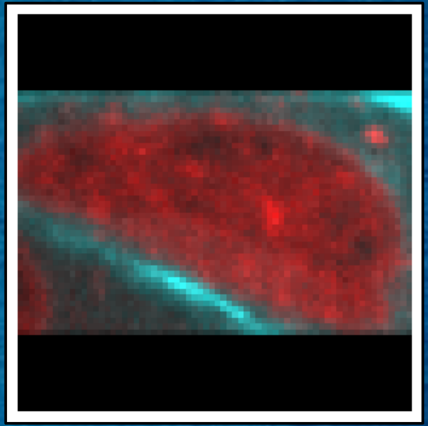
Implementation
CNN
Data overview

- Padding images to square and rescale to median



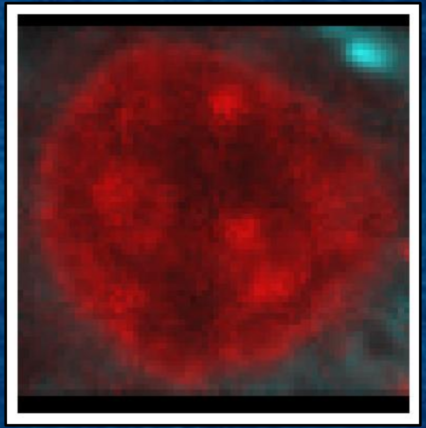
Scaling factor:

1.022



Scaling factor:

0.701



Scaling factor:

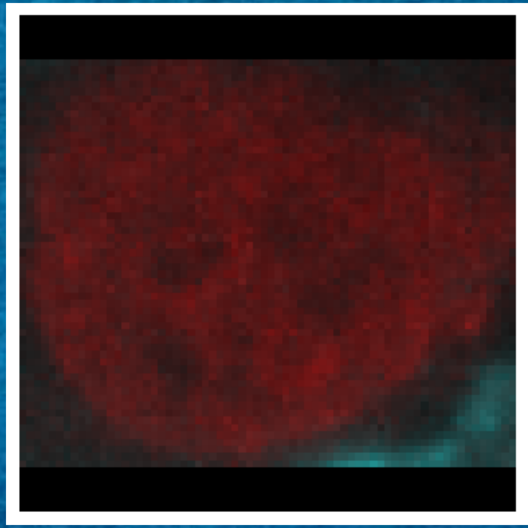
1.062

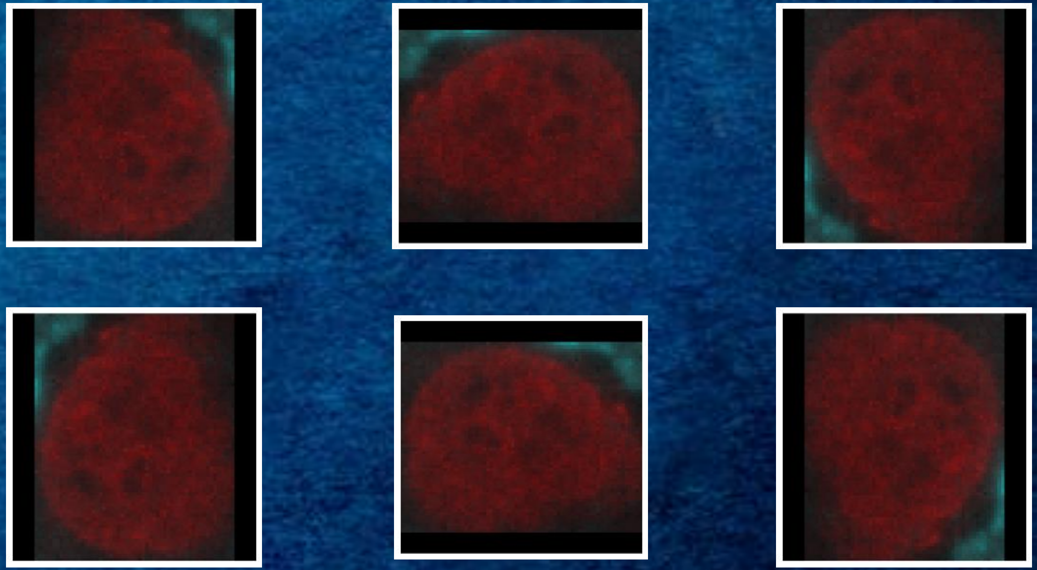
- Saving scaling factors for later use in the CNN

Implementation
CNN

Data overview

- Padding images to square and rescale to median
- Saving scaling factors for later use in the CNN
- Rotating and flipping the images for data augmentation



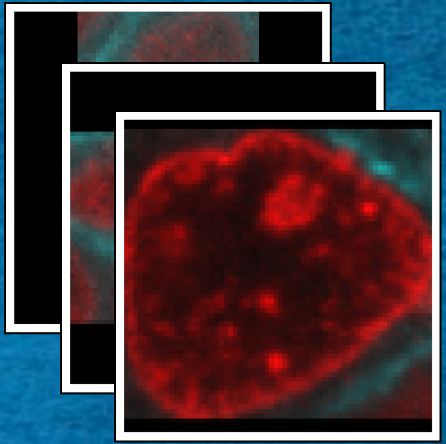


Implementation

CNN

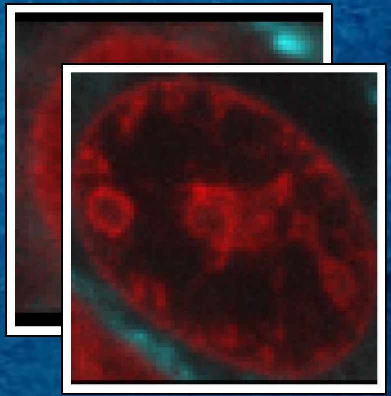
Data overview

- Padding images to square and rescale to median
- Saving scaling factors for later use in the CNN
- Rotating and flipping the images for data augmentation
- Splitting images to training, validation and test sets



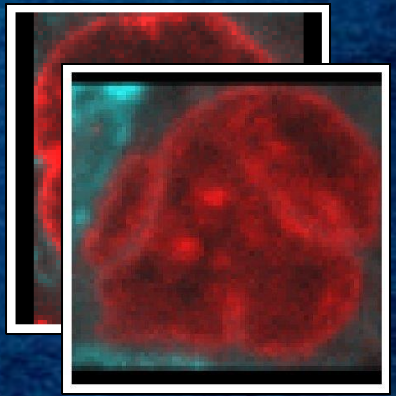
Training

0.8



Validation

0.1



Testing

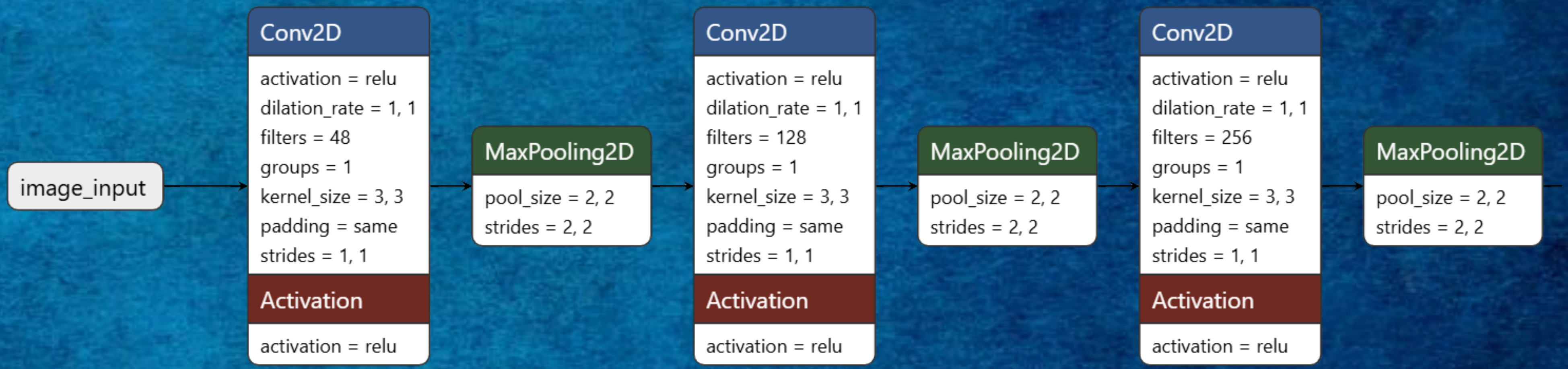
0.1

Implementation
CNN
Model architecture

Implementation

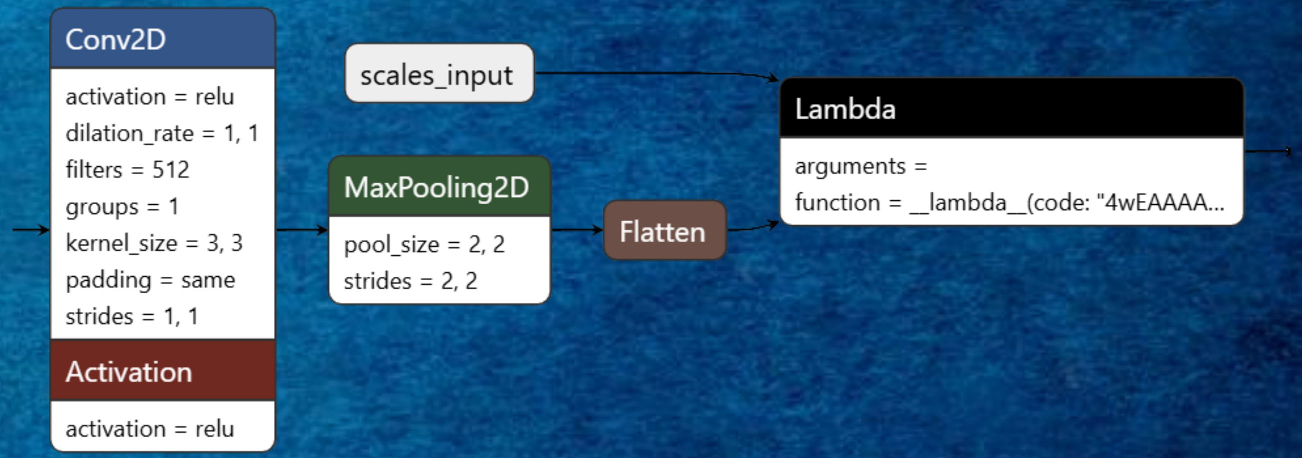
CNN

Model architecture



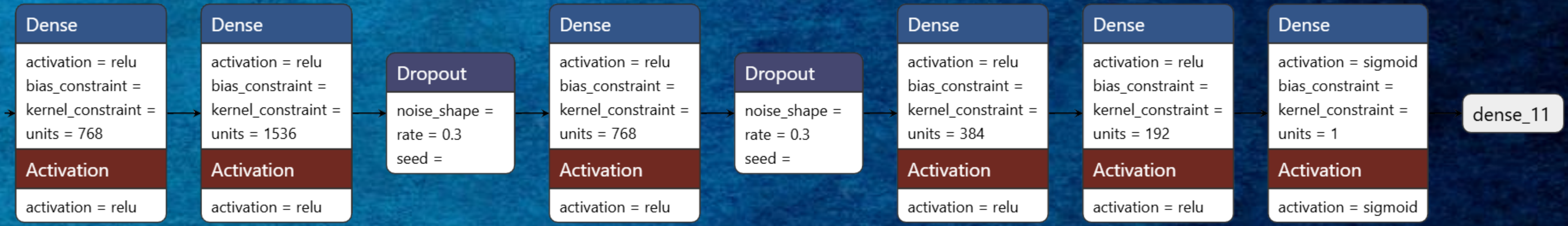
Implementation
CNN

Model architecture



Implementation
CNN

Model architecture



Implementation

CNN

Model architecture – Hyperparameter Bayesian optimization from keras_tuner.

Hyperparameters:

- Number of Convolutional layers
- Number of filters in each layer
- Number of Dense layers
- Nodes in each layer
- Dropout rate
- Learning rate of the optimizer

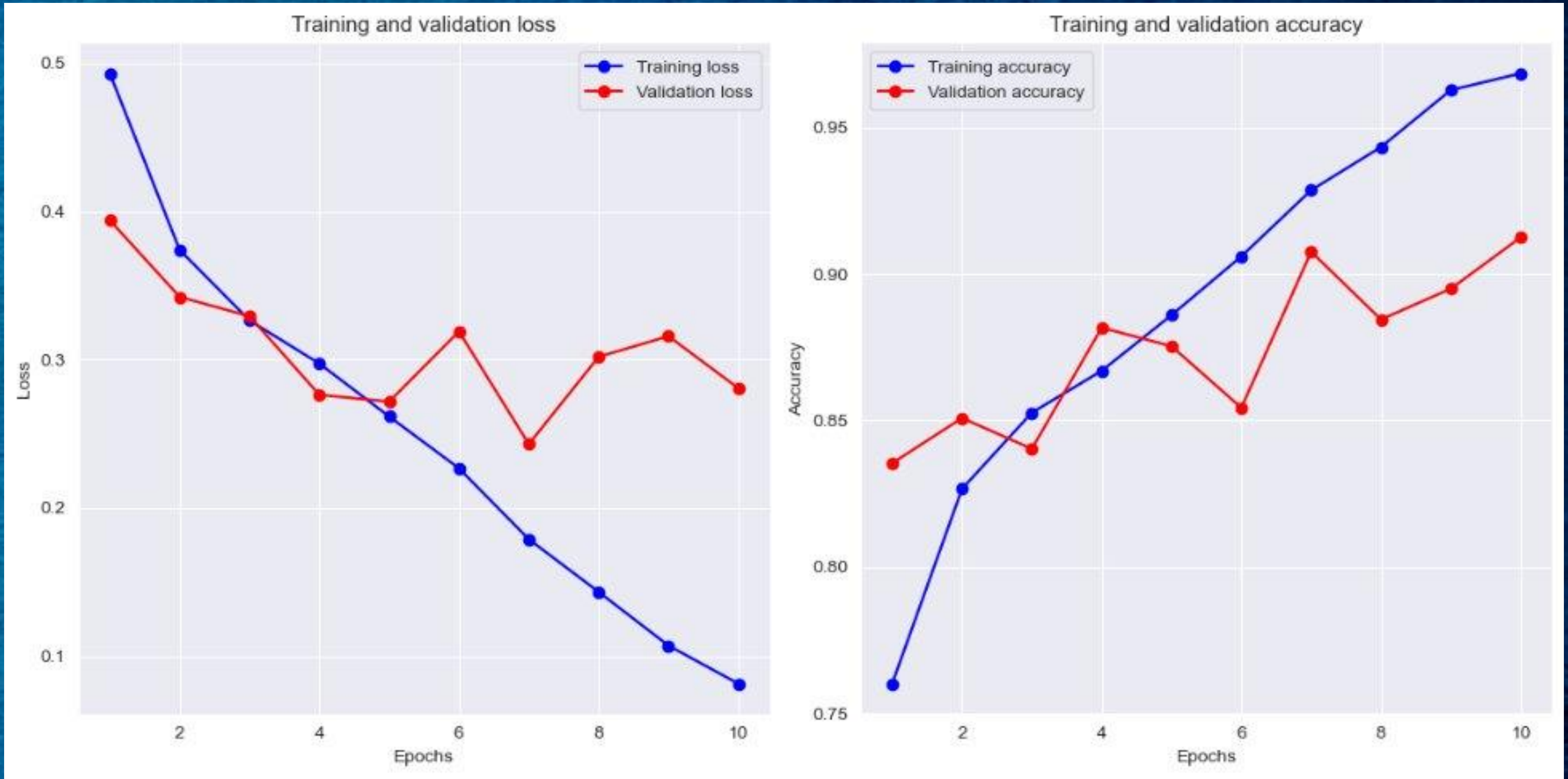
~ 5 hours

Didn't work out
10% less accuracy than the original guess

Implementation
CNN

	Accuracy
Features(raw):	0.874
Features (DBSCAN):	0.885
Manual:	0.912

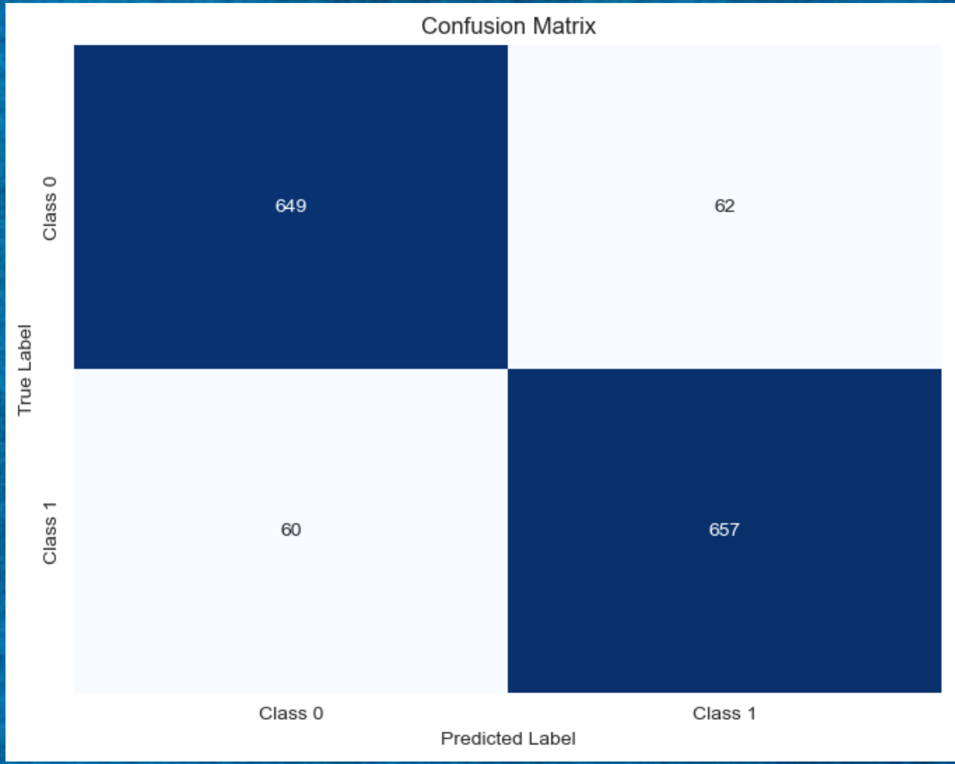
Implementation
CNN
Training process

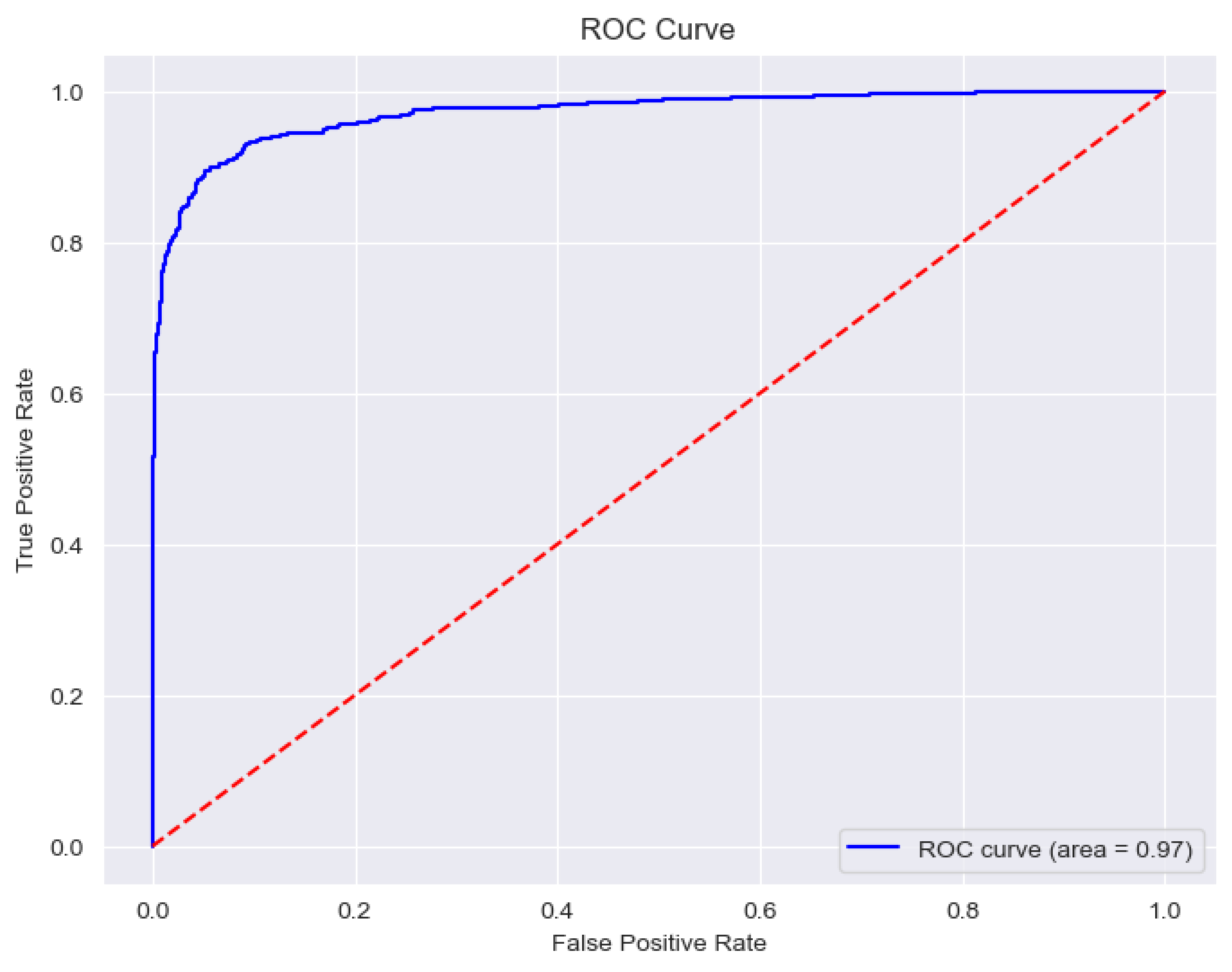


Implementation

CNN

Model performance





Average accuracy with 5 cross validation folds: 0.914

Remarks

- Preprocessing is important
- The unsupervised approach gave us insights about the data
- XGBoost performs better than the Random Forest
- The 12th Haralick feature impacted our model the most
- The outlier identifications worked in our favour
- CNN beats both Tree implementations but lacks the feature importance ranking

Failed Attempts & Future Work

- Simulating Data
  - VAE
- ResNet18
  - Better Image Processing and Segmentation
  - Improved Outlier Identification
  - Combined CNN and XGBoost Implementation

Failed Attempts & Future Work

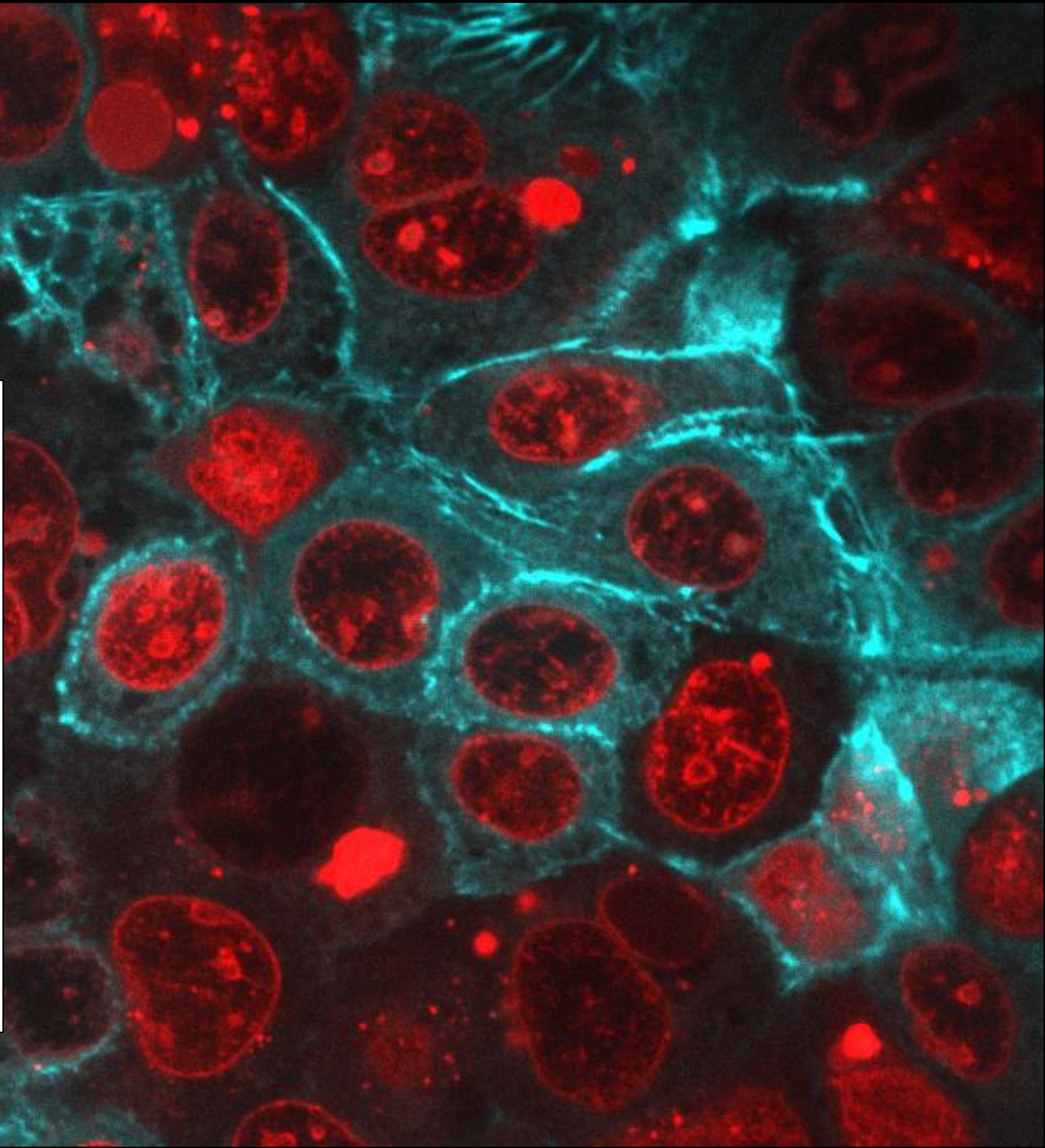
- Simulating Data
- VAE
- ResNet18

- Better Image Processing and Segmentation
- Improved Outlier Identification
- Combined CNN and XGBoost Implementation

Thank you for your attention!

Questions?

UNIVERSITY OF COPENHAGEN



Appendix
Realistic Expectations

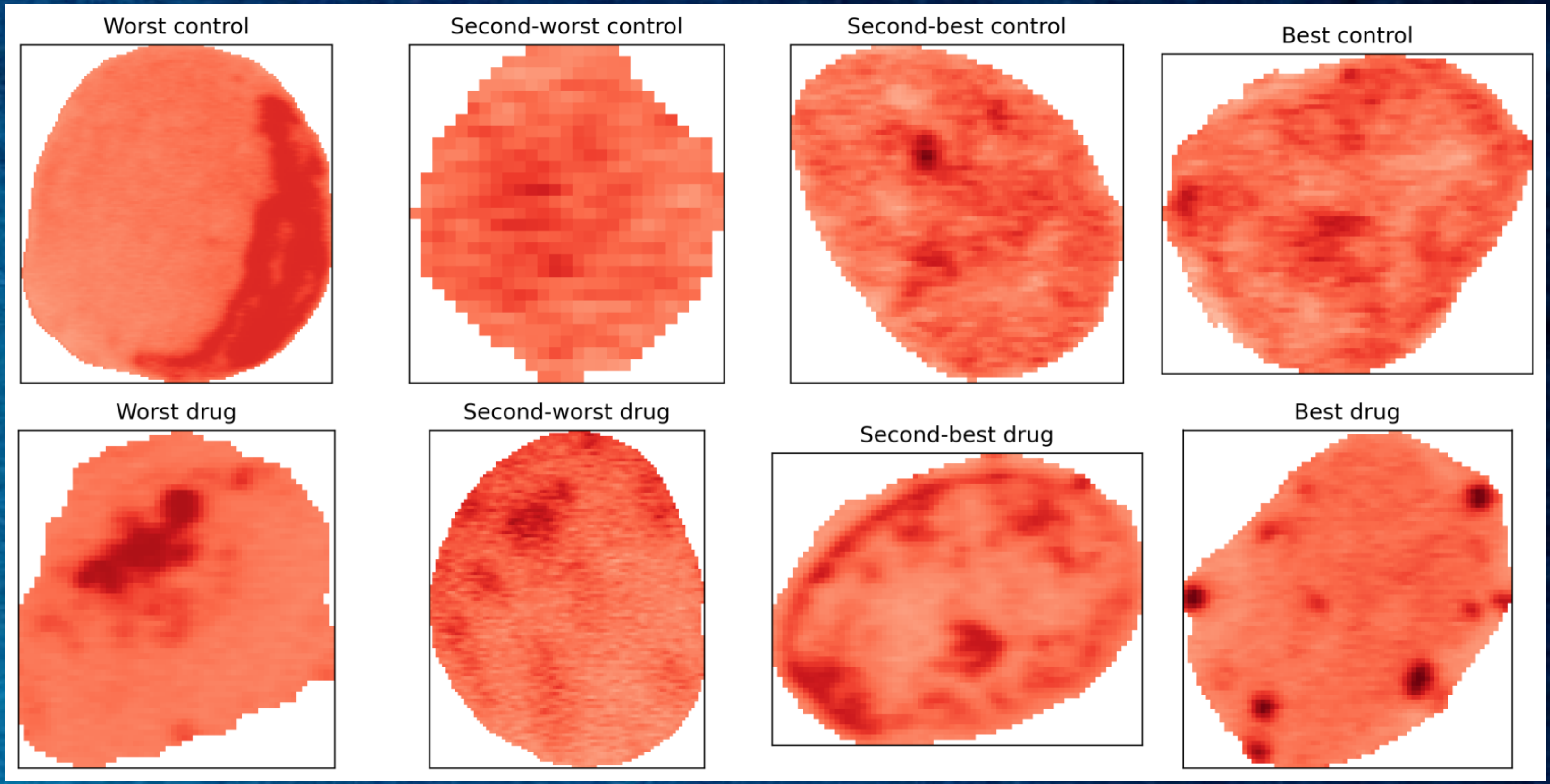
As we're dealing with living cells, we're dealing with a high number of uncertainties (these cells are known for being inhomogeneous, and therefore the segmentation step was crucial for the final outcome):

1. Not all control samples are guaranteed to be healthy.
2. Not all drug samples are guaranteed to be damaged.
3. There are cells undergoing mitosis (cell splitting: dramatic change of the nucleus characteristics)

We can therefore never expect an accuracy of 100%, unless we select the images by hand (which isn't machine learning). Our goal was to get the best out of it by engaging with no manual means (i.e. selecting the "good" images from the "bad" ones or even clipping the shadowed regions out).

Appendix

Outlier Identification



Using the probabilities given by the KDEs, we were able to identify the worst- and best-fitting nuclei. Although the control results make sense, the drug images shows the complexity of the reduced feature space.

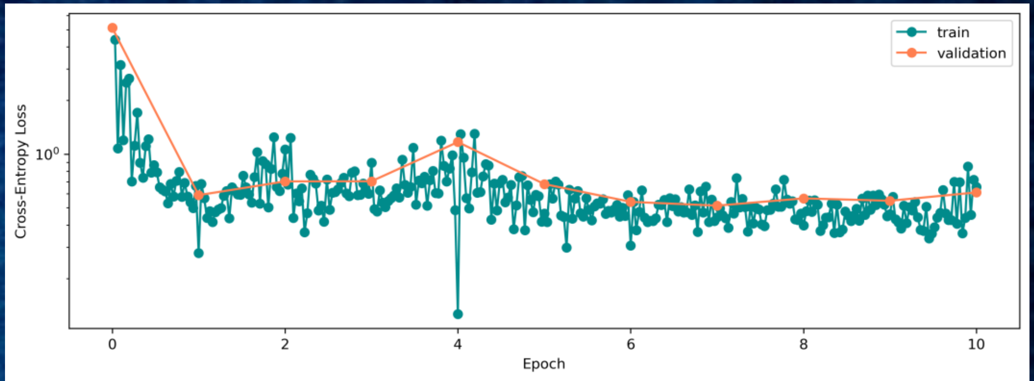
Appendix

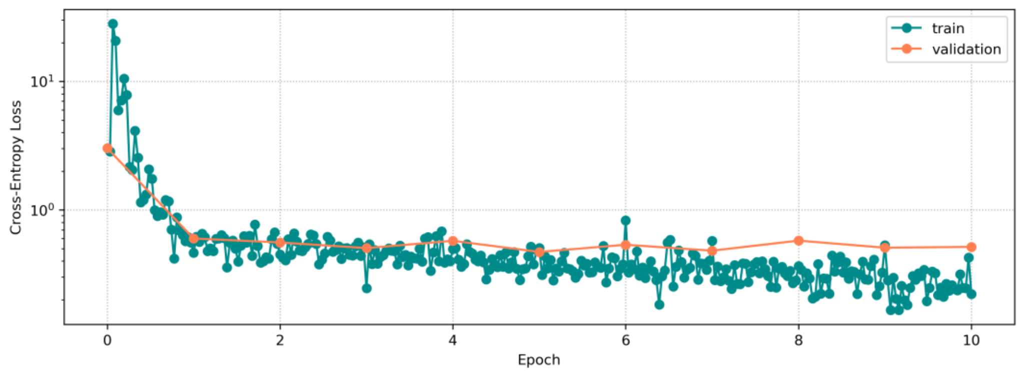
ResNet-18

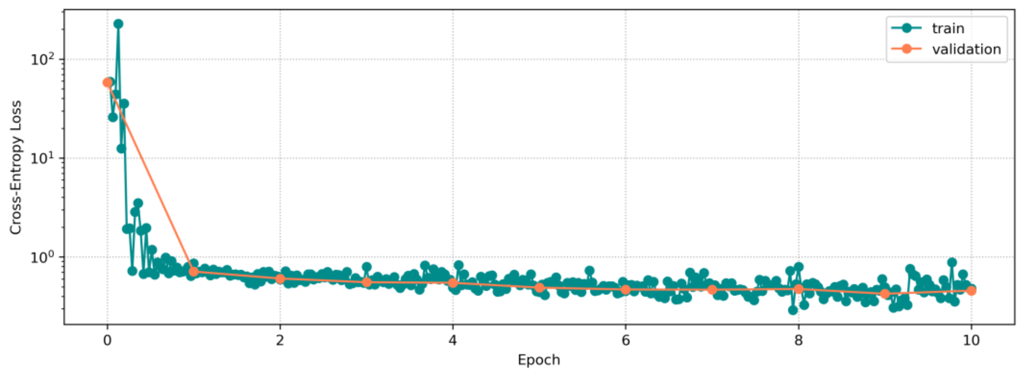
ResNet-2k: Pretrained ResNet-18 CNN with an additional linear layer to convert from an output of 1000 to 2. Only ~2000 parameters to train.

ResNet-500k: More trainable layers (~500k parameters).

ResNet-heavy: Fully trainable ResNet-18 (~15M parameters).







Appendix

VAE

Whole Images: The original sized Images (1200, 1200, 3) did not yield any results due to limited resources (out of memory).

Segmented Images: Even though it run without crashing, this implementation did not yield any fruitful results. Just 3 convolutional layers with 64, 128 and 256 filters and one dense layer of 512 were enough to eventually crash the laptop.

Tabular Data: After the previous failed attempts, we turned to the tabular data extracted from the segmented images. The VAE was easier and way faster to implement but eventually, not all distributions of the simulated data matched the distributions of the real ones. We even implemented optuna to fine tune it (number of hidden dims, layers, learning rate, dropout rate and batch size), but to no avail. By visual inspection it was clear that it did not work.

By computing the means, it was accurate but the standard deviations were at times an order of magnitude off. We also tried a mixture of Gaussians (BGM) to sample the latent space as well but it was even worse.