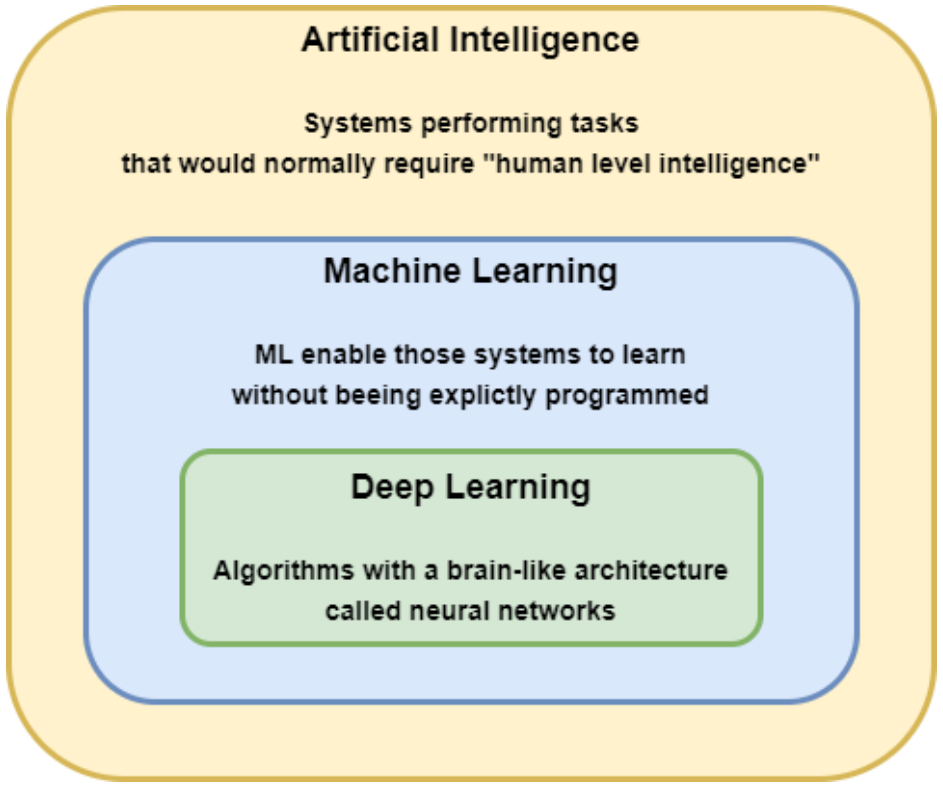
ZEN and APEER - Open Ecosystem for integrated Machine-Learning Workflows

Machine learning

ZEISS

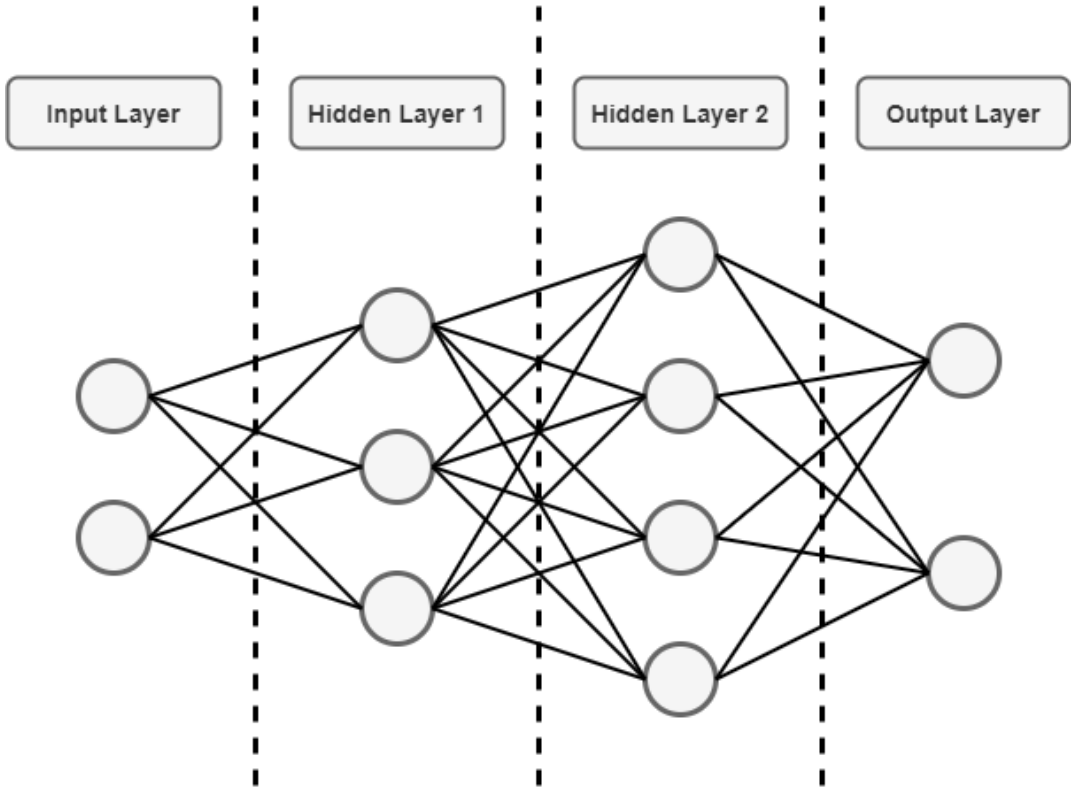
Sebastian Rhode
Product Owner Machine Learning, Technology Center Software, Munich

Artificial Intelligence and Machine Learning and Deep Learning



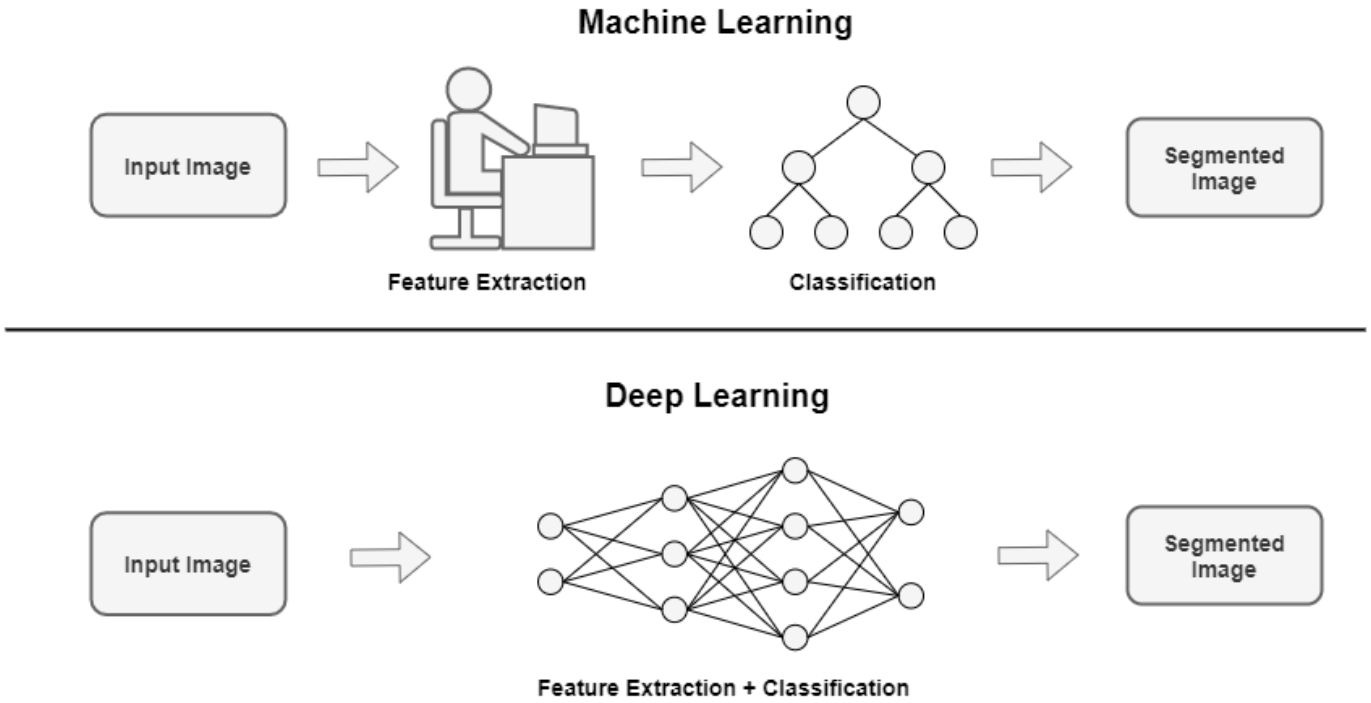
- **Deep Learning** is a special subset of general **Machine Learning**, which is again a subset of **Artificial Intelligence (AI)**.
- Machine learning gives computers the ability to **learn without being programmed** explicitly for a specific task
- Two main categories exist in Machine Learning: Supervised and Unsupervised methods

Deep Learning – What are Artificial Neural Networks?



- Deep Learning itself consists of methods to analyze data using an approach that mimics the way a person would try to make sense of it
- Deep Learning methods use a layered structure of algorithms called artificial neural networks (ANN)
- ANNs have an input and output layer
- Layers in between are hidden layers → their values cannot be observed (easily) during the training
- ANN gets "deeper" if the number of hidden layers increases

Machine Learning – Deep Learning versus Machine Learning
(simplified)



- Machine Learning algorithms are rather" simple" compared to ANNs
- classical Machine Learning **methods are using features typically "engineered"** by human intervention
- Deep Learning **methods extract features automatically** and learn which features work best
- Deep Learning typically requires a lot more data due to the complex algorithms used

Application of Machine Learning methods in Microscopy

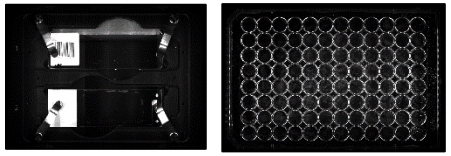
Machine learning

ZEISS

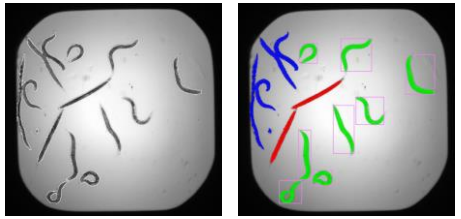
Image Analysis and Processing

Classification

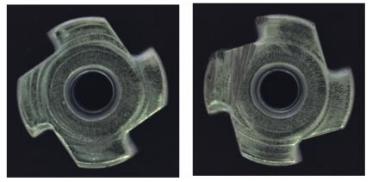
Classifying an entire image or individual objects



Slide vs 96 WellPlate (CD7 or SampleFinder)



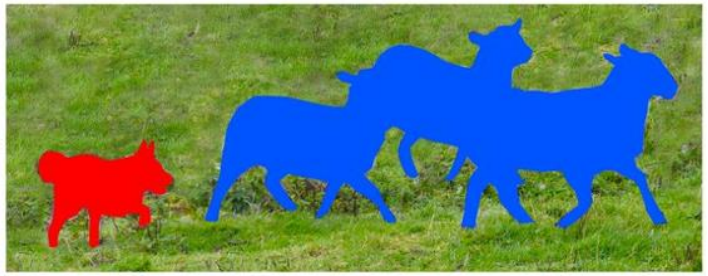
Classify Objects (ZEN Object Classification)



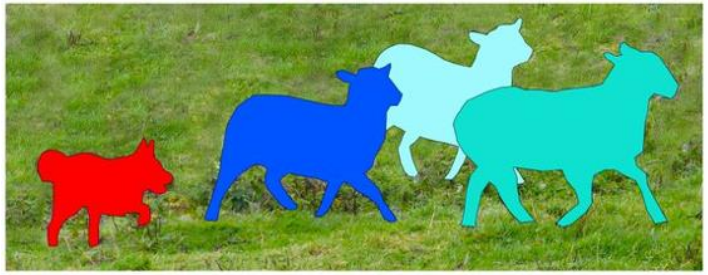
Good part vs anomaly

Segmentation

Refers to classifying at a pixel level



Semantic Segmentation



Instance Segmentation

Processing

Image corrections and enhancements

Denoising

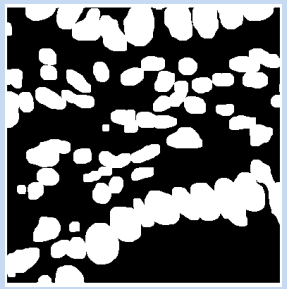
Super-resolution

Machine-Learning - The complexity of the task defines the approach

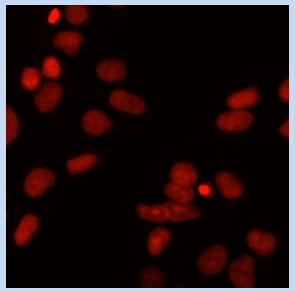
Machine learning

ZEISS

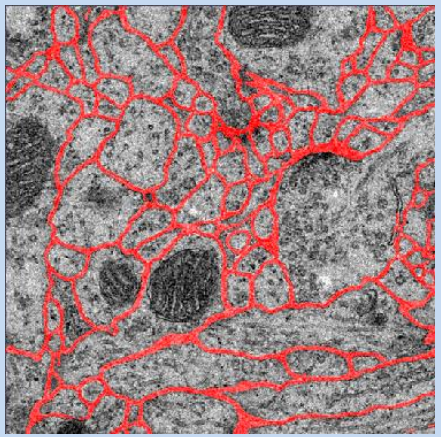
Complexity of the application



Binary segmentation



Nuclear segmentation



Segmenting Membranes

amount of data required

single image

several images (>= 2 images)

>> tens to hundreds (depending on their size)

parameters for training

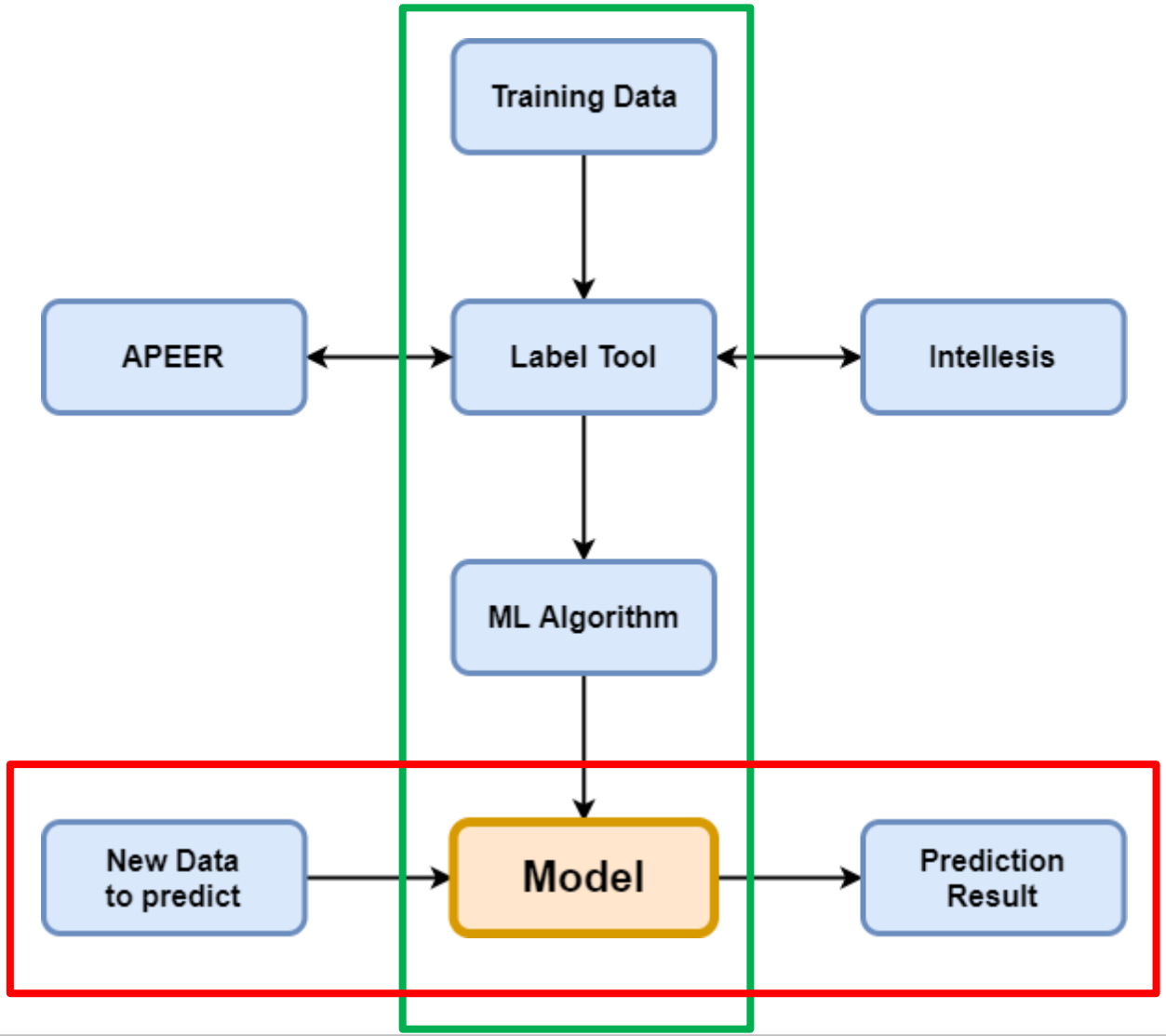
1 | 10 | 100 | 1e+3 | 1e+4 | 1e+5 | 1e+6 | 1e+7

Thresholding

Pixel Classification

Deep Neural Networks

Machine Learning – Train a Model and use a Model



Train a model

Use a model

General Problem Statement

Machine learning

ZEISS

“I can get decent images from anywhere but I am looking for information”

- Potential customer during her visit to Pleasanton

Segmentation and Analysis are today’s biggest problems.

A powerful and reproducible segmentation is the mandatory precursor for any downstream image analysis and measurements.

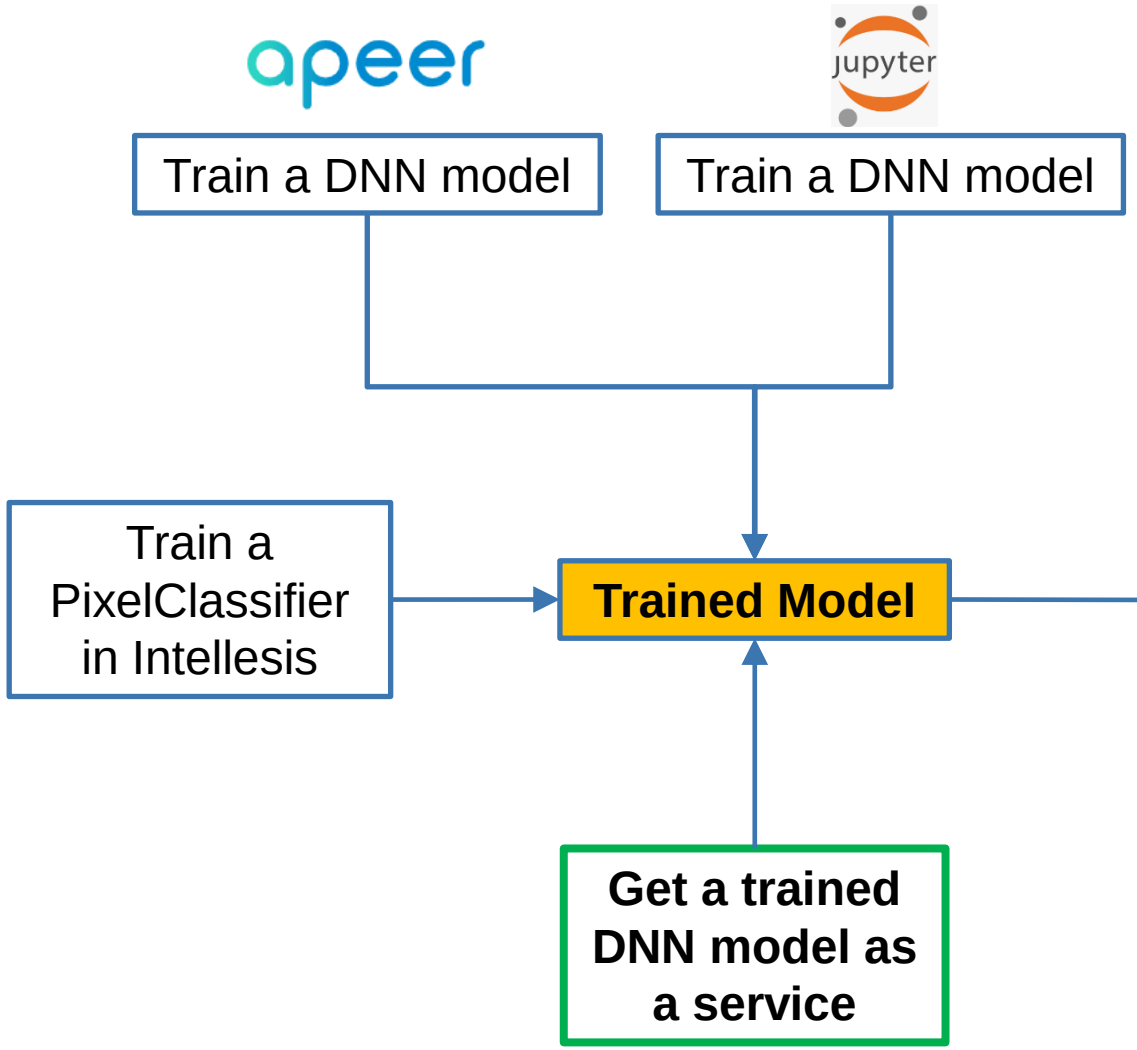
The actual purpose of an image is “to be analyzed”.

- Images can be “hard” or challenging to segment because of their nature.
- Further sorting and classifying segmented objects can be difficult.
- Users are often no experts and need a simple tool.

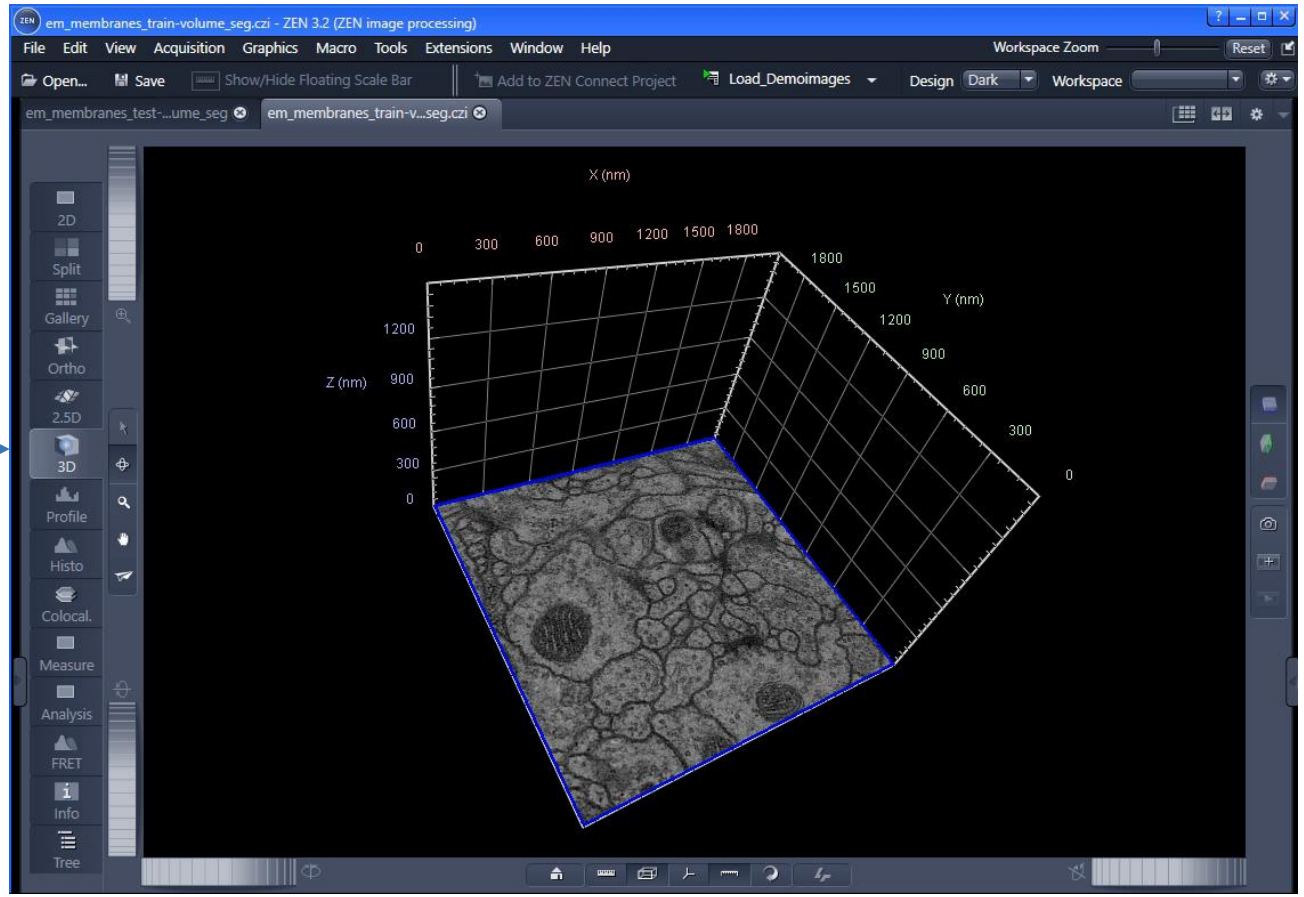
ZEN and APEER – Open Ecosystem for integrated Machine-Learning Workflows

Trained model from Segmentation from “anywhere”

Machine learning

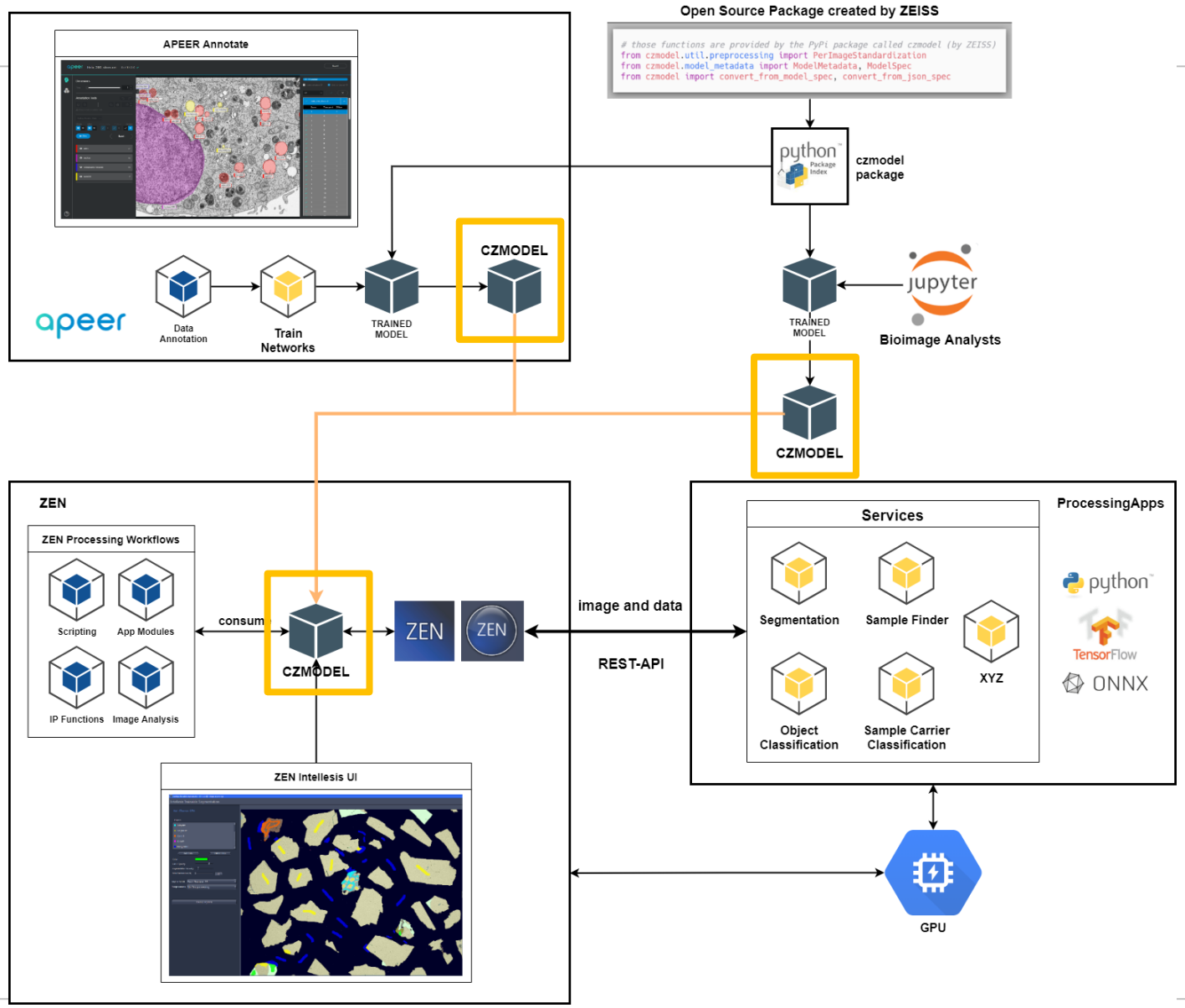


EM Data taken for A.Cardona et.al, PLOS Biology, 2010, ISBI Challenge



ZEISS – Open Ecosystem for integrated Machine-Learning workflows

Machine learning



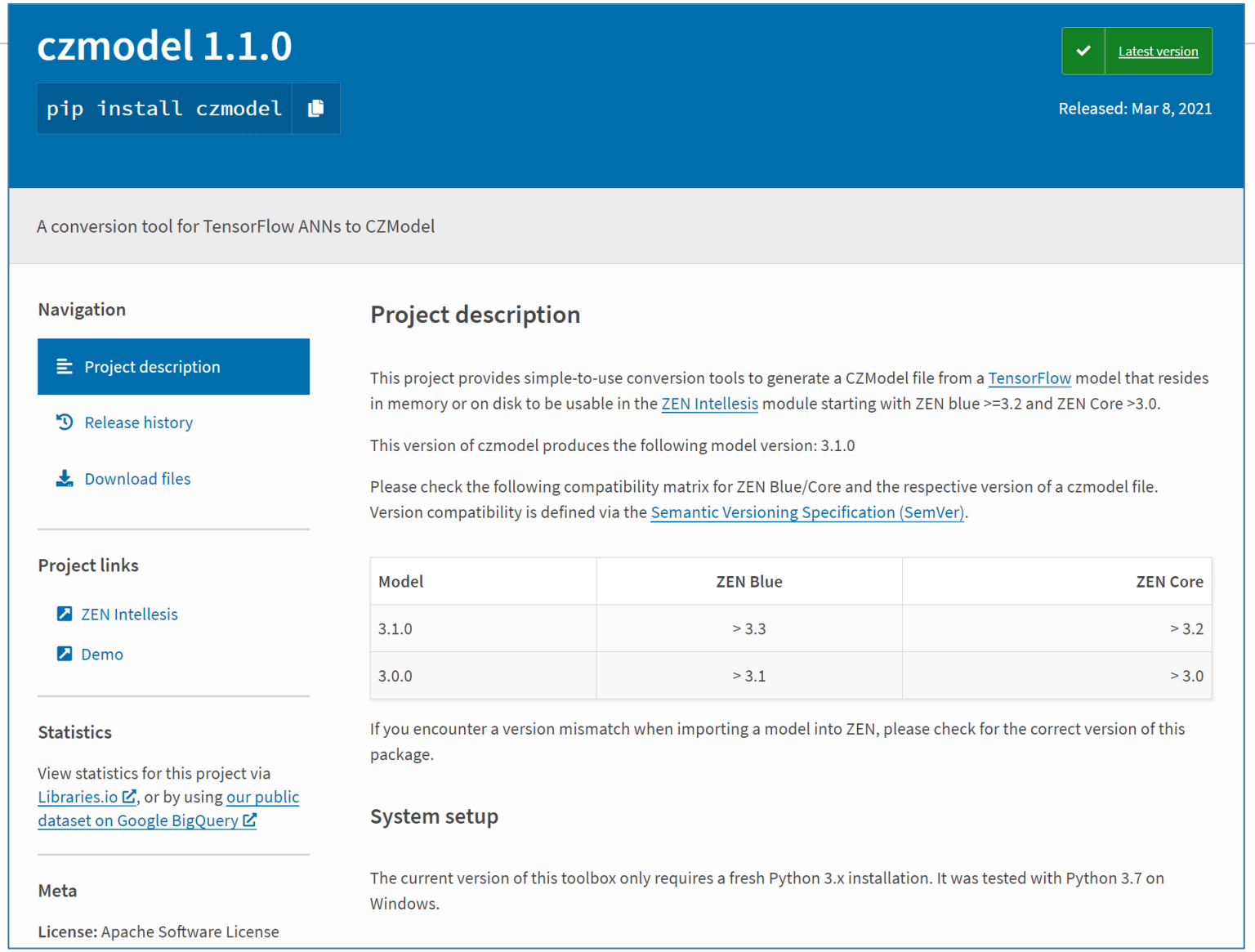
- APEER and ZEN are part of the same ecosystem
  - Benefit from the Annotation and Training tools
  - Use trained neural networks inside APEER workflows
- Creating a model on can be done **by users** or as a **paid service** (SW-SCS)
- **The "exchange" currency inside this ecosystem is the trained model**
- ML models developed on APEER (or anywhere) can be used in integrated workflows
  - Use inside ZEN workflows like Image Analysis, Guided Acquisition, Material Module etc.
  - Benefit for support for large data sets, Tiling Capabilities and Integration

ZEN Intellesis – Open Ecosystem for integrated Machine-Learning Workflows

Use your own TF2 models in ZEN – Open Source PyPi package CZMODEL

Machine learning

ZEISS



czmodel 1.1.0

pip install czmodel

Latest version

Released: Mar 8, 2021

A conversion tool for TensorFlow ANNs to CZModel

Navigation

- Project description
- Release history
- Download files

Project links

- ZEN Intellesis
- Demo

Statistics

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Meta

License: Apache Software License

Project description

This project provides simple-to-use conversion tools to generate a CZModel file from a TensorFlow model that resides in memory or on disk to be usable in the ZEN Intellesis module starting with ZEN blue >=3.2 and ZEN Core >3.0.

This version of czmodel produces the following model version: 3.1.0

Please check the following compatibility matrix for ZEN Blue/Core and the respective version of a czmodel file. Version compatibility is defined via the Semantic Versioning Specification (SemVer).

Model	ZEN Blue	ZEN Core
3.1.0	> 3.3	> 3.2
3.0.0	> 3.1	> 3.0

If you encounter a version mismatch when importing a model into ZEN, please check for the correct version of this package.

System setup

The current version of this toolbox only requires a fresh Python 3.x installation. It was tested with Python 3.7 on Windows.

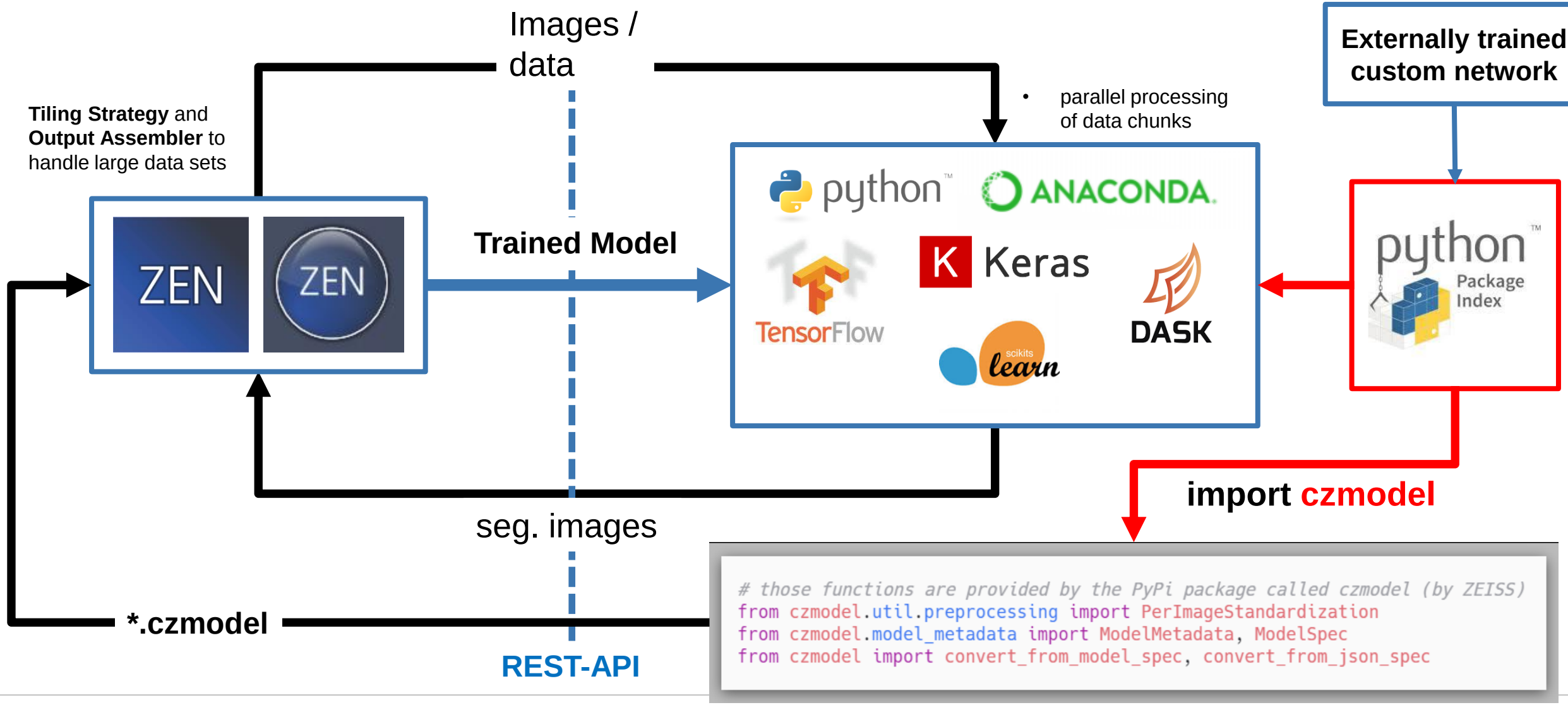
- Open and standardized “container” to store ML models and PyPi
- The CZMODEL is the “glue” or “exchange format” between SW tools for Machine-Learning at ZEISS
- No new model format!
- Support for TF2.SavedModels and ONNX models
- Internally used by ZEN blue, ZEN core, APEER-ML and other parts of ZEISS (even beyond Microscopy)
- Free to use for everybody
- Allows external data scientists to integrate their own models into ZEN workflows

ZEN Intellesis – Open Ecosystem for integrated Machine-Learning Workflows

Use your own TF2 models in ZEN – Open Source PyPi package CZMODEL

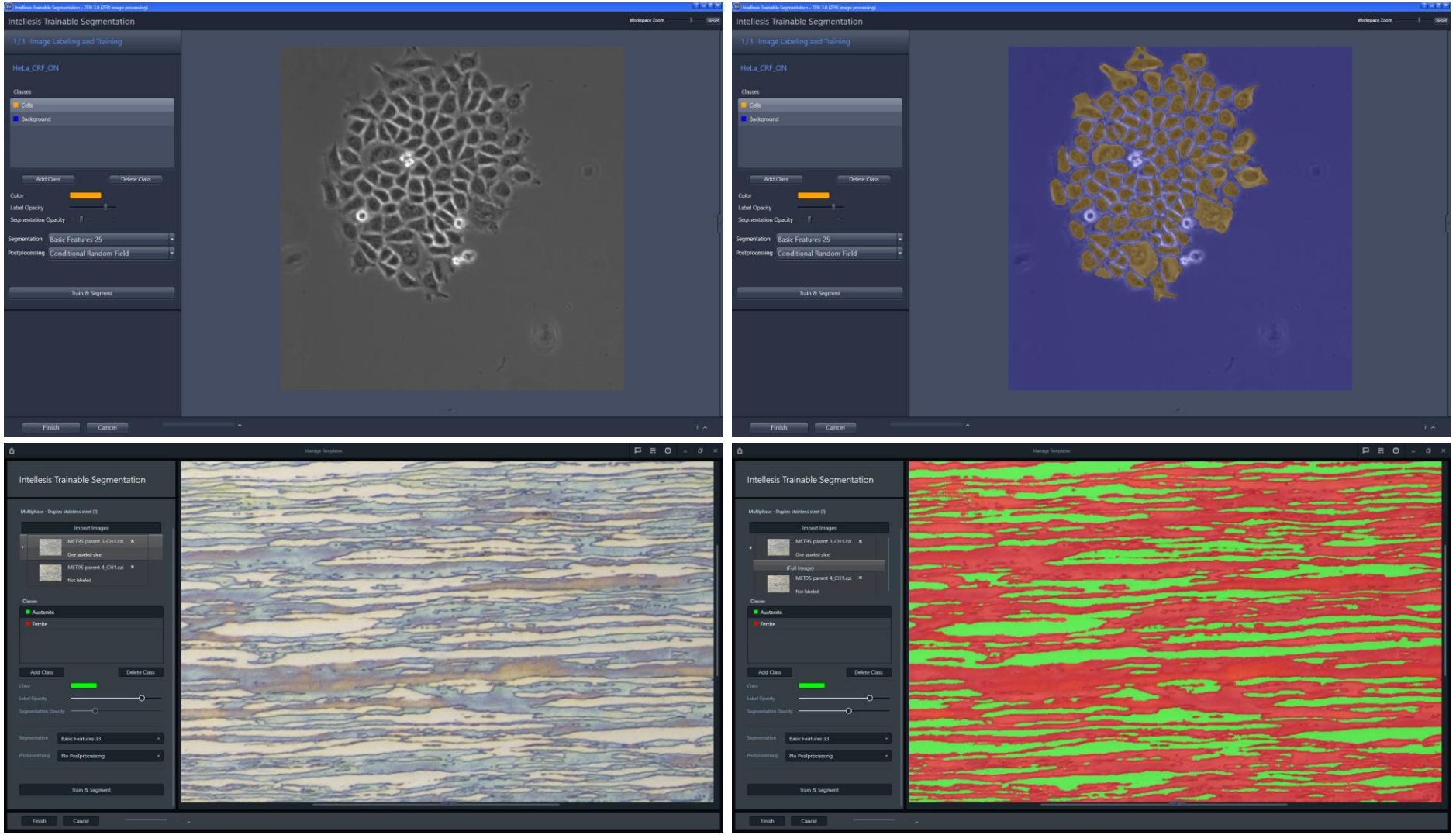
Machine learning

ZEISS



ZEN Intellesis – Open Ecosystem for integrated Machine-Learning Workflows

Machine learning



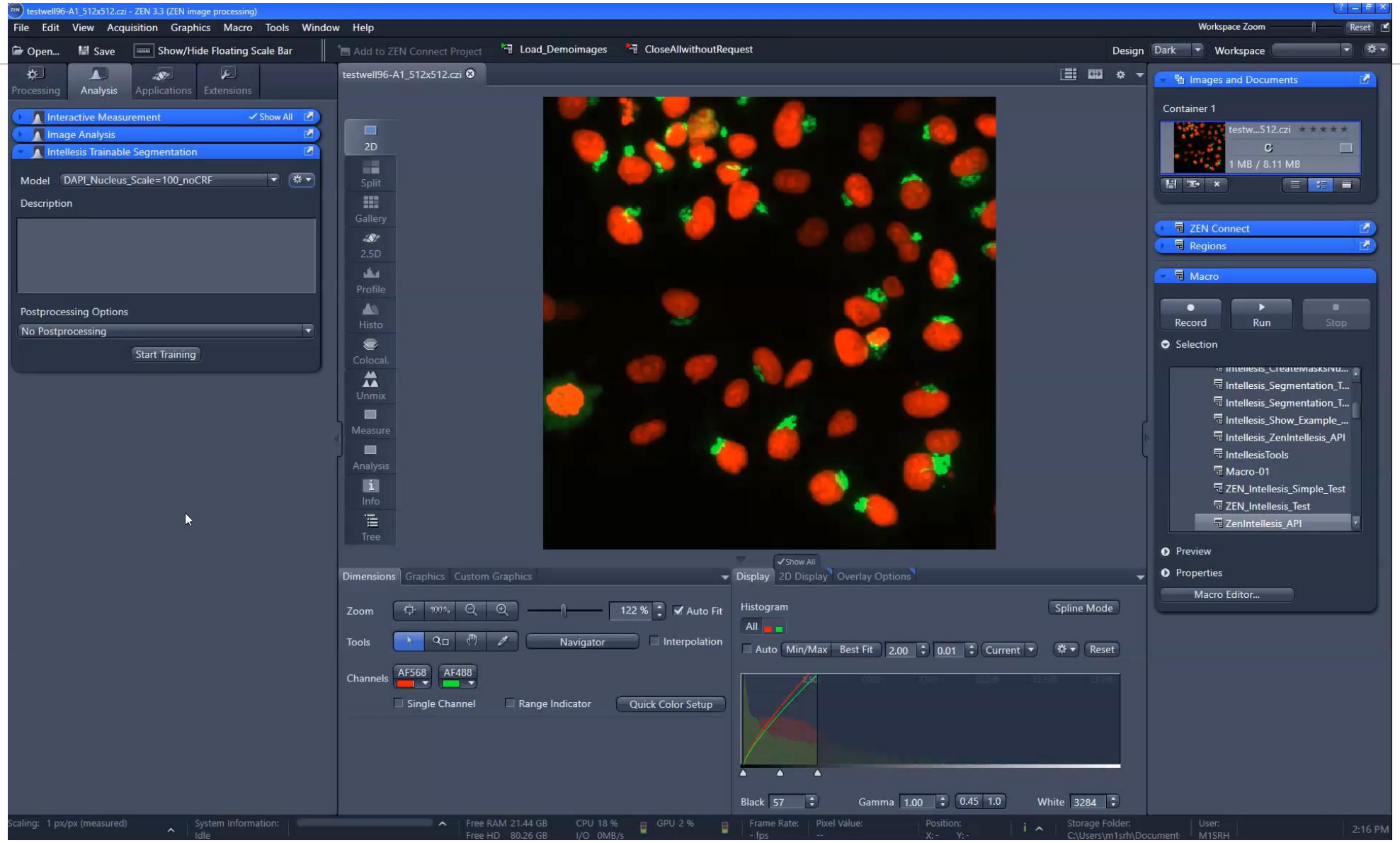
ZEN Intellesis Segmentation
Core Strengths

Machine learning

- **Simple User Interface for Labelling and Training**
  - Label your datasets using a clean and simple UI, train a Pixel Classifier (see: Feature Extractors)
- **Integration into ZEN Measurement and Processing Framework**
  - Use trained models easily inside complete Image Analysis pipelines, Material Modules or inside scripted workflows
- **Open Platform – Import your own trained networks**
  - Import your own model and use it seamlessly integrated in ZEN workflows and benefit from Image Tiling & Fusion strategy (see: Importing external networks into ZEN)
- **Support for Multi-dimensional Datasets**
  - Import any multi-dimensional dataset incl. 3rd party file formats from other vendors

Simple User Interface for Labeling and Training

Machine learning



ZEISS Machine-Learning Ecosystem – Important Components

Machine learning

- Simple and intuitive User Interface to train
  - Segmentation models (APEER and ZEN)
  - Object Classifier (ZEN)
- ZEN Software with Image Analysis Tool, Scripting Interface and pre-defined workflows
- AppManager Infrastructure and ZeissPy distribution integrated in installer
- Tiling Strategy and Image Fusion for nD datasets incl. support for 3rd party data formats
- Open-Source Package *czmodel*: https://pypi.org/project/czmodel/
- Import of externally trained deep-learning models into ZEN platform
- **Support for TensorFlow2 and ONNX models**
- Cloud-based Annotation and Training Platform for neural networks: www.apeer.com
- **Data Annotation and Model Training as a Service**

Tech Notes – ZEN Intellesis Segmentation

Machine learning

ZEISS

- **Multispectral Feature Extraction** – all channels will be used to segment a pixel
- **Class Segmentation** – hierarchical structures with independent segmentation per object class
- **Engineered Feature Sets** and **Deep Feature Extraction** and **pre-trained networks**
- **Random Forrest Classifier + DNN via TF.SaveModels**
- **Public DNN Model Specification on ZEISS Microscopy GitHub page**
- Post processing: **Conditional Random Fields** and **confidence thresholds**
- IP-Functions for **Creating Masks** and **Scripting Integration**
- Client-Server Architecture (REST-API) with **Client-side Tiling Strategy**

```
def runmodel(image, model,
             use_confidence=False,
             confidence_threshold=0,
             format='MultiChannel',
             extractclass=False,
             class2extract_id=0,
             addseg=False,
             adapt_pixeltype=True):

    # define the desired output format
    if format == 'MultiChannel':
        # output image will have one channel per model class
        segf = ZenSegmentationFormat.MultiChannel
    if format == 'Labels':
        # output image will have one channel with distinct labels per model class
        segf = ZenSegmentationFormat.Labels

    # classify pixels using a trained model incl import deep-learning models
    if use_confidence:
        try:
            # run the segmentation and apply confidence threshold to segmented image
            outputs = Zen.Processing.Segmentation.TrainableSegmentationWithProbabilityMap(image, model, segf)
            seg_image = outputs[0]
            conf_map = outputs[1]
            print('Apply Confidence Threshold to segmented image.')
            seg_image = Zen.Processing.Segmentation.MinimumConfidence(seg_image, conf_map, confidence_threshold)
            conf_map.Close()
            del outputs
        except ApplicationException as e:
            seg_image = None
            print('Application Exception : ', e.Message)

    if not use_confidence:
        try:
            # run just the segmentation
            seg_image = Zen.Processing.Segmentation.TrainableSegmentation(image, model, segf)
        except ApplicationException as e:
            seg_image = None
            print('Application Exception : ', e.Message)

    return seg_image
```

Tips - Labelling and Training

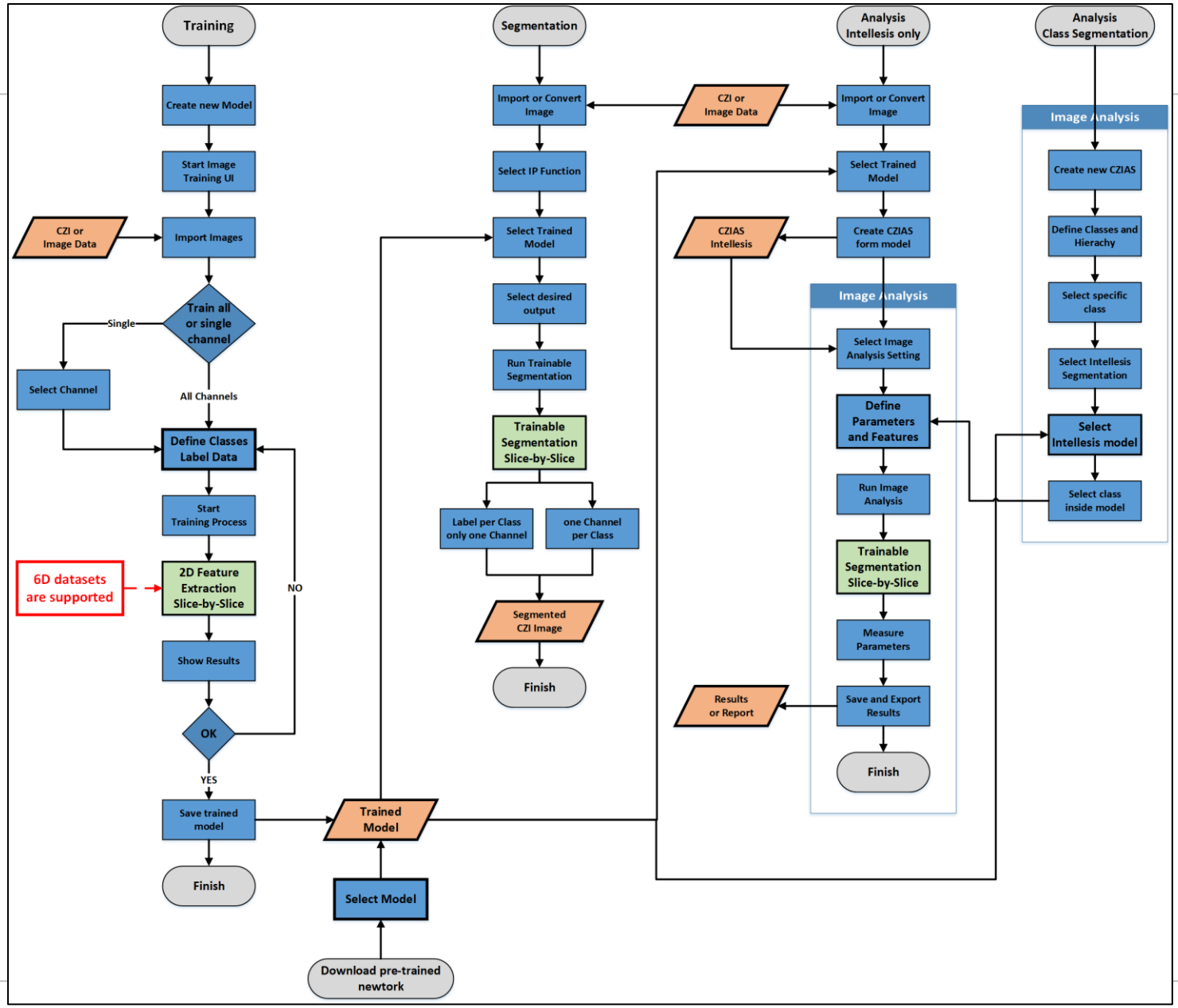
Machine learning

ZEISS

- **Labels is valid for all channels of an image** and the information from all channels will be used to classify that pixel (X,Y)
- **Label as few pixels as possible** and rather do more iterations of training.
- Sometimes it is helpful to **use more classes in order to get better results for each class**.
- **Labeling large homogenous areas with many pixels will slow down the training** but give not necessarily giver better results.
- It is **good practice to label “roughly” the same number of pixels for every class**, e.g., try to avoid labeling much, much more pixels for one class compared to another class
- The **default feature extractors are using the fewest number of parameters** (start here first) while the Deep Feature extractors use the largest number. Use those only when the Default one do not work well.
- If those methods do not work, **consider training a Deep Neural Network on APEER or consider getting a model as a service**.

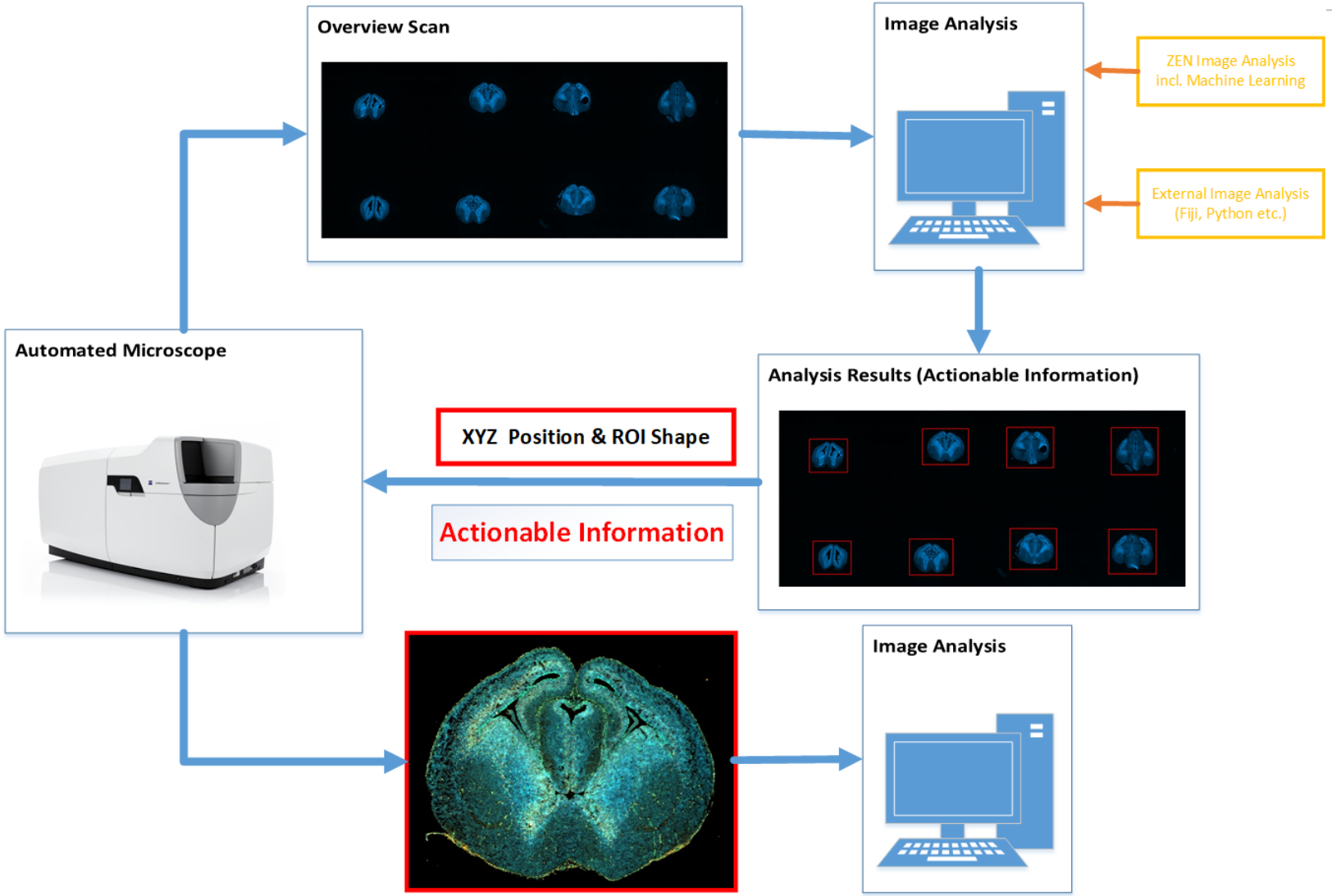
Intellesis Segmentation - Workflows and Measurement Integration

Machine learning



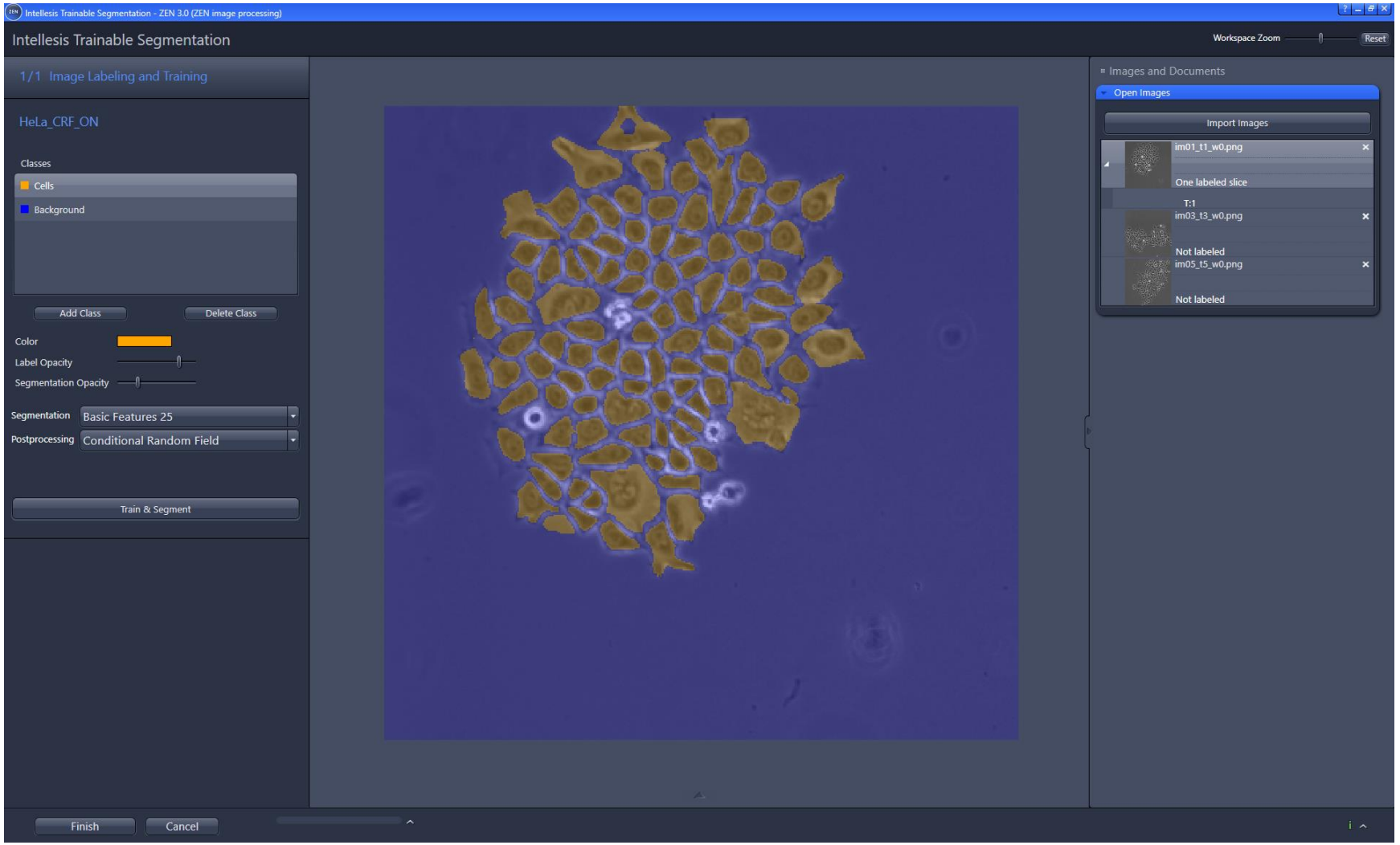
Actionable Information – Where can trained models be used

Machine learning



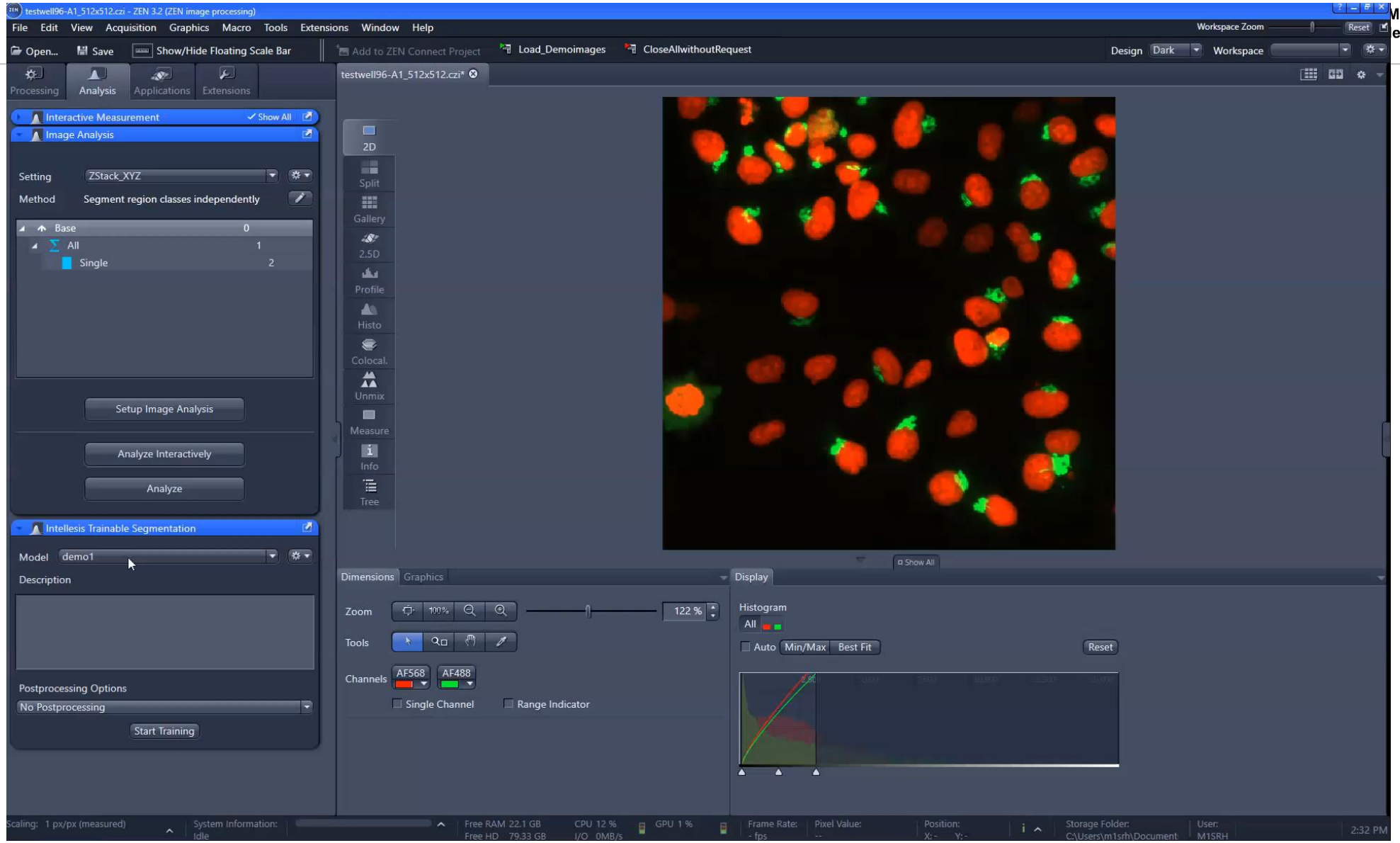
Training UI in ZEN blue

Machine learning



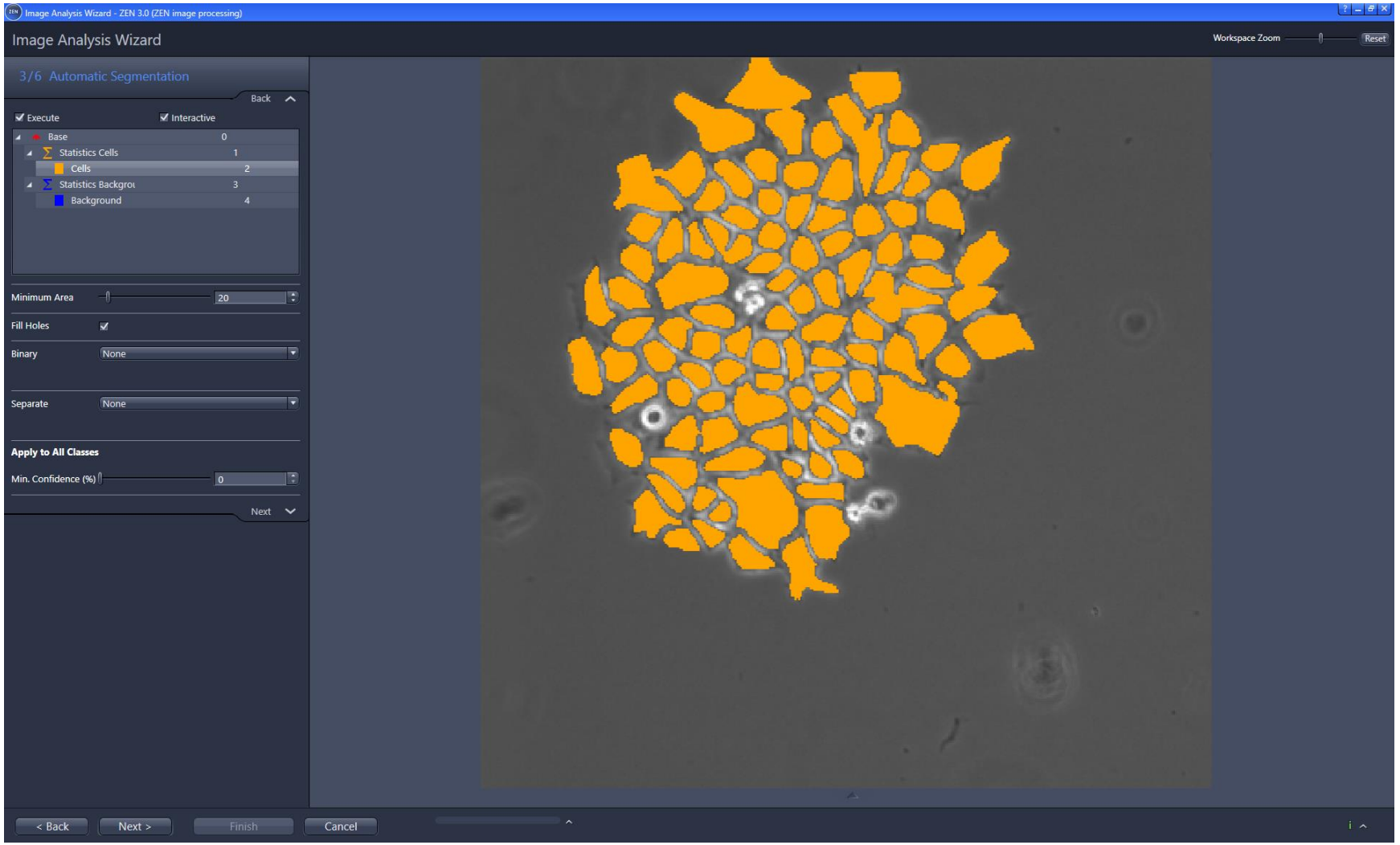
Integration into ZEN Image Analysis and Software Modules

Machine Learning



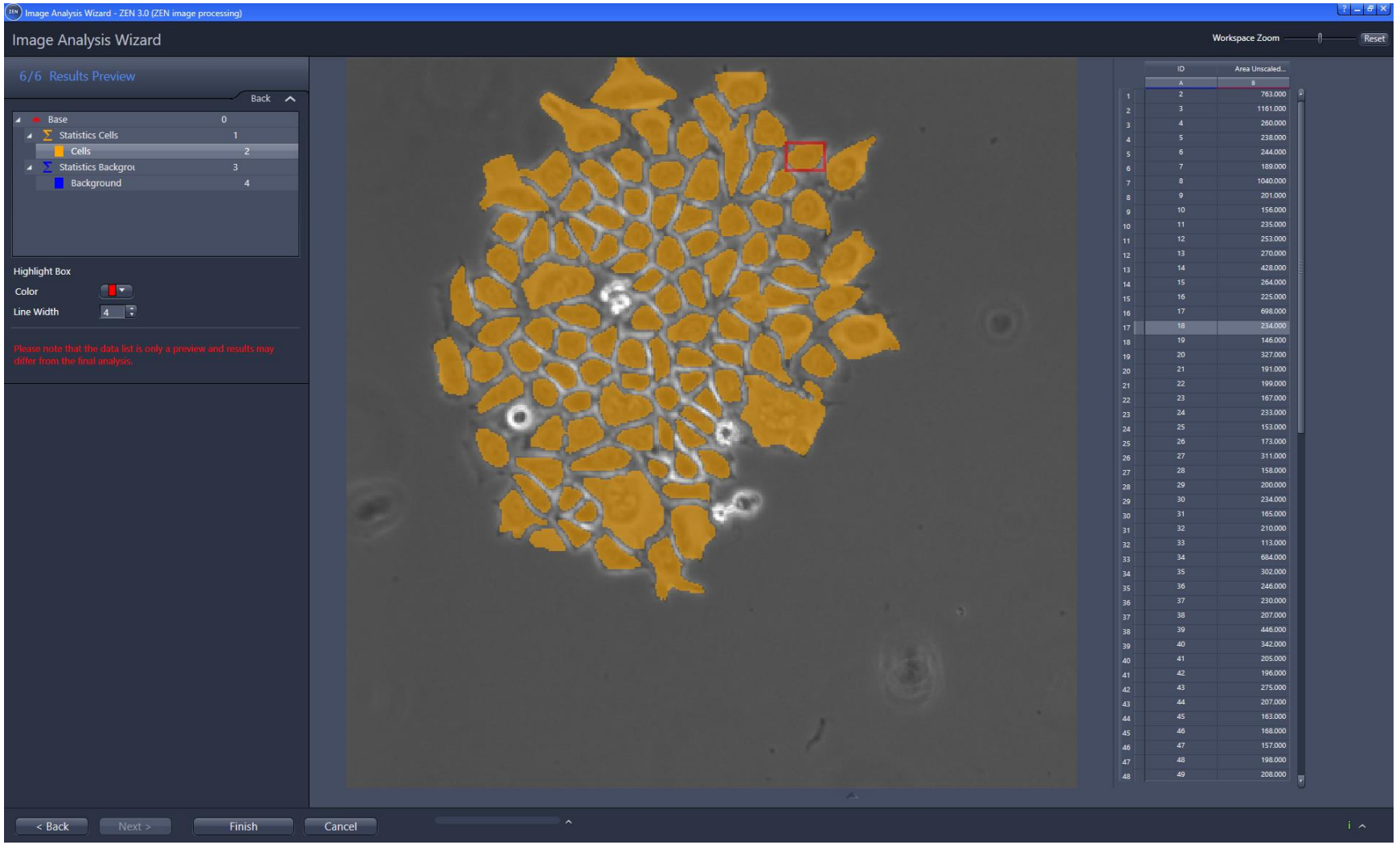
Integration into Image Analysis

Machine learning



Integration into Image Analysis

Machine learning



Feature Extractors – General Remarks Information

Machine learning

- Intellesis Segmentation allows to use 7 different feature extractors to create the feature vector that will be used for the subsequent Pixel Classification using Random Forrest.
- For calculating the features of one specific pixel, various filters with various filter sizes and parameters are applied to the region around this pixel.
- All the filter results are concatenated yielding the final feature vector describing the pixel. Considering the filter sizes of all filters we obtain a dimensionality of 25/33 for the Basic Features 25/33
- For Deep Feature Extraction ZEN Intellesis uses the results of an intermediate layer of the VGG19 network as input features for the Pixel Classification using a Random Forest Classifier.

For more detailed Information:

Intellesis Segmentation - Feature Extraction for Pixel Classification

Remark: Do not confuse Intellesis Segmentation via PixelClassification with using pre-trained deep neural networks, which is also possible with Intellesis Segmentation.

Random Forest Classifier

Machine learning

ZEISS

- Each decision tree in the forest considers a subset of features when forming “questions” and only has access to a subset set of the training data
- This increases diversity in the forest leading to more robust overall predictions and the name ‘**Random Forest**.’
- Prediction = average of all the individual decision tree estimates
- Regression task → predicting a continuous value of …
- **Classification task → discrete class labels such as A or B**
    - Random Forest will take a majority vote for the predicted class

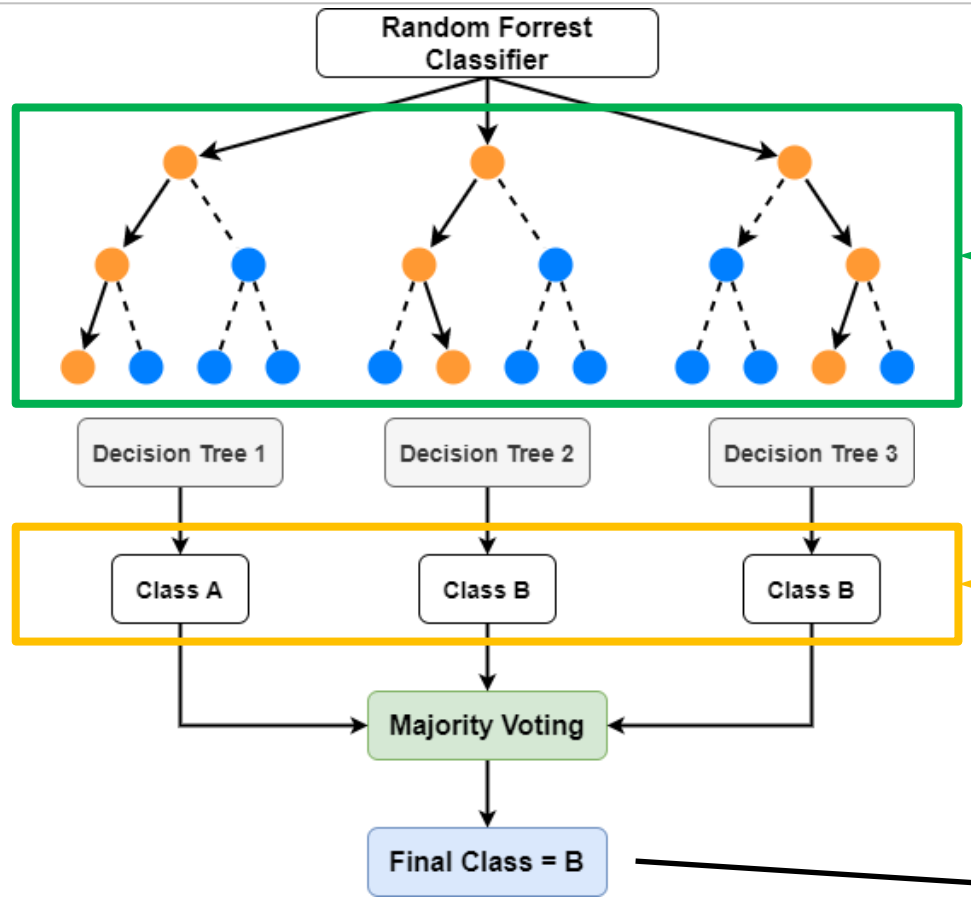
https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

scikit-learn - Random Forest

Wikipedia - Random Forrest

Random Forest Classifier – Decision Trees

Machine learning

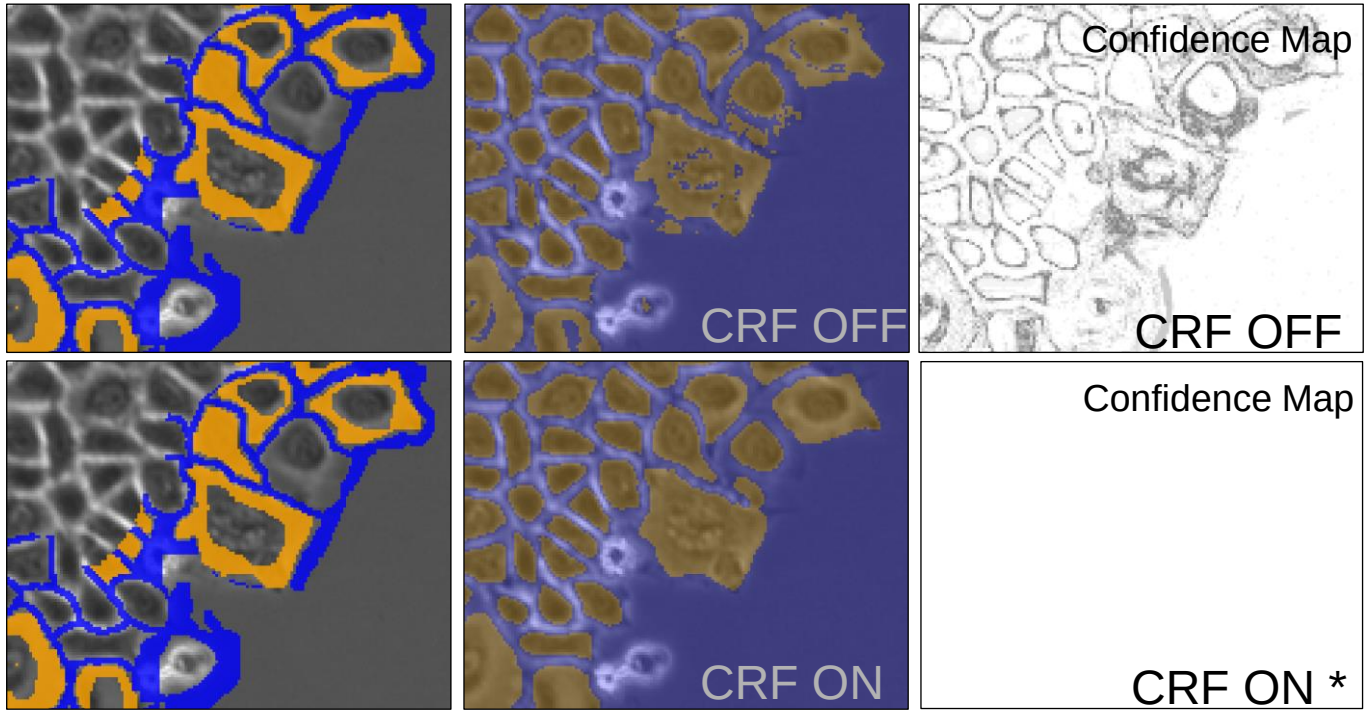


subset of features and different decision trees

2 out of 3 votes = 66% confidence

Pixel will be assigned class B

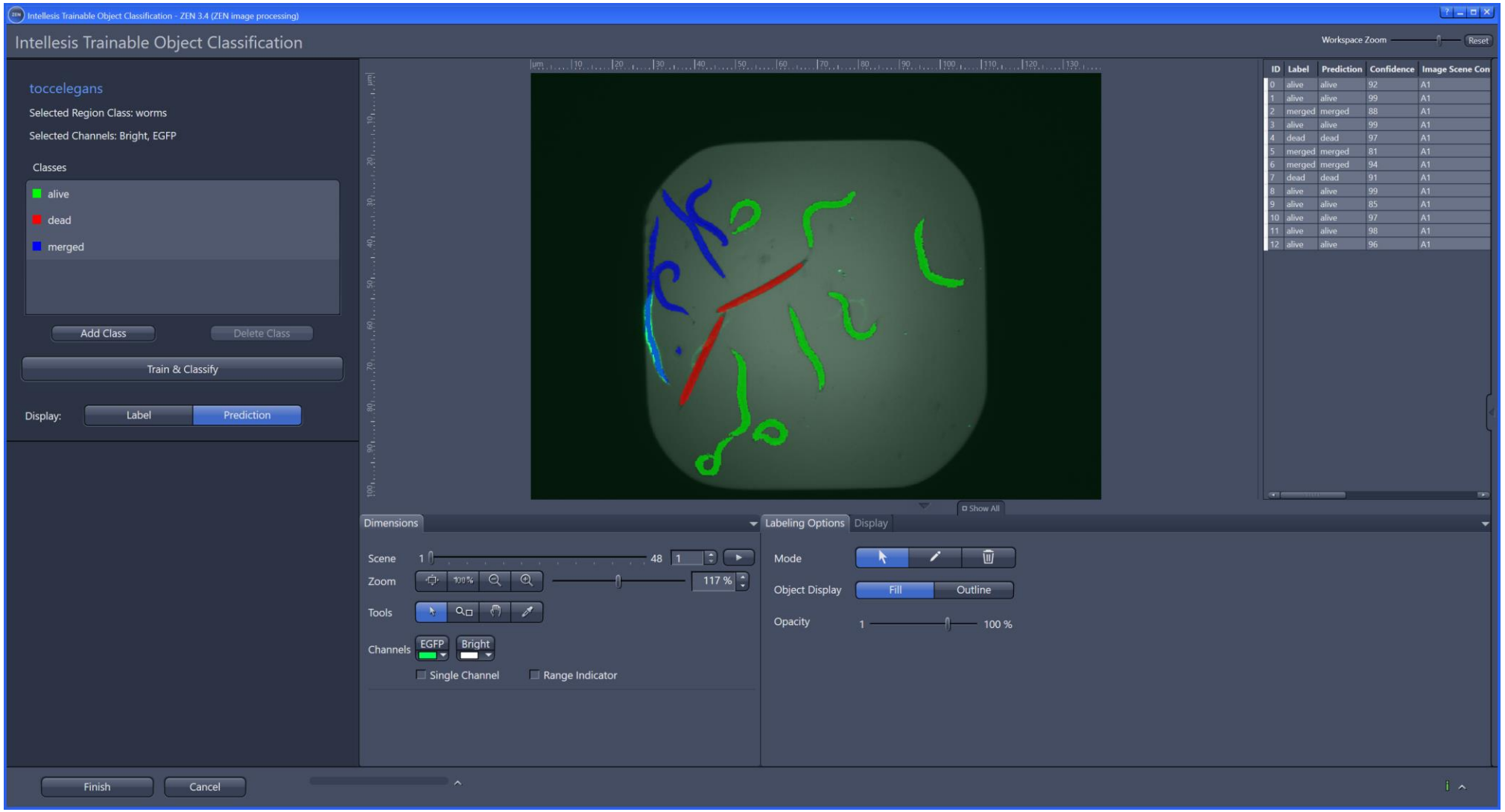
Postprocessing – Conditional Random Fields (CRF)



- CRF tries to create smooth and shaper edges by re-classifying pixels based on confidence levels in their neighborhood
- after CRF there is no real probability map from the Random Forrest anymore, all confidence values will be set to 100%
- For more details: [PostProcessing - Conditional Random Fields]

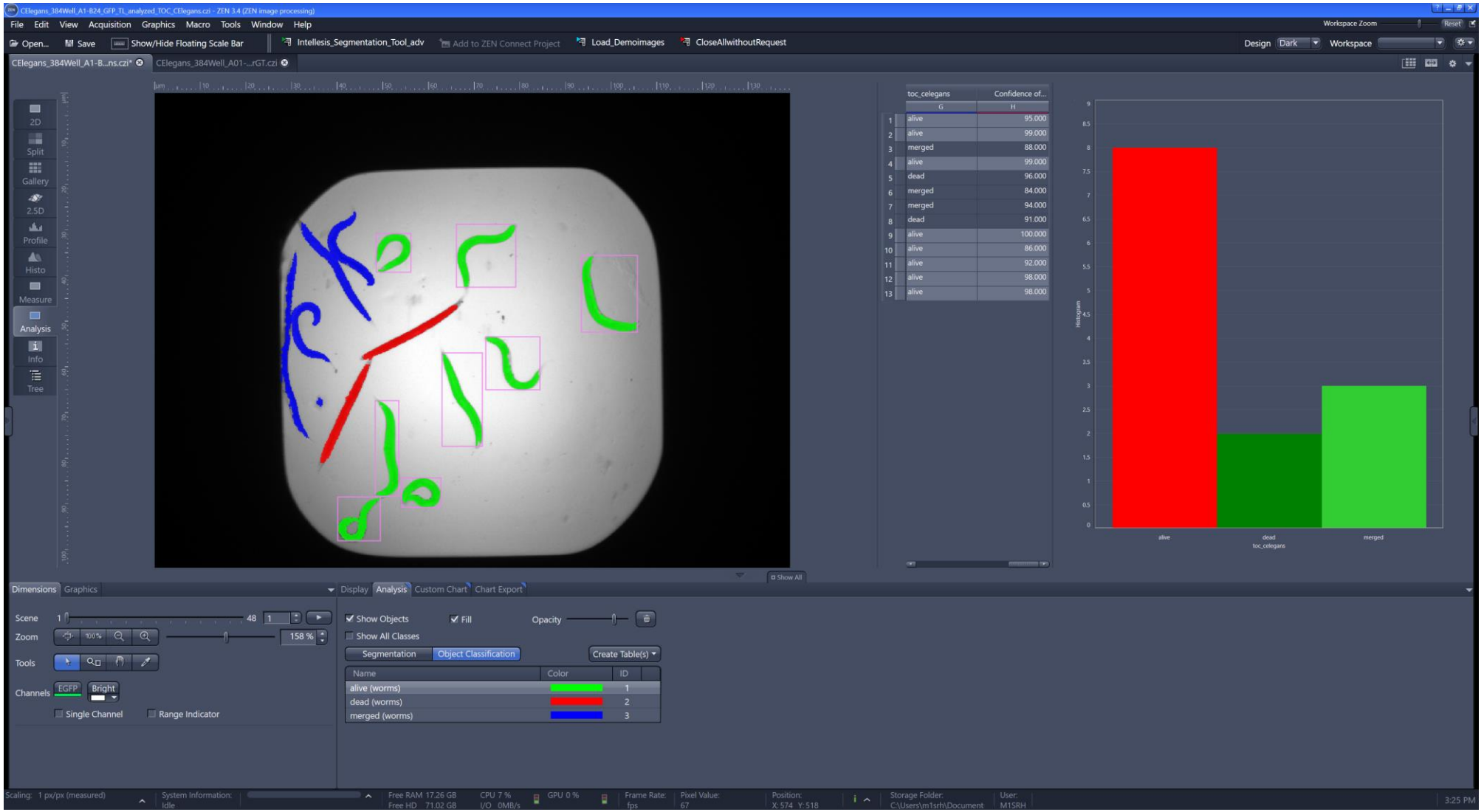
ZEN Intellesis Object Classification – Training UI

Machine learning



ZEN Intellesis Object Classification – Result Visualization

Machine learning



ZEN Intellesis Object Classification
Core Strengths

Machine learning

- **Simple User Interface for Labelling Objects and Train a Classifier**
  - Label the segmented objects using a clean and simple UI and train an Object Classifier using already analyzed images
- **Integration into ZEN Measurement and Processing Framework**
  - Use trained Object Classifier inside complete Image Analysis pipelines, Material Modules or inside scripted workflows
- **Support for Multi-dimensional Datasets**
  - Import any multi-dimensional dataset incl. 3rd party file formats from other vendors

Tech Notes – ZEN Intellesis Object Classification

Machine learning

- **Requires an analyzed image** with already segmented objects
  - segmentation method can be "anything"
  - Thresholds, PixelClassifier, DeepNeuralNetwork
- **Multispectral Feature Extraction** – all channels can be used to extract intensity-based features
- **Geometrical and Shape Features** – all available features from the Image Analysis will be used automatically
- **Random Forrest Classifier** to process the feature table
- IP-Functions for Classification and **Scripting Integration** for **Automation**

```python
# get all available object classification models
ocmodels = ZenIntellesis.ObjectClassification.ListAvailableModels()

for obcmodel in ocmodels:
    print('TOC Model Name:', obcmodel.Name)
    print('Description   :', obcmodel.Description)
    print('Status        :', obcmodel.Status

    # get training images
    training_images = obcmodel.TrainingImages
    for ti in training_images:
        print(ti)

# select a specific model
myobjmodel = find_objclassmodel('my_toc_model')
print('Use model:', myobjmodel.Name)

# import a model
model2import = r"C:\Object_Classification_Models\model1.cztoc"

# classify an analyzed image (inplace)
myobjmodel.Classify(image, appendFeatures=False) # Where is append feature option

# define IA class of interest
IAclass_all = 'Statistics objectA'
IAclass_single = 'objectA'

# define the object class of interest
objclass = 'alive'

# export table for ALL objects (statistics)
table_all = Zen.Analyzing.CreateRegionsTable(image, IAclass_all)
Zen.Application.Documents.Add(table_all)

# export table for SINGLE objects (the ones classified by the model)
table_single = Zen.Analyzing.CreateRegionTable(image, IAclass_single)

# extract a specific object class for from table by filtering (in place)
table_single.RemoveRows(lambda row: row[myobjmodel.Name] ≠ objclass)
```

ZEN Intellesis Object Classification - Things to understand

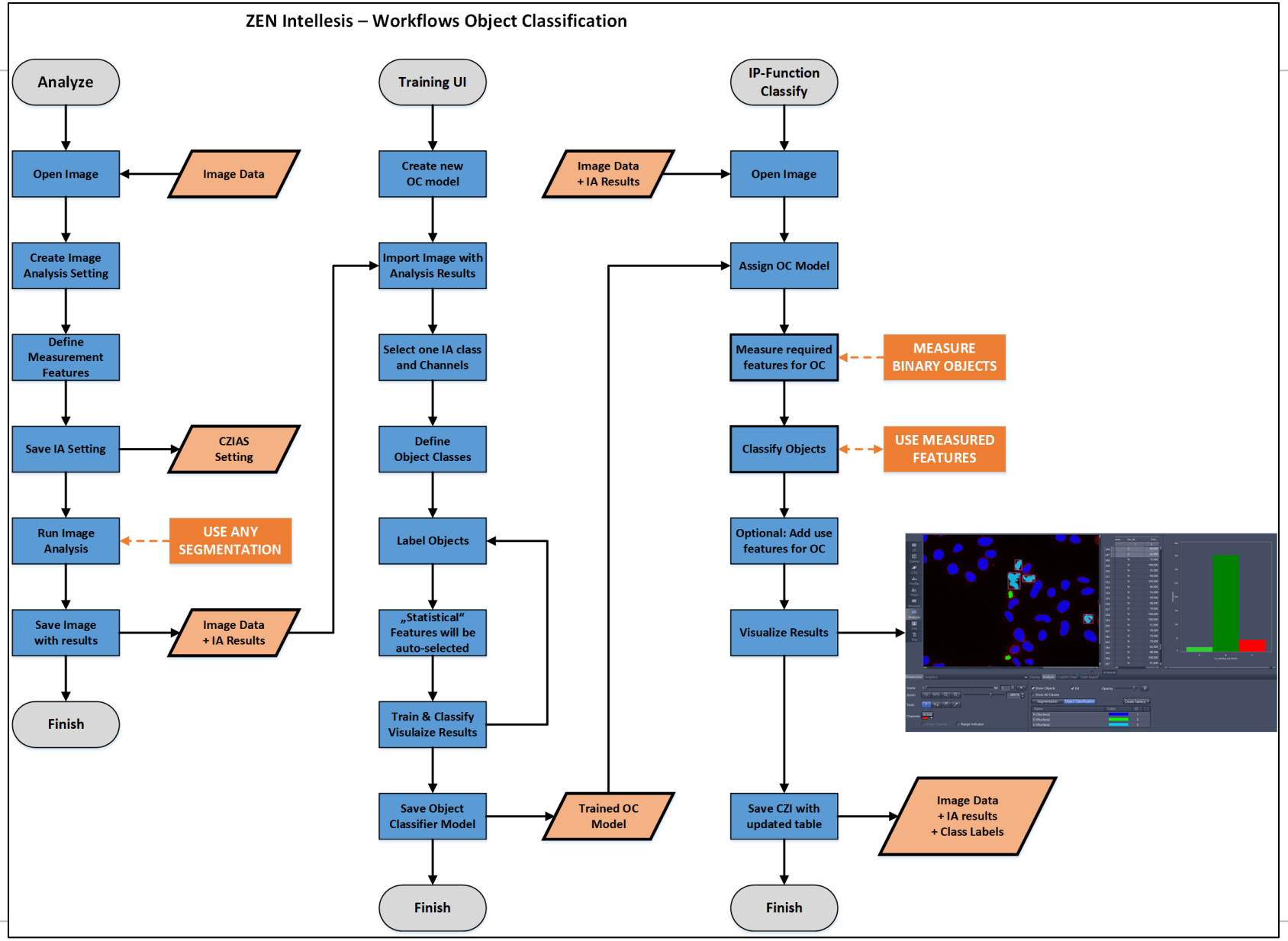
Machine learning

ZEISS

- ZEN Intellesis Object Classification is **a dedicated SW module** for object classification, and it is independent from ZEN Intellesis Segmentation.
- Its task is to **classify (already segmented) objects based on measured features**, not to measure parameters.
- Object Classification **uses intensity-based features** and **all** available **geometrical and shape-based features**.
- In other words, **object classification "sorts" objects** of one specific region class from image analysis (cells) into different type of objects (cell type A and type B).
- It **requires analyzed images (!) for training and for classification** and can be used after ZEN Image Analysis (not within) – it a subsequent workflow step.
- The **type of segmentation does not matter** – any methods (classical or machine-learning) can be used to create the segmented objects.
- The **input** for the classifier is the **table with measured features**, not the actual image itself.

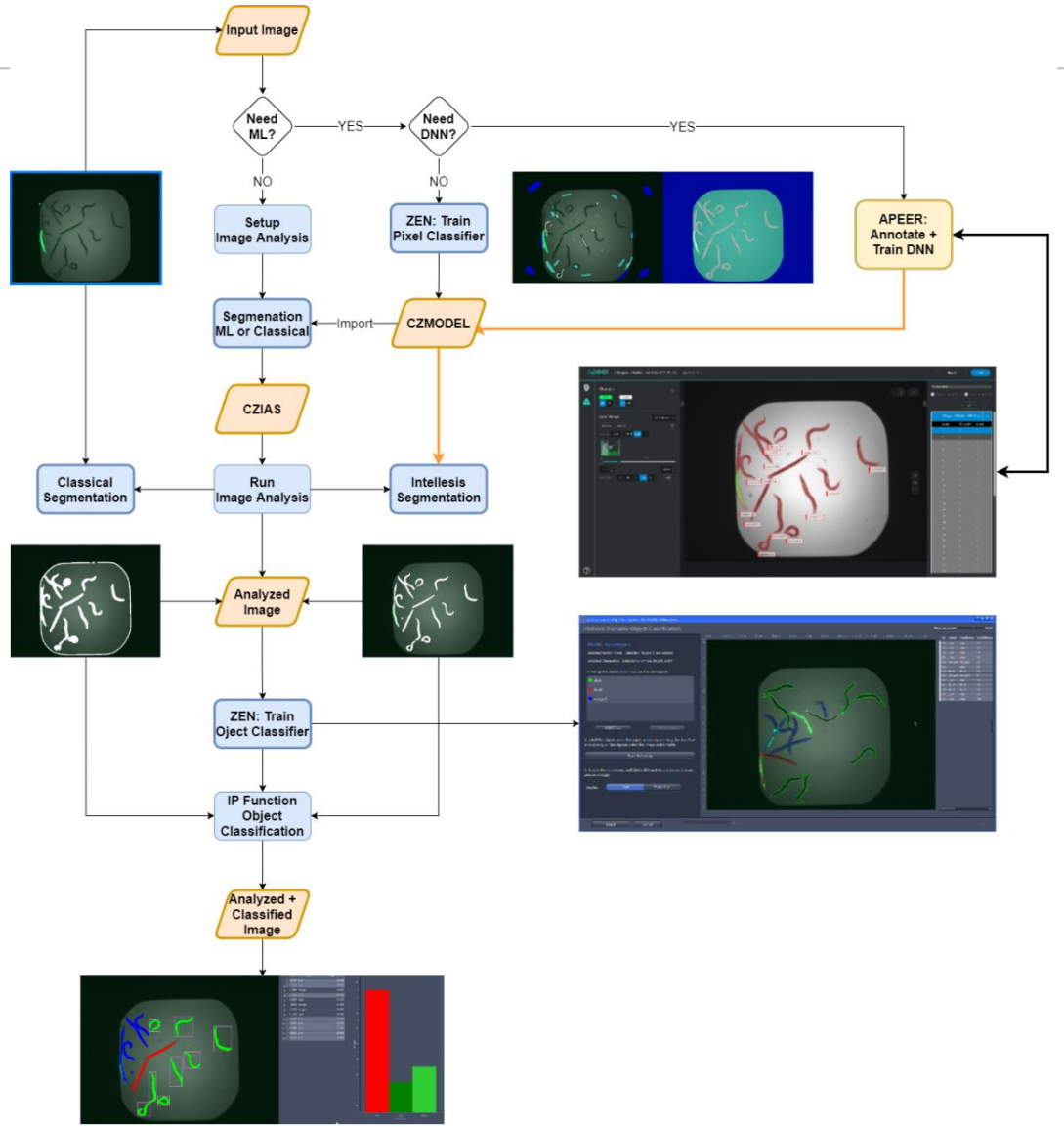
Intellesis Object Classification - Workflows and Measurement Integration

Machine learning



ZEN Intellesis Segmentation and Classification Workflow

Machine learning



ZEN Machine-Learning – Full integration into Python Scripting Interface

Machine learning

```python
def runmodel(image, model,
             use_confidence=False,
             confidence_threshold=0,
             format='MultiChannel',
             extractclass=False,
             class2extract_id=0,
             addseg=False,
             adapt_pixeltype=True):

    # define the desired output format
    if format == 'MultiChannel':
        # output image will have one channel per model class
        segf = ZenSegmentationFormat.MultiChannel
    if format == 'Labels':
        # output image will have one channel with distinct labels per model class
        segf = ZenSegmentationFormat.Labels

    # classify pixels using a trained model incl import deep-learning models
    if use_confidence:
        try:
            # run the segmentation and apply confidence threshold to segmented image
            outputs = Zen.Processing.Segmentation.TrainableSegmentationWithProbabilityMap(image, model, segf)
            seg_image = outputs[0]
            conf_map = outputs[1]
            print('Apply Confidence Threshold to segmented image.')
            seg_image = Zen.Processing.Segmentation.MinimumConfidence(seg_image, conf_map, confidence_threshold)
            conf_map.Close()
            del outputs
        except ApplicationException as e:
            seg_image = None
            print('Application Exception : ', e.Message)

    if not use_confidence:
        try:
            # run just the segmentation
            seg_image = Zen.Processing.Segmentation.TrainableSegmentation(image, model, segf)
        except ApplicationException as e:
            seg_image = None
            print('Application Exception : ', e.Message)

    return seg_image
```

```python
# get all available object classification models
ocmodels = ZenIntellesis.ObjectClassification.ListAvailableModels()

for obcmodel in ocmodels:
    print('TOC Model Name:', obcmodel.Name)
    print('Description   :', obcmodel.Description)
    print('Status        :', obcmodel.Status

    # get training images
    training_images = obcmodel.TrainingImages
    for ti in training_images:
        print(ti)

# select a specific model
myobjmodel = find_objclassmodel('my_toc_model')
print('Use model:', myobjmodel.Name)

# import a model
model2import = r"C:\Object_Classification_Models\model1.cztoc"

# classify an analyzed image (inplace)
myobjmodel.Classify(image, appendFeatures=False) # Where is append feature option

# define IA class of interest
IAclass_all = 'Statistics objectA'
IAclass_single = 'objectA'

# define the object class of interest
objclass = 'alive'

# export table for ALL objects (statistics)
table_all = Zen.Analyzing.CreateRegionsTable(image, IAclass_all)
Zen.Application.Documents.Add(table_all)

# export table for SINGLE objects (the ones classified by the model)
table_single = Zen.Analyzing.CreateRegionTable(image, IAclass_single)

# extract a specific object class for from table by filtering (in place)
table_single.RemoveRows(lambda row: row[myobjmodel.Name] ≠ objclass)
```

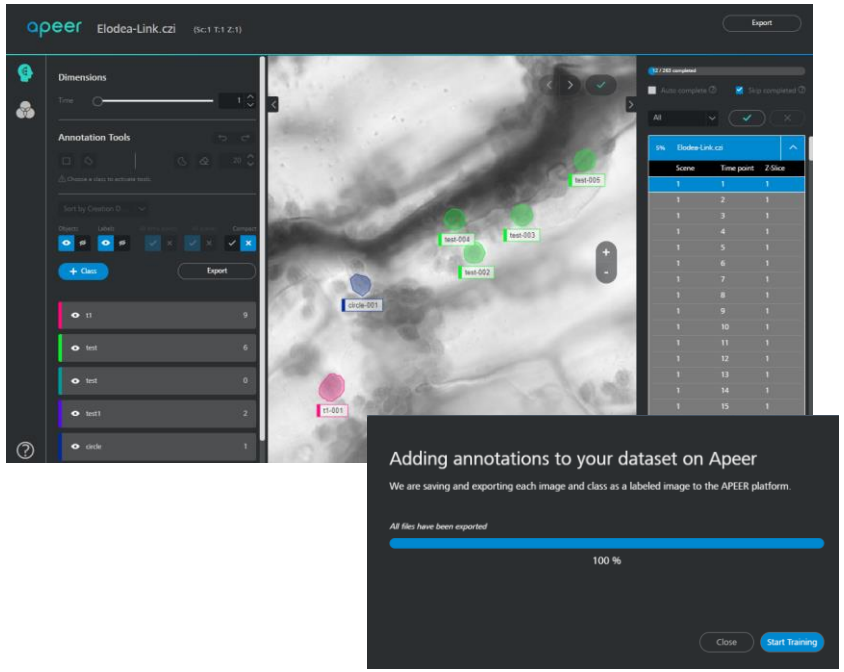
Clearly defined steps to guide the user from annotation to training to segmentation

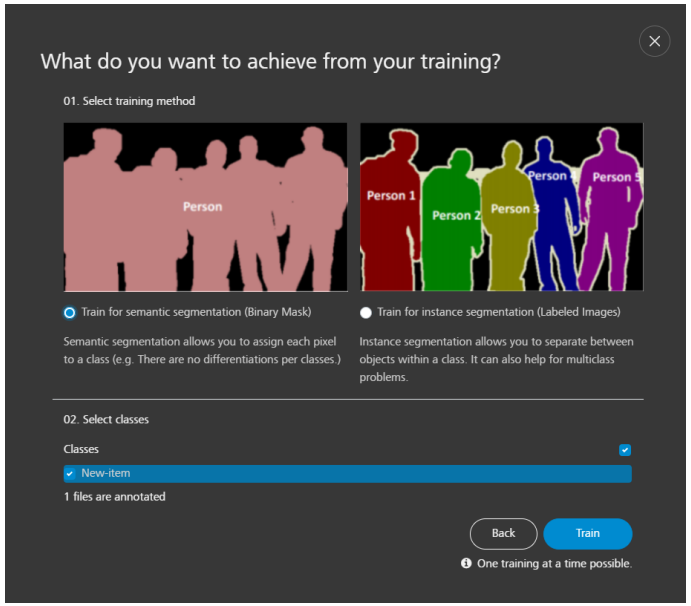
Machine learning

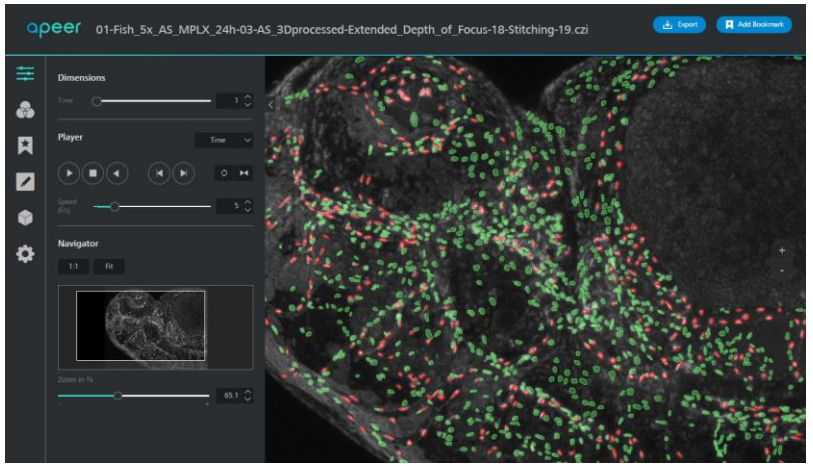
Annotate on APEER

Train your specific U-net

Segment other images – on APEER or in Intellesis



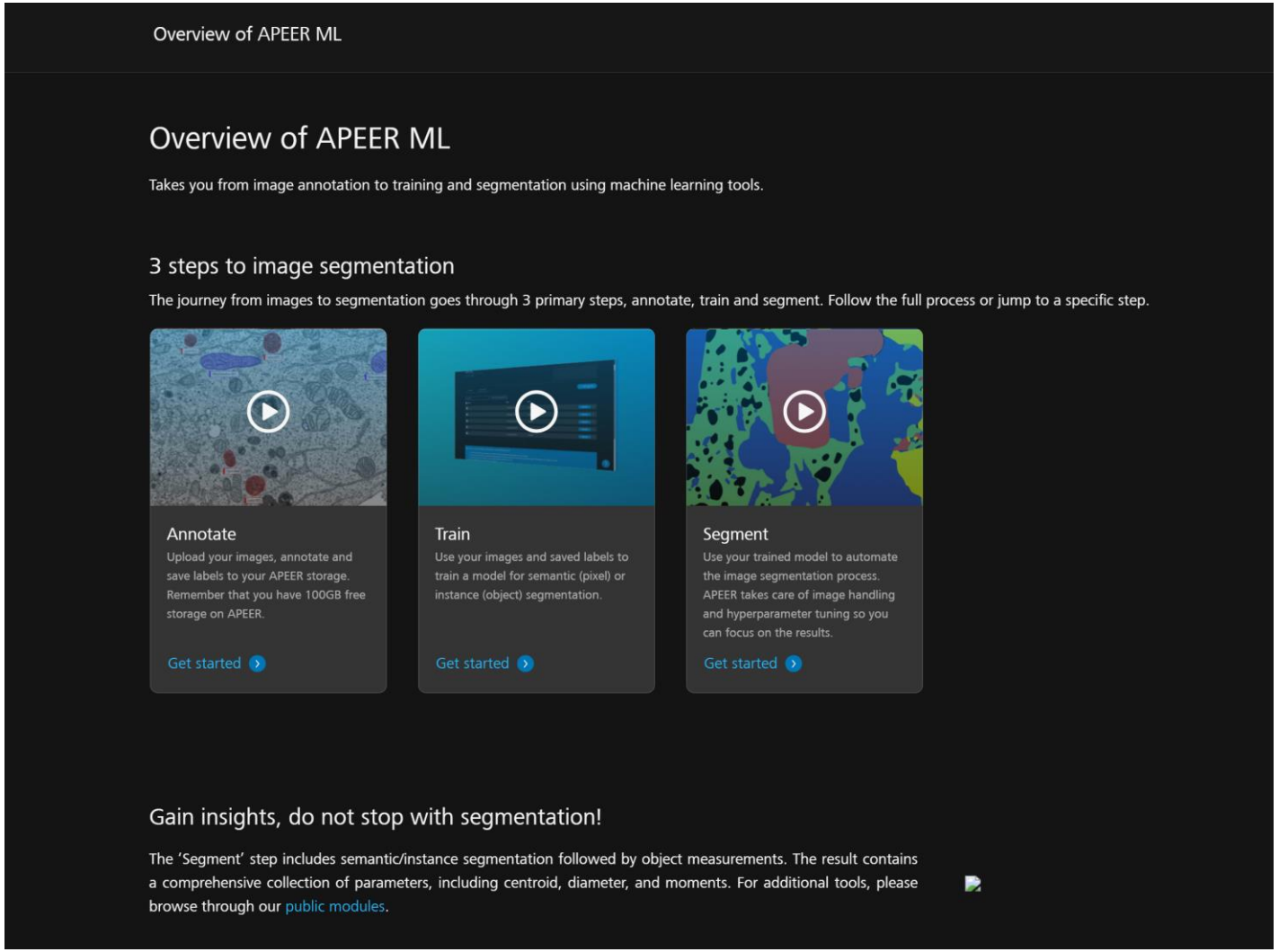




Clearly defined steps to guide the user from annotation to training to segmentation

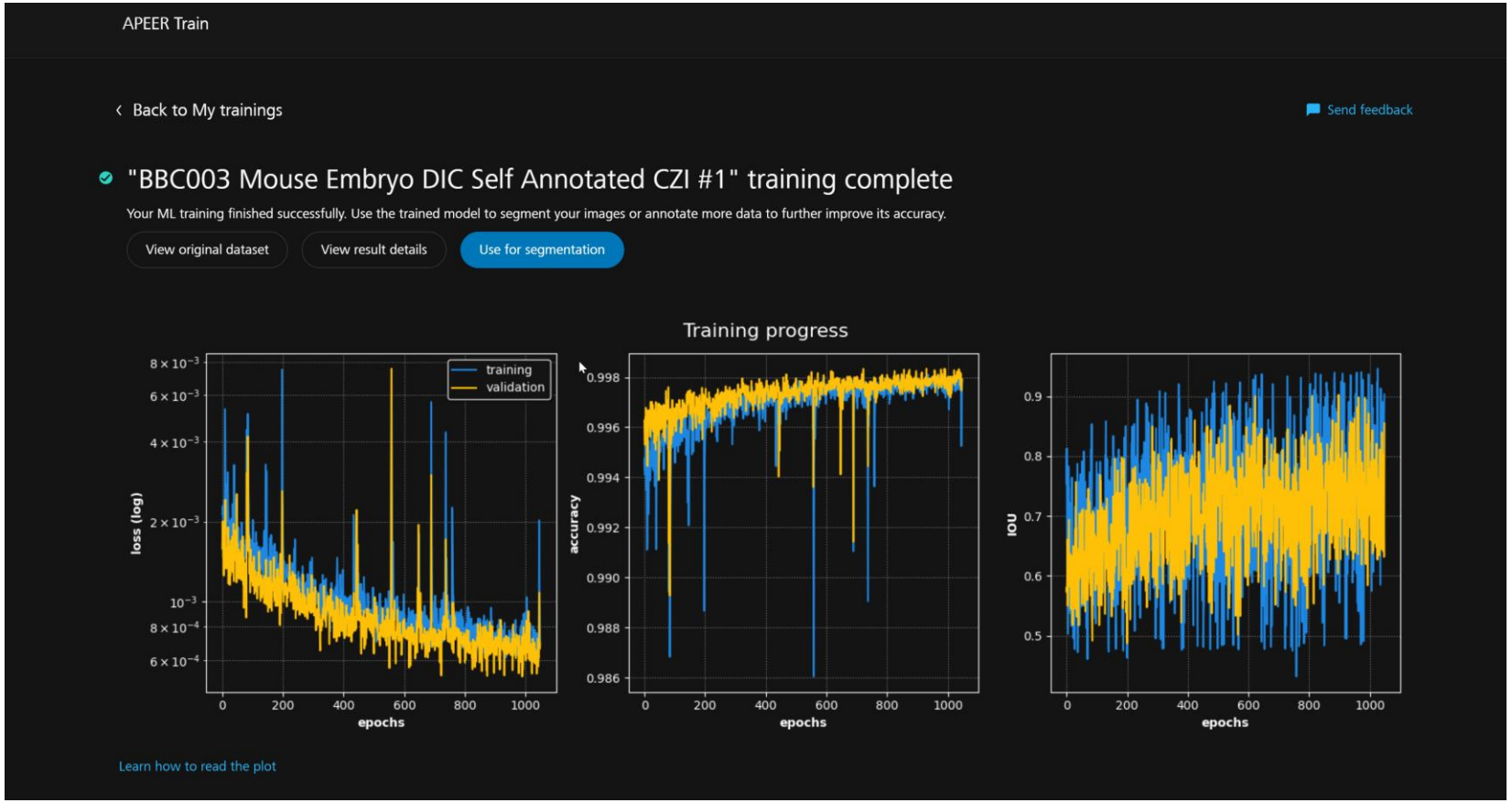
Machine learning

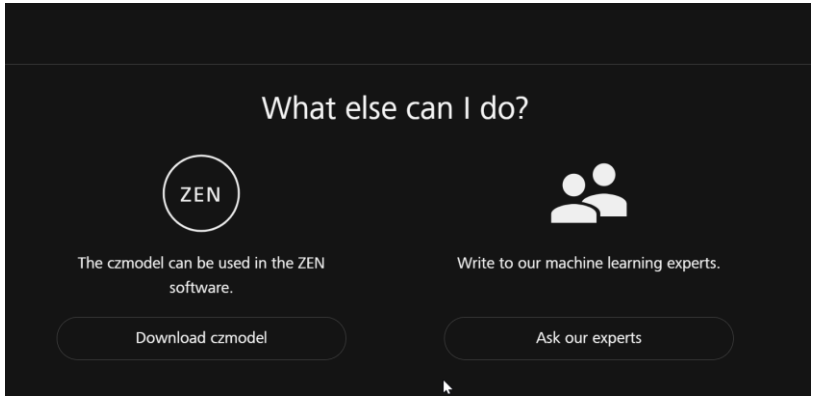
ZEISS



Clearly defined steps to guide the user from annotation to training to segmentation

Machine learning



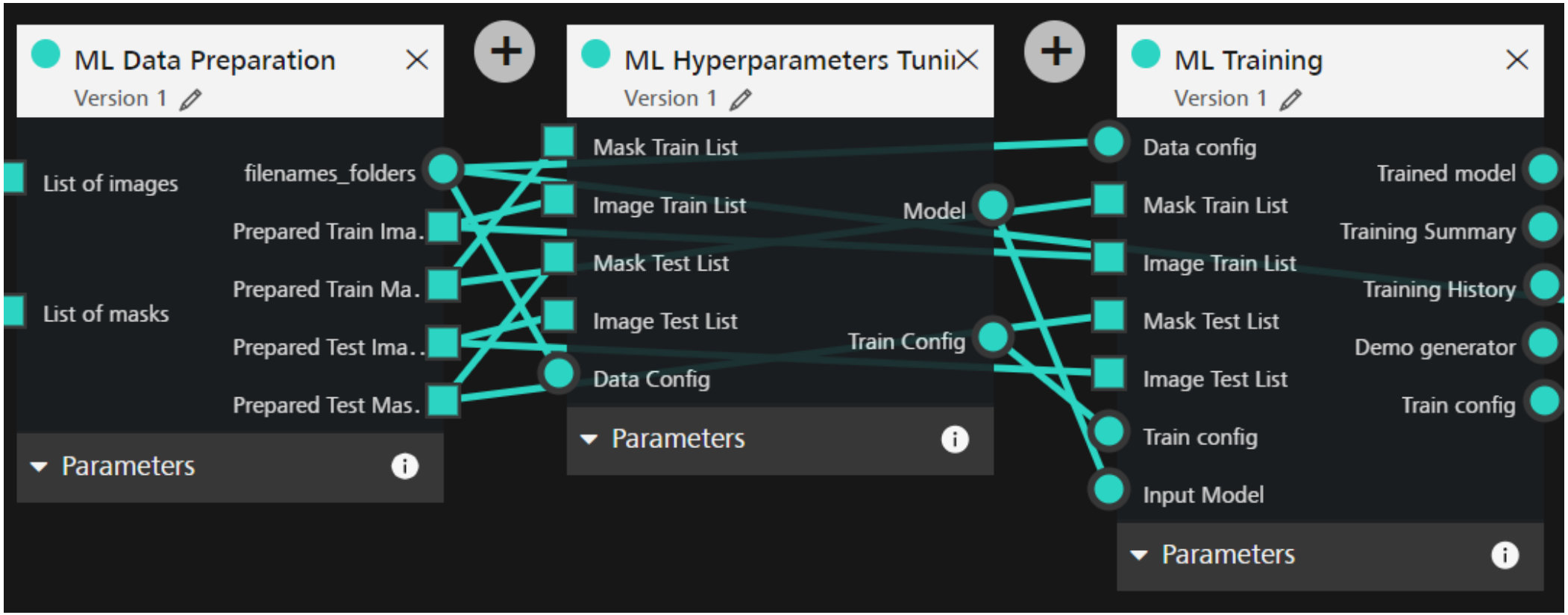


Clearly defined steps to guide the user from annotation to training to segmentation

Machine learning

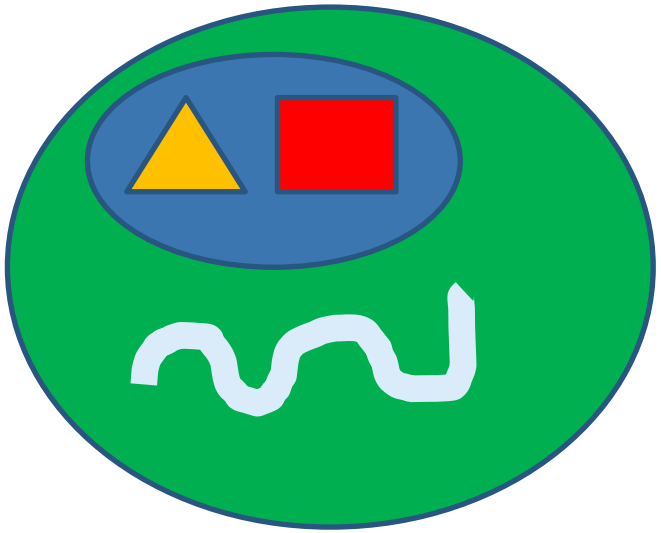
ZEISS

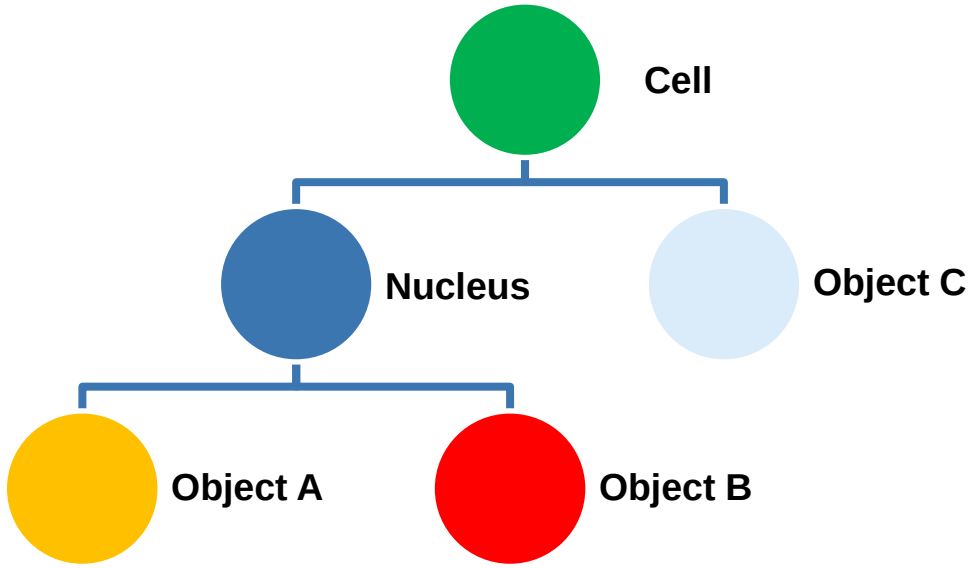
User will never see nor must interact with these complex background tasks.



Class Segmentation – Hierarchical structures with independent segmention per class

Machine learning





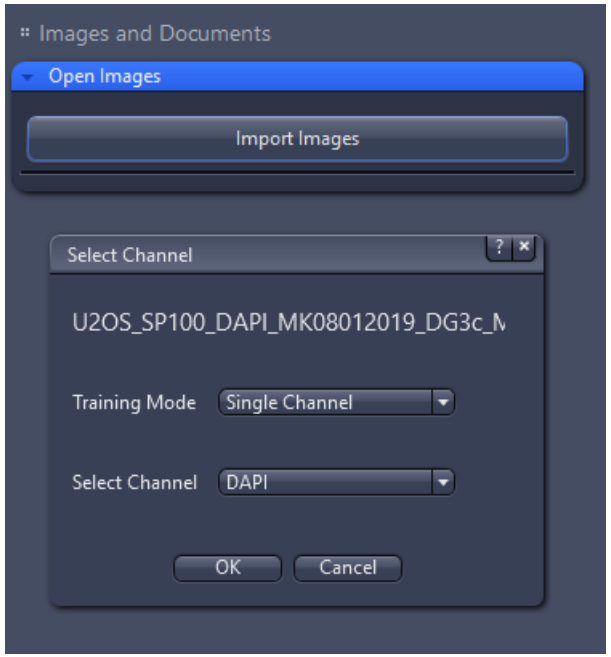
Flexible & Hierarchical Segmentation Scheme (Class Segmentation)

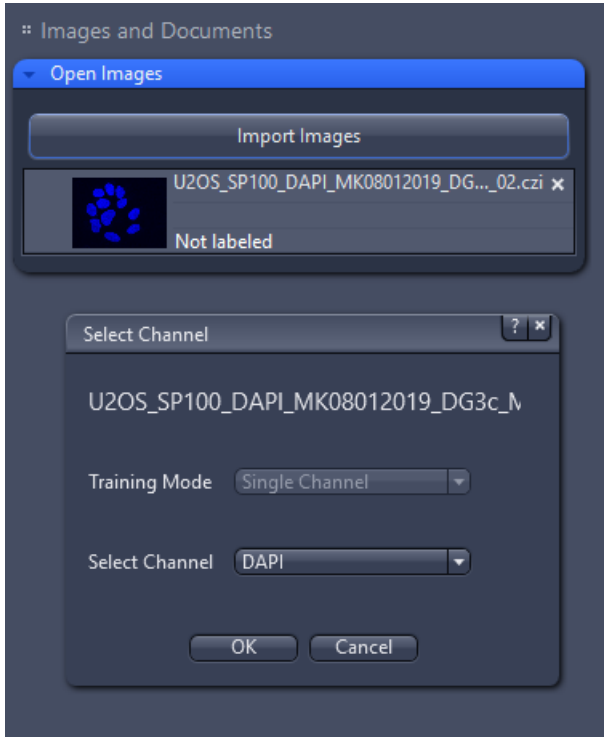
- Cell - classical threshold
  - Nucleus – DNN Nucleus Detector
    - Object A - trained model A
    - Object B - trained model B
  - Object C - classical threshold

Class Segmentation – Hierarchical structures with independent segmenter per class

Machine learning

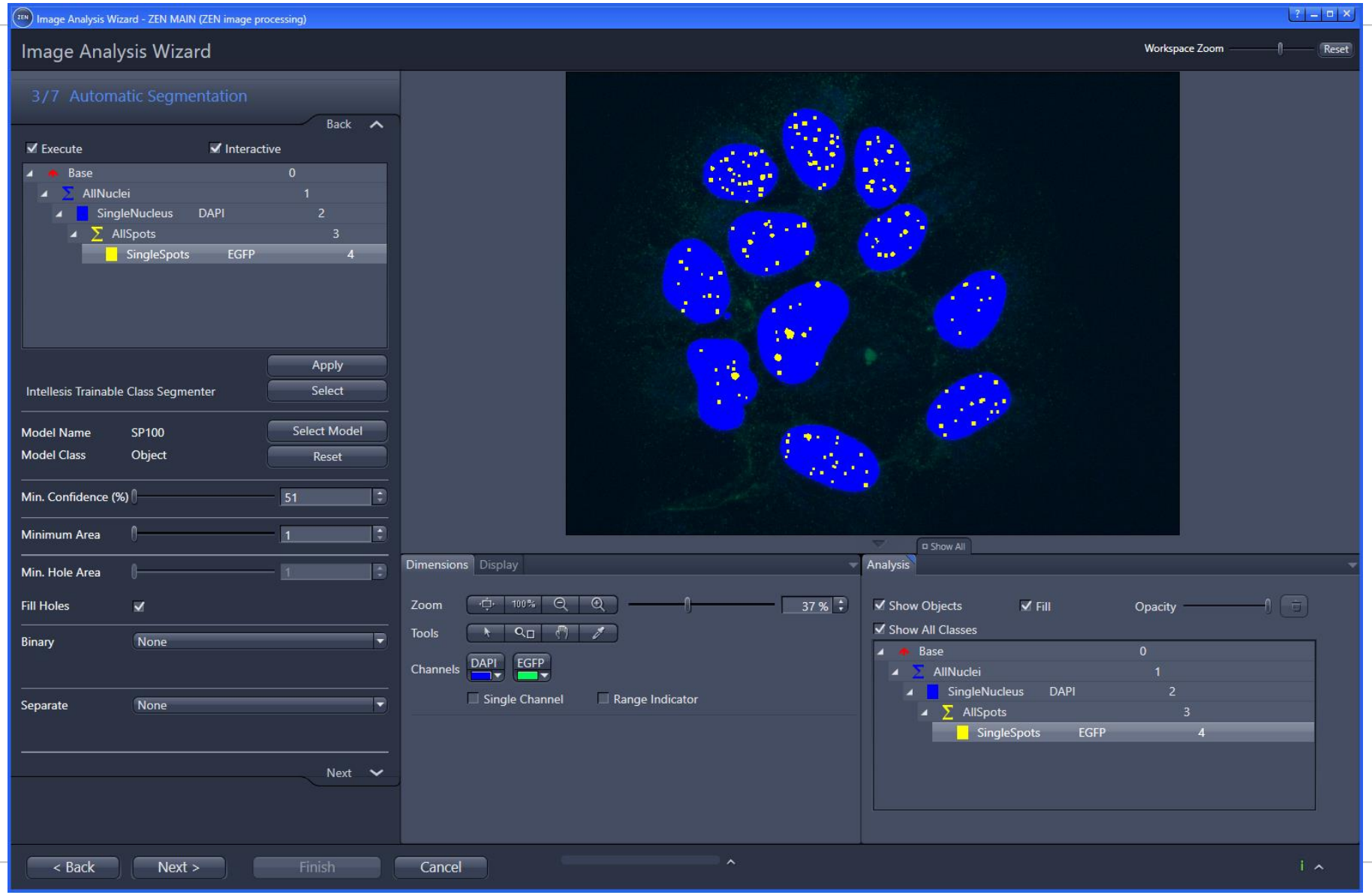
- Import dialog for channel selection incase of multichannel image
- Single Channel is the default
- Next import allows only selecting the channel





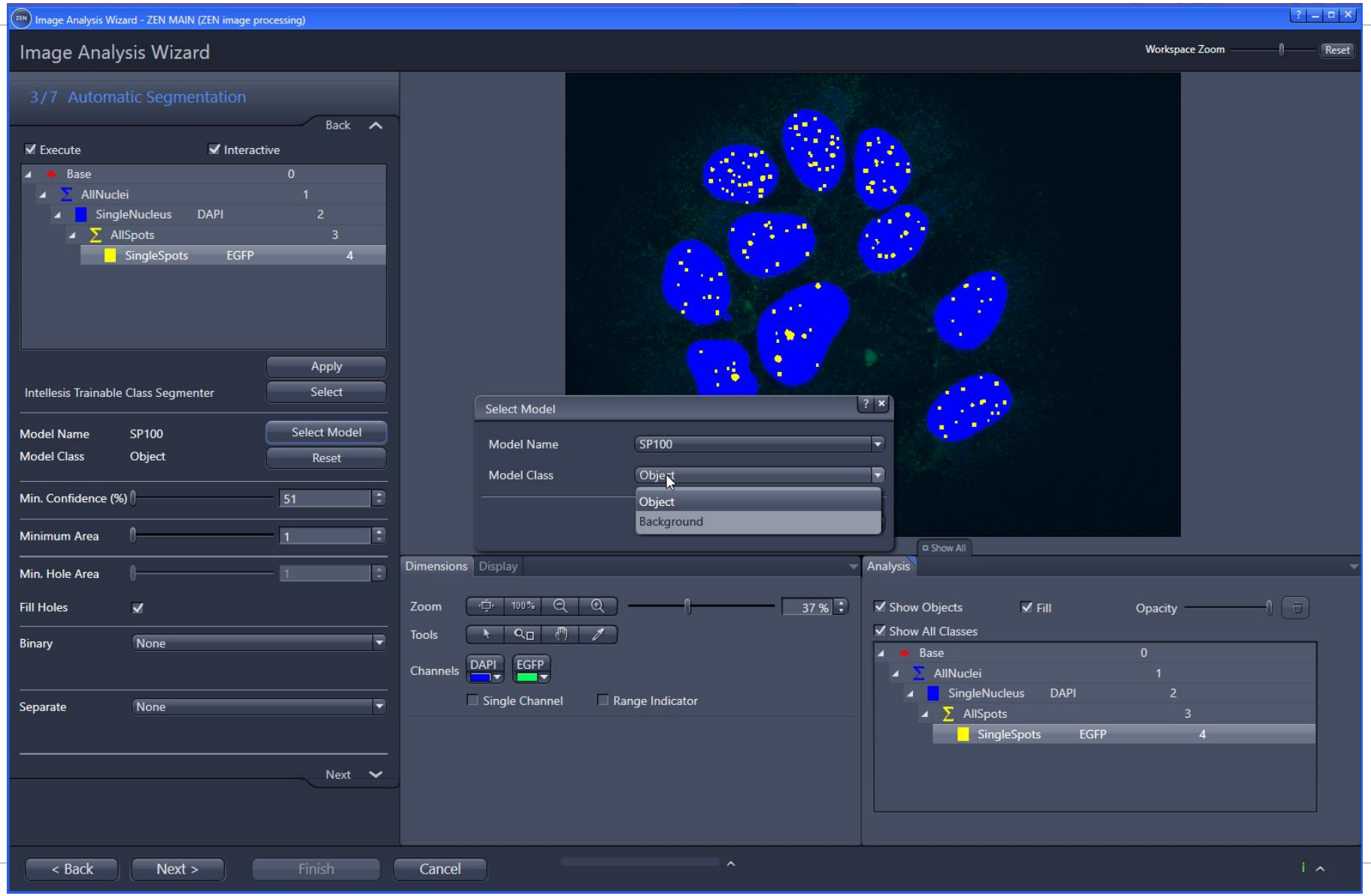
Class Segmentation – Hierarchical structures with independent segmenter per class

Machine learning



Class Segmentation – Hierarchical structures with independent segmenter per class

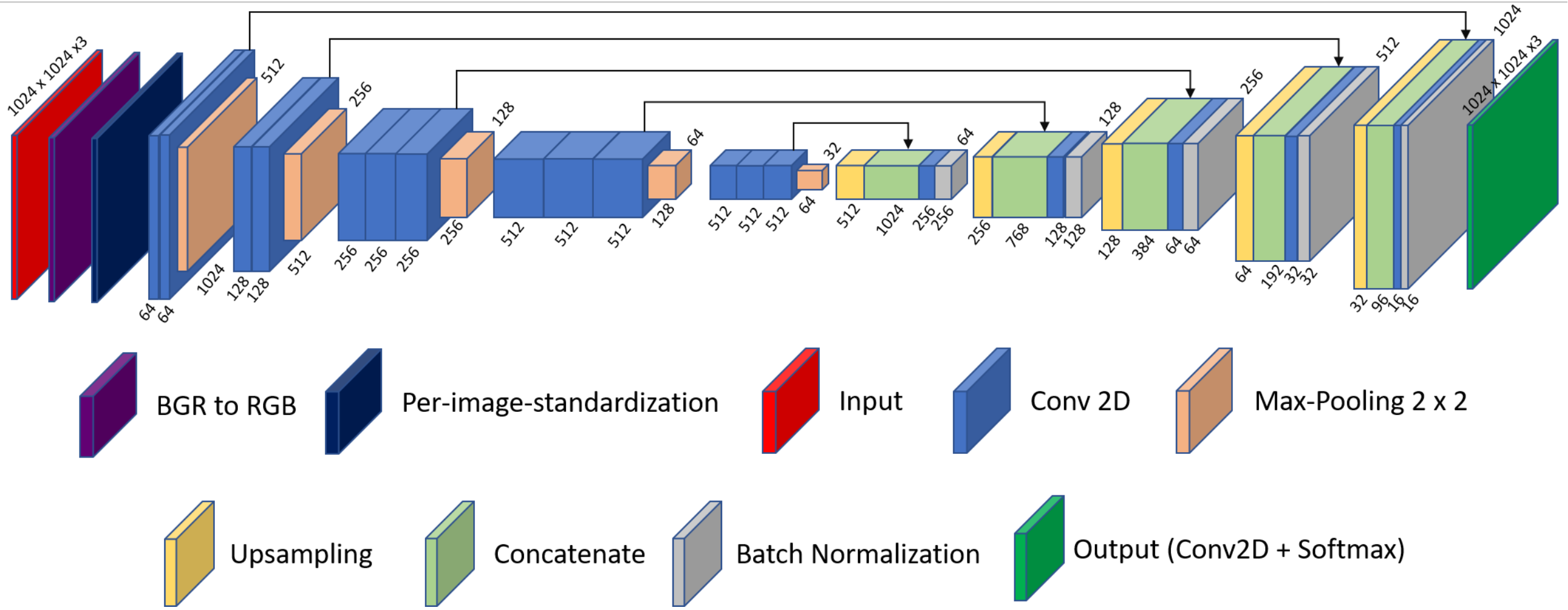
Machine learning



Nucleus Detector – UNet architecture (vgg16 encoder)

Try out pretrained models from ZEISS

Machine learning

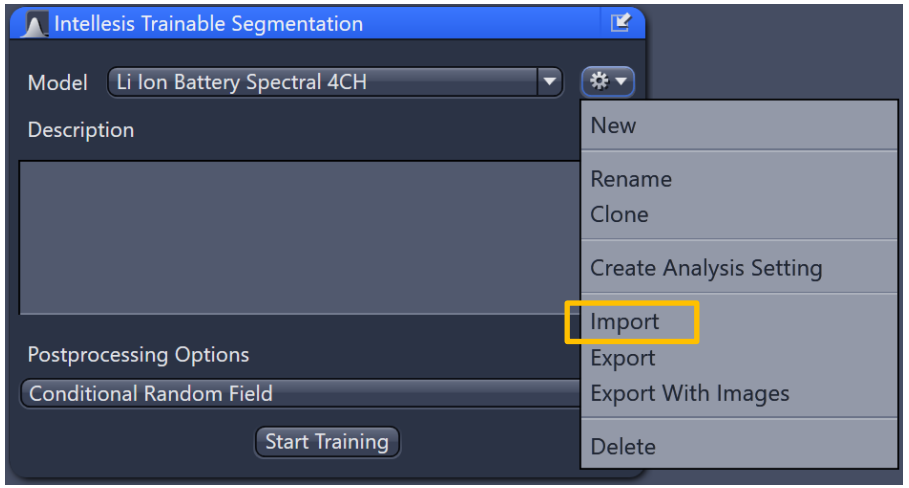


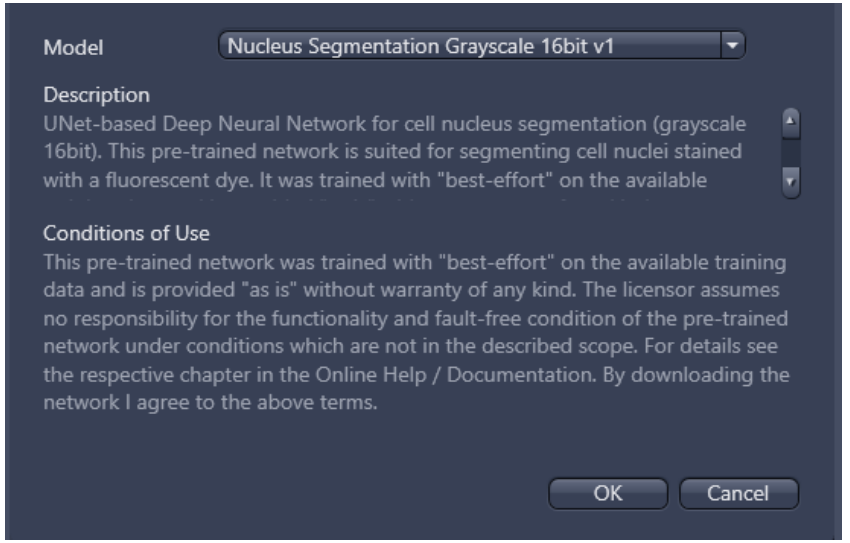
- Number above represent the array sizes
- numbers below represent the number of feature maps (per layer)

Use Deep Neural Networks for a specific application

Machine learning

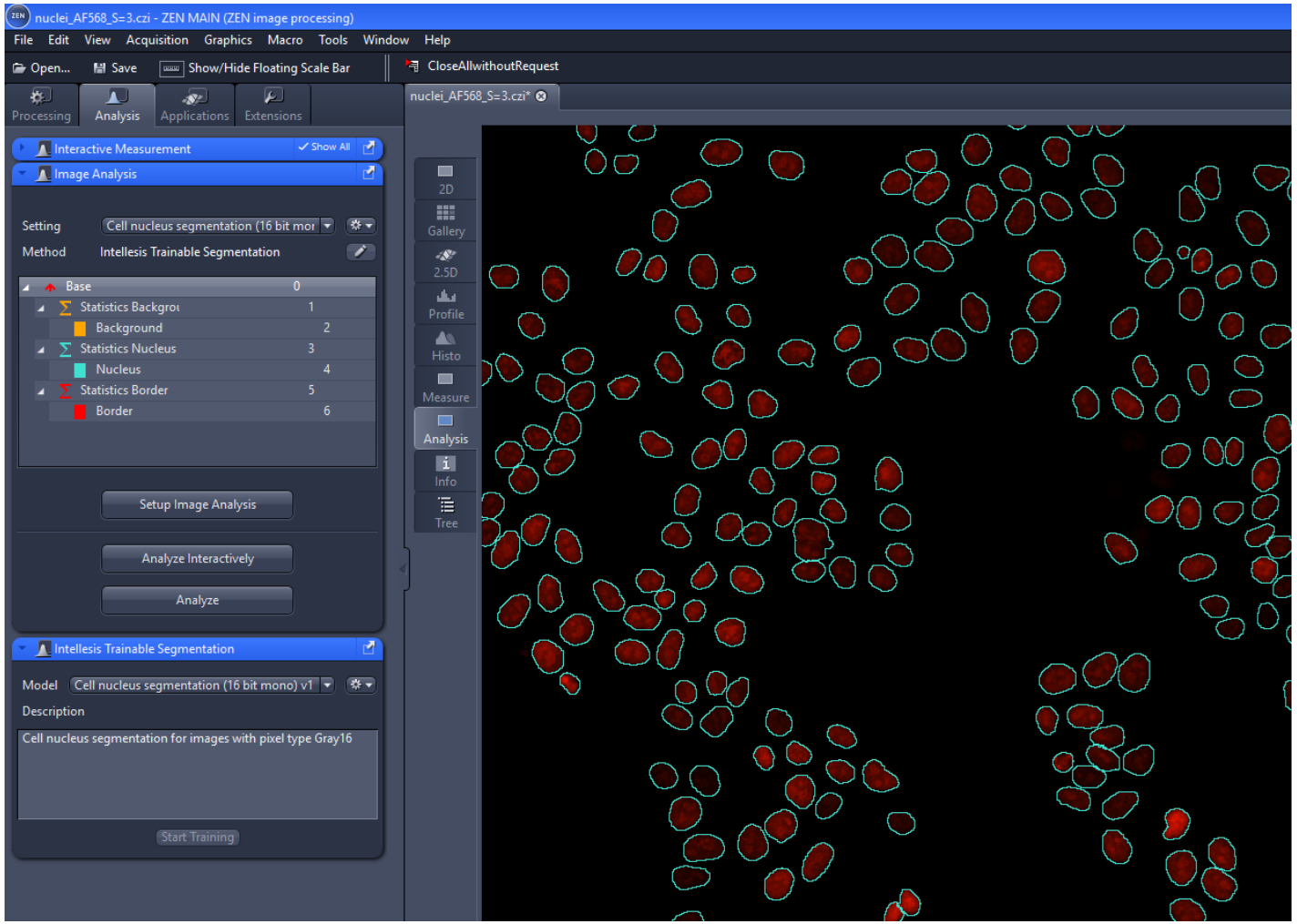
- Import pretrained networks provided ZEISS, APEER, SCS-Team or Researchers
- ZEISS example application: Pre-trained Nucleus Detection
- Use directly for Segmentation, Image Analysis incl. Class Segmentation





Pretrained Network – Integration into ZEN Image Analysis

Machine learning



- Network can be used to cerate to create a fully functional Image Analysis Pipeline by one single click
- Such IA settings will now also have the advanced binary postprocessing options
- Confidence Threshold is available

Side Note – Deep Neural Network versus Threshold

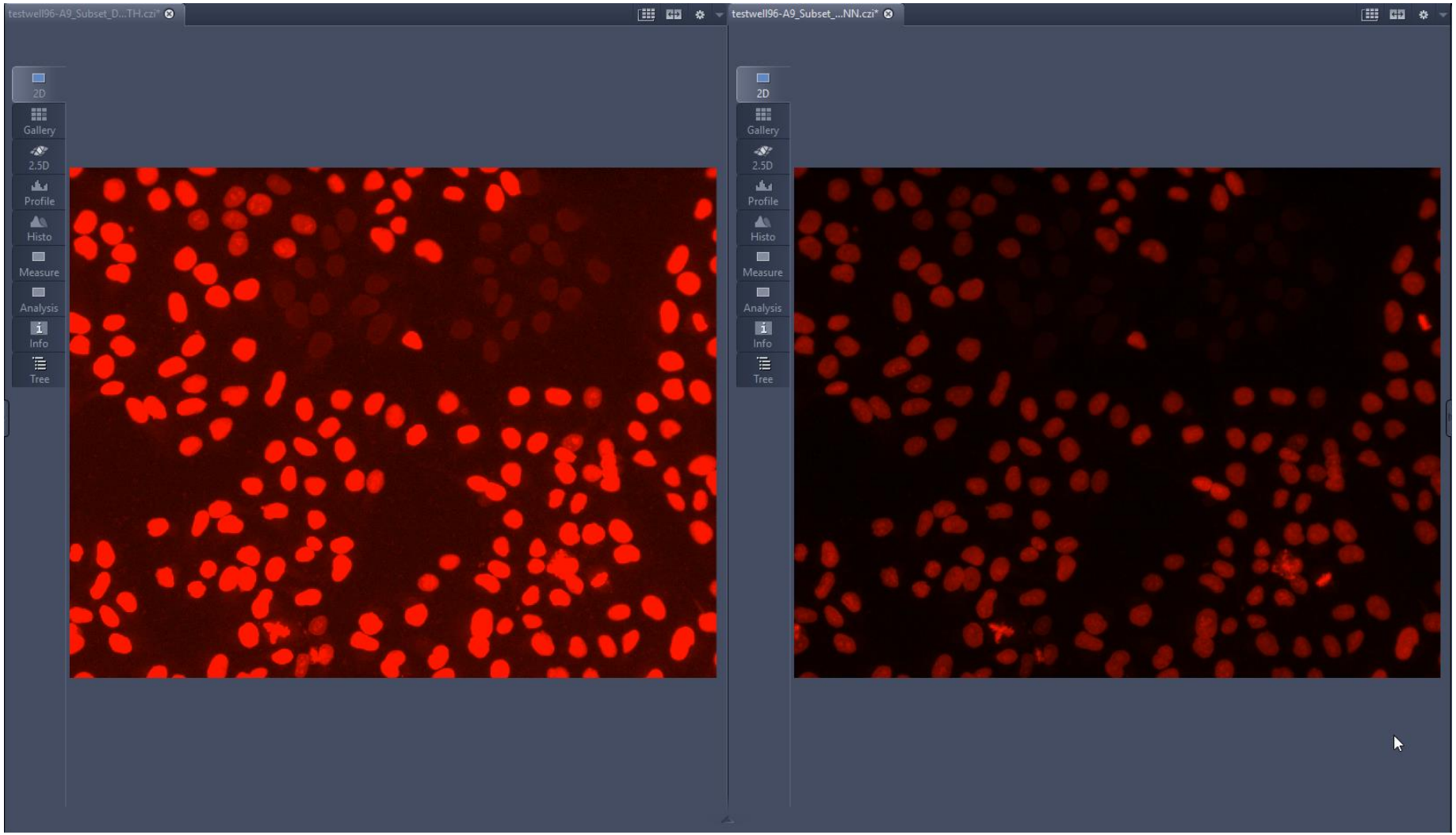
Machine learning

- Starting with ZEN Blue 3.1 it is possible to download / import pretrained networks for Segmentation
- Check it out: Model Download
- While Thresholding combined with classical PreProcessing function is really fast, it can be challenging (or impossible), especially for the non-expert user, to get the same results
- To illustrate this the following slides shows the same image analyzed by an Image Analysis Setting using
  - preProcessing + Thresholding
  - DNN-based Segmentation

Side Note – Deep Neural Network versus Threshold

Same image data – different display curve

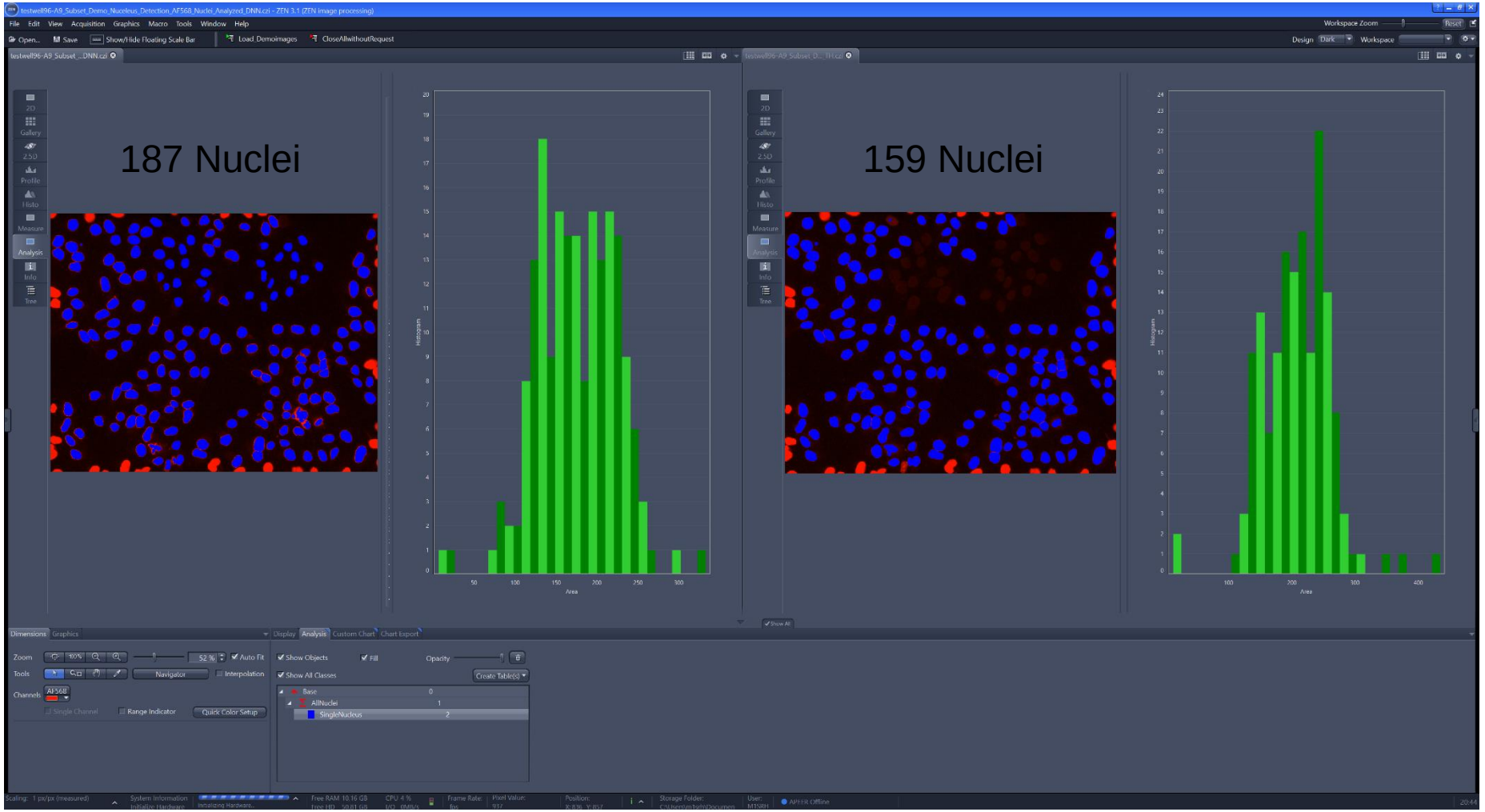
Machine learning



Side Note – Deep Neural Network versus Threshold

Same image data – different results

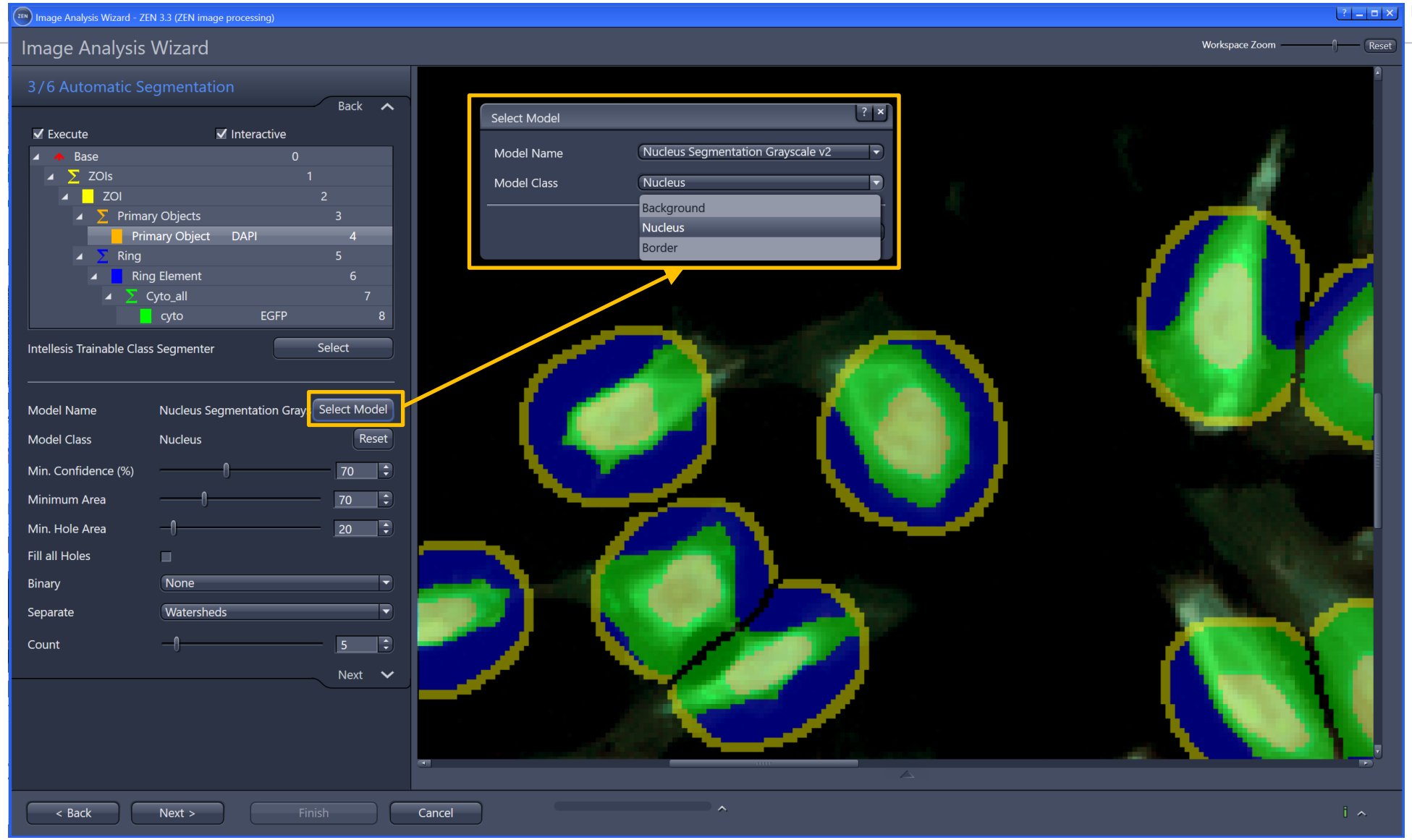
Machine learning



Advanced Class Segmentation incl. trained models

Plugin your model into any image analysis class

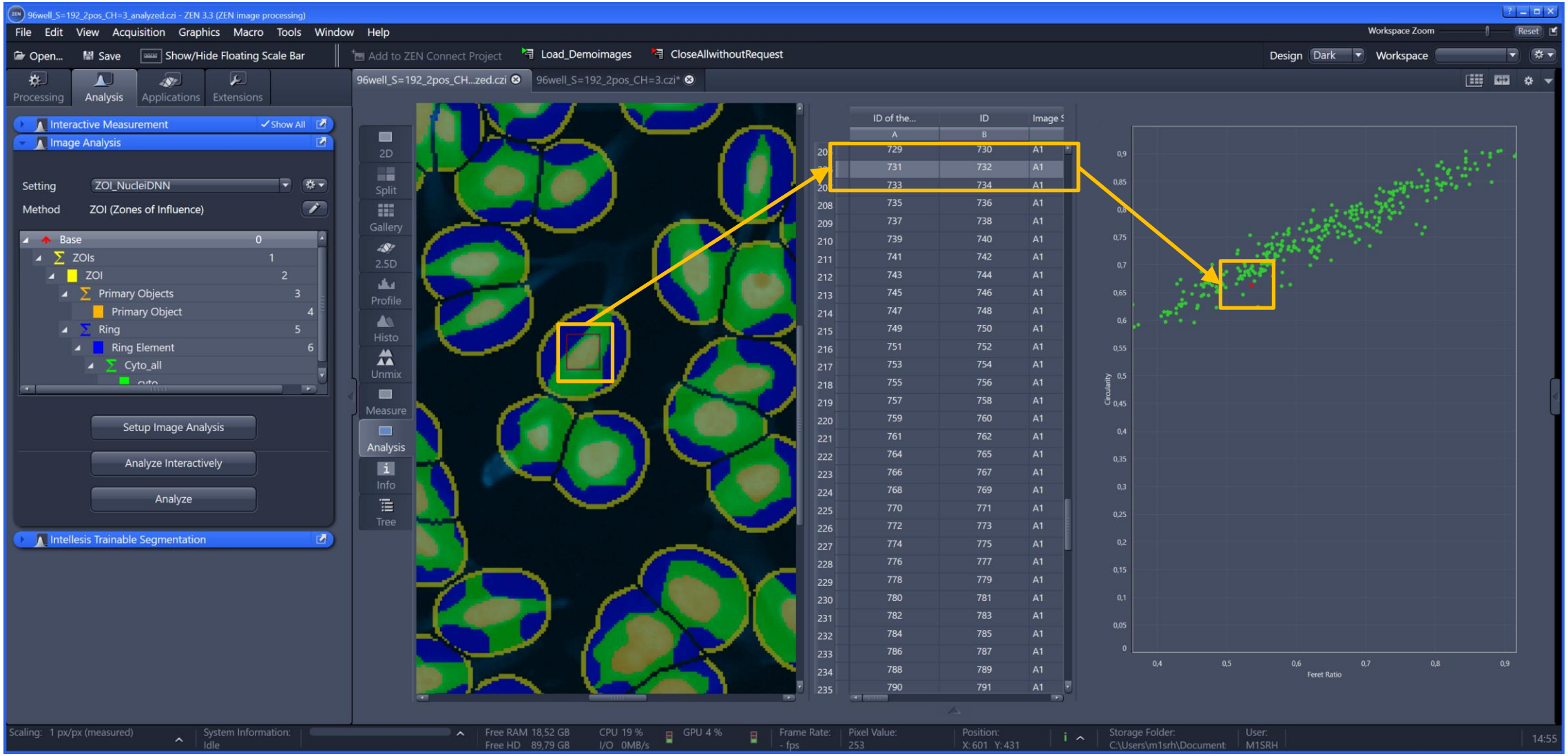
Machine learning



Advanced Class Segmentation incl. trained models

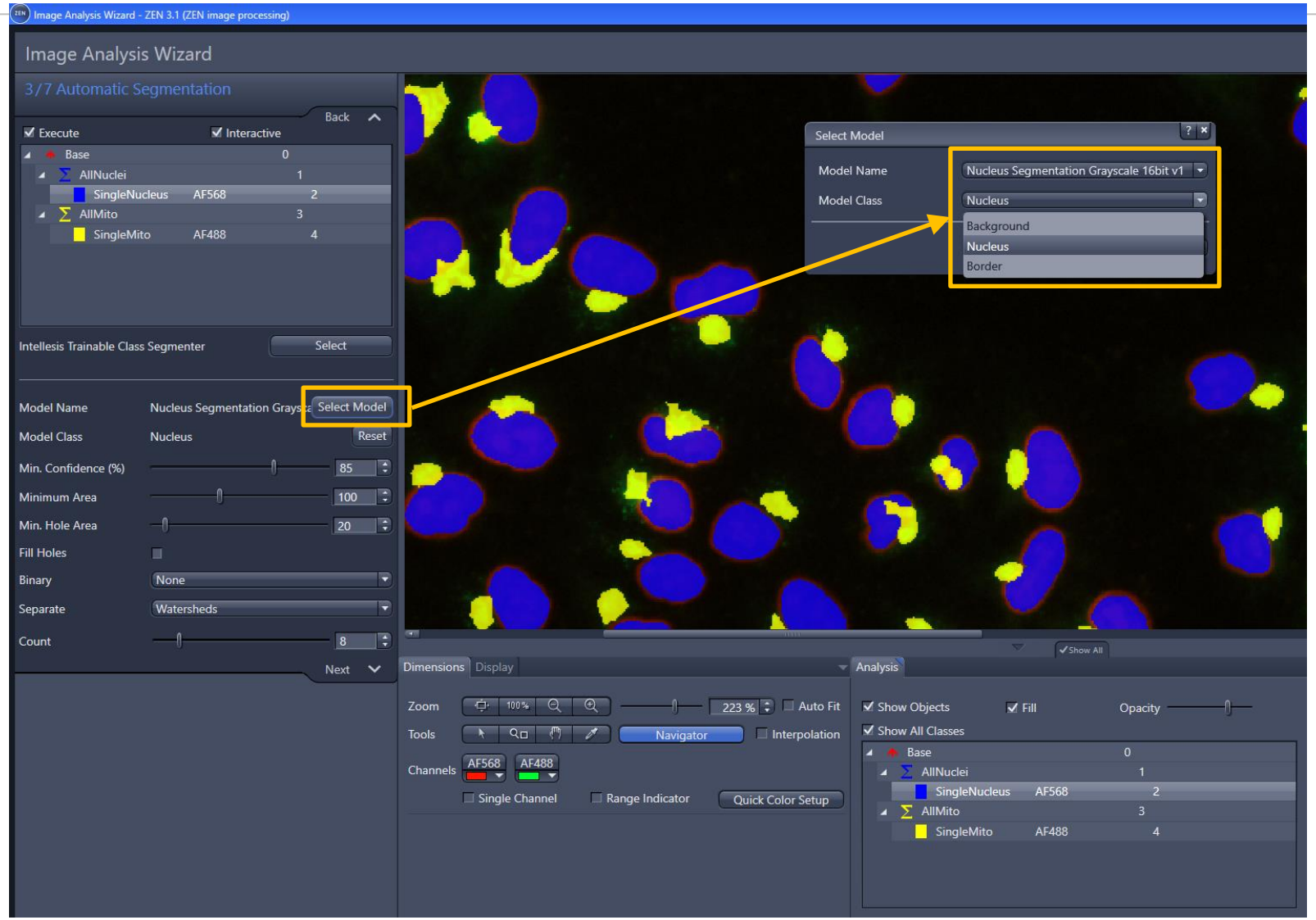
Easily visualize the result of your machine-learning based segmentation

Machine learning



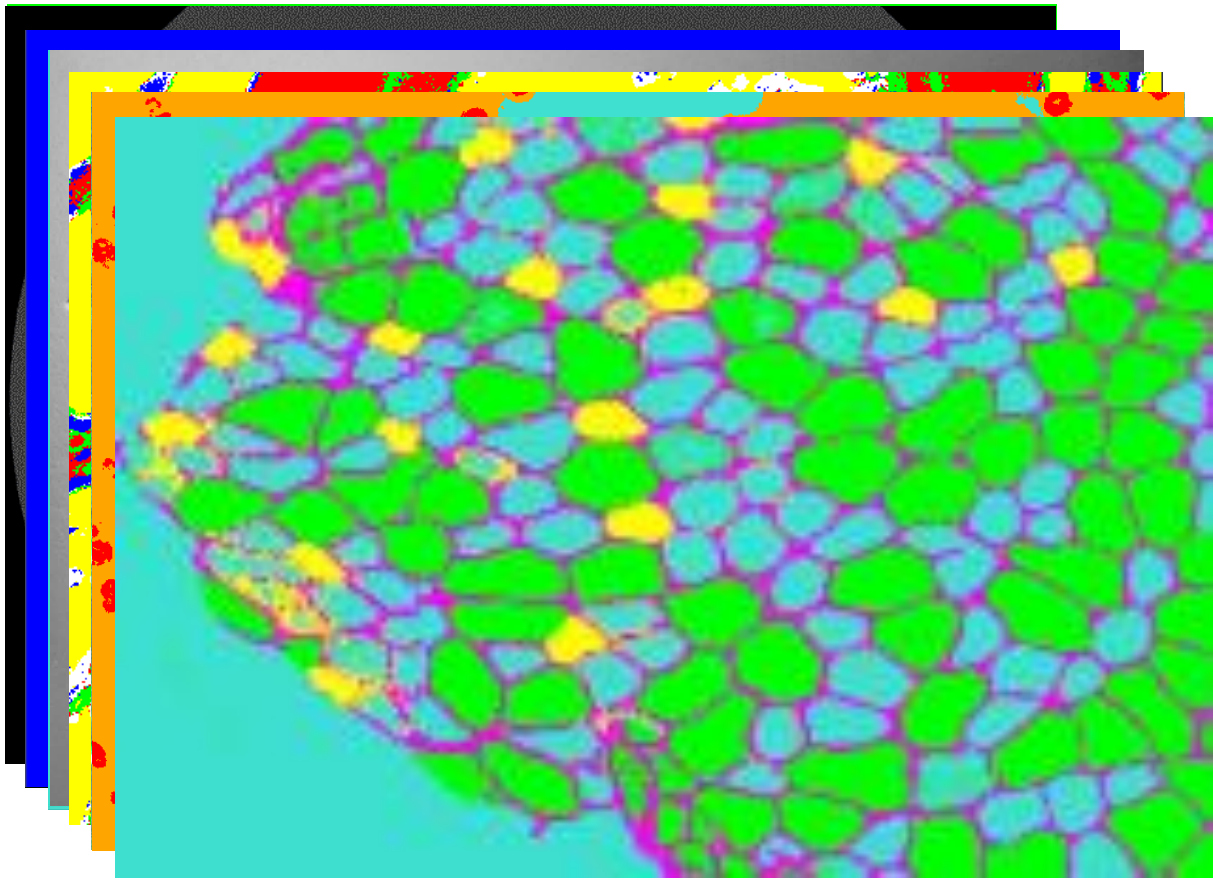
Nucleus Detector – Use inside Class Segmentation

Machine learning



Application Examples – Life Science

Machine learning

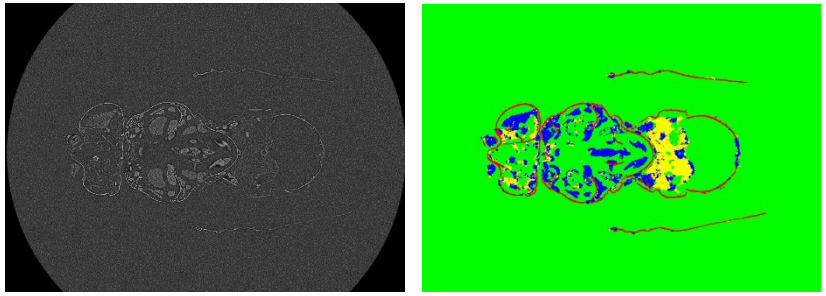


Sample courtesy of [illegible]

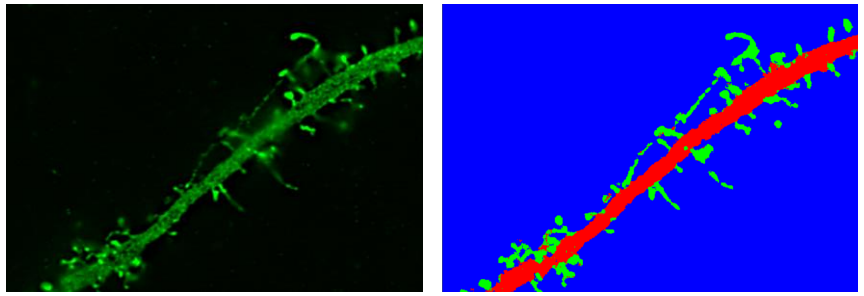
- XRM scan: Drosophila
- Elyra SIM: Neuron – spines & dendrite
- Celldiscoverer 7 (BF): Organoid and migrated cells
- EM: classification of cell compartments
- Celldiscoverer 7 (BF + FL): Scratch assay – area and mitotic cells
- Airyscan Fast: Muscle section – different myosin types

Application Examples – Life Science

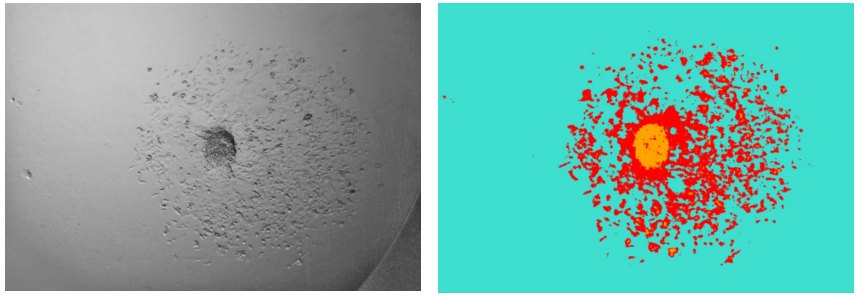
Machine learning



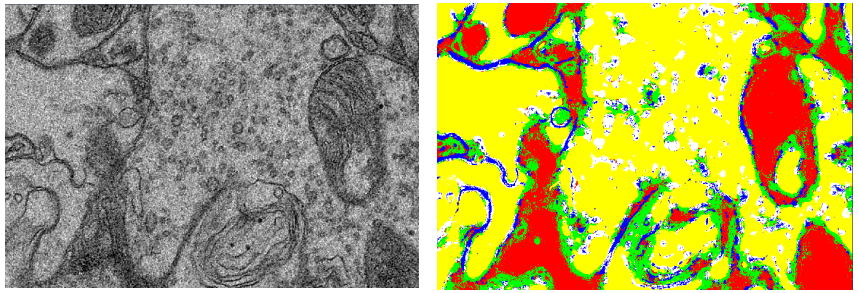
XRM scan: Drosophila



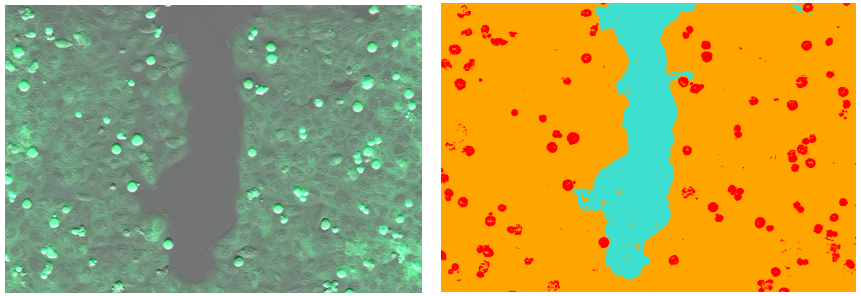
Neuron – spines & dendrite



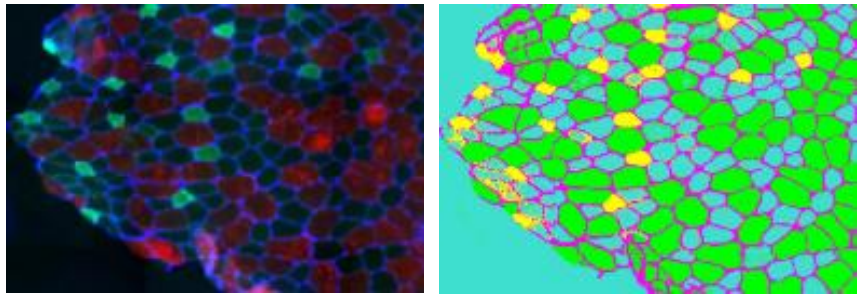
Organoid and migrated cells



Classification of cell compartments



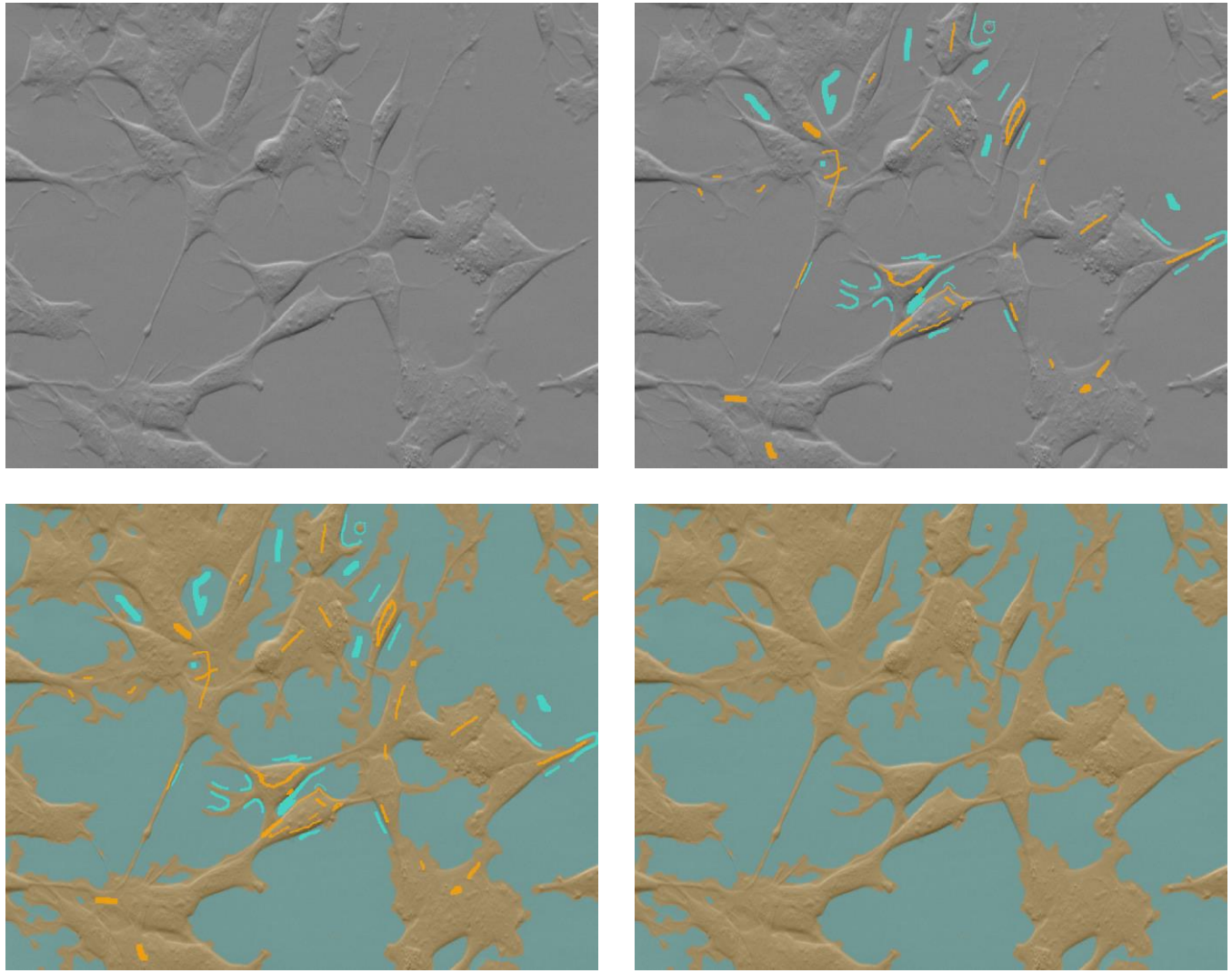
Scratch assay – area and mitotic cells



Muscle section – different myosin types

Application Examples – Life Science

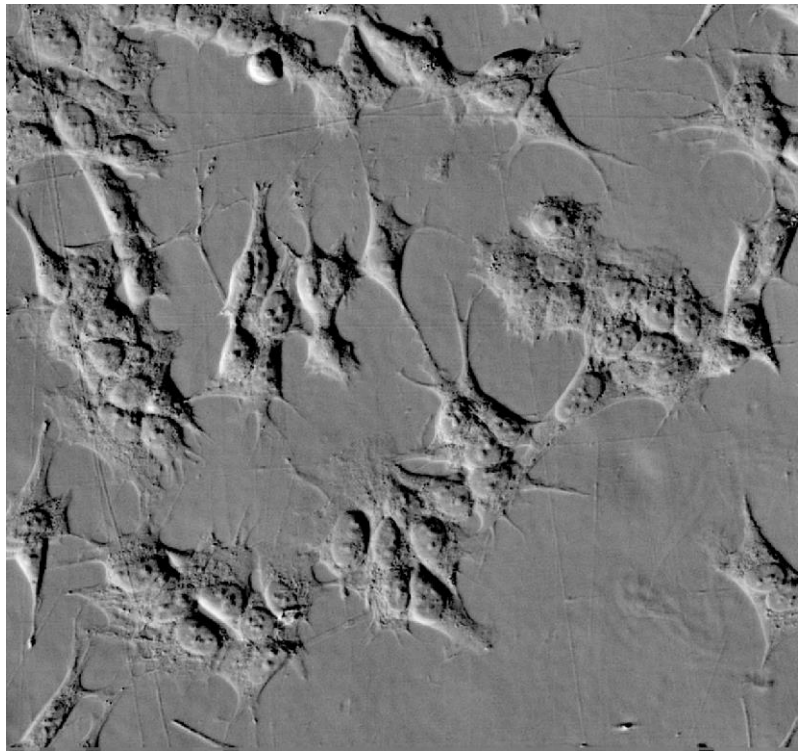
Machine learning

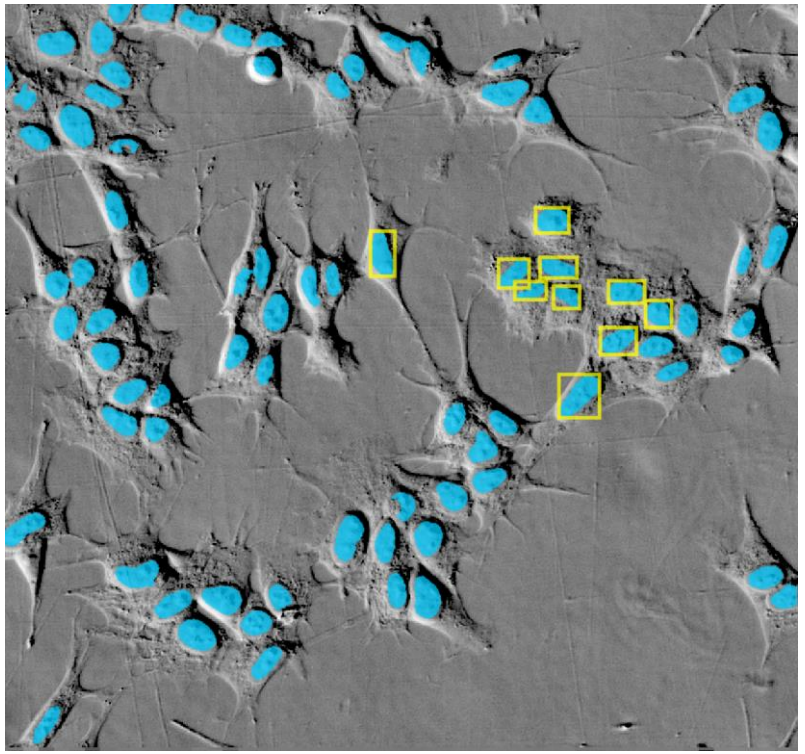


- Cells image using Phase-Gradient Contrast on a CD7
- Labeled with 2 classes inside Intellesis Training UI
- Feature Extractor: DeepFeatures256 + CRF Postprocessing

Application Examples – Life Science

Machine learning





- Cells image using Phase-Gradient Contrast on a CD7
- Nucleus Segmentation using a Deep Neural networks trained on the APEER platform
- Segmented nuclei shown inside ZEN blue Image Analysis

ZEN and APEER - Open Ecosystem for integrated Machine-Learning Workflows

Machine learning

ZEISS

Intellesis Landing Page

Get the trial license and try it out!

APEER - Machine Learning Overview

Additional Content on GitHub:

Machine Learning

Machine Learning - Feature Extractors

Post Processing

We make it visible.