How to create an open ecosystem for data-centric model development at ZEISS

ZEN blue / ZEN core + APEER-ML + Vision4D

Dr. Sebastian Rhode

Product Owner Machine Learning Solutions
Staff Expert

Product Center for Software

06.05.2022

Mission Statement for AI @ZEISS

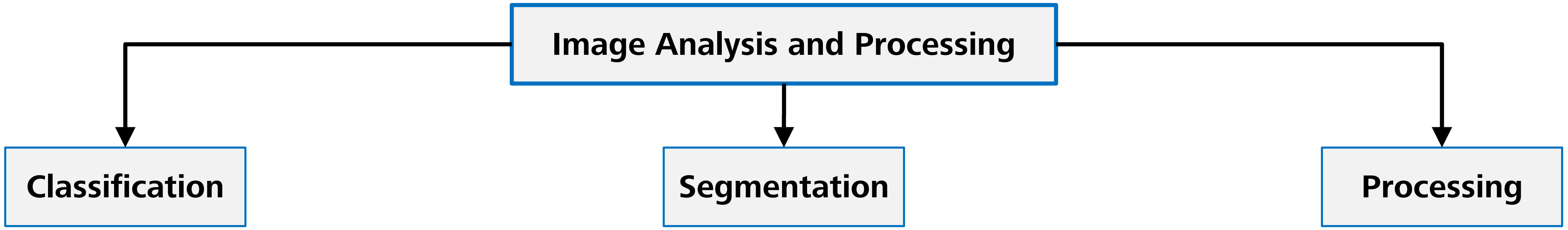
Our mission statement for AI @ZEISS Microscopy could be described as:

“Put the scientist back into the driver seat for Deep Learning”

One of our core messages when it comes to Image Analysis & AI solutions is:

“Better data beat better models”

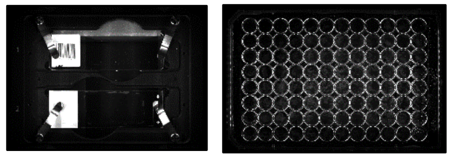
Focus of AI solutions @ZEISS microscopy

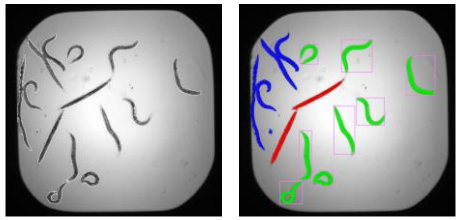


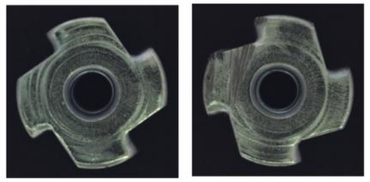
Classifying an entire image or individual objects

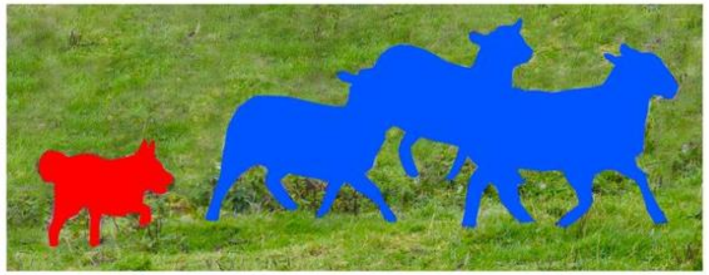
Refers to classifying at a pixel level

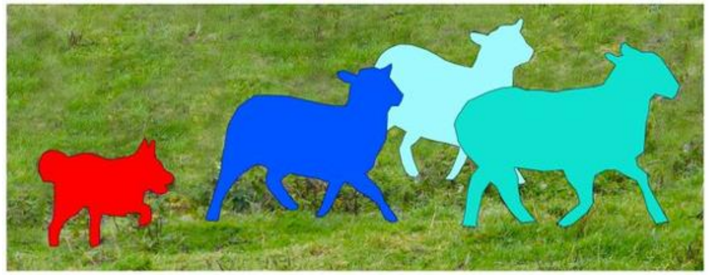
Image corrections and enhancements

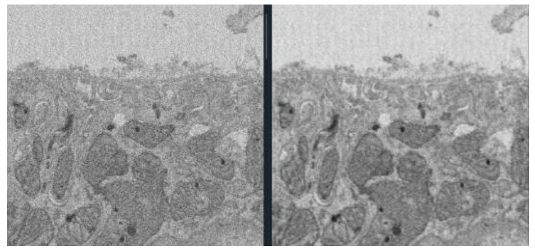

Recognize a Sample Carrier

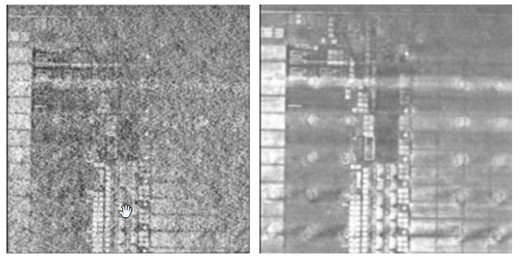

Classify objects in analyzed images


“Good” part vs “Anomaly”


Semantic Segmentation


Instance Segmentation

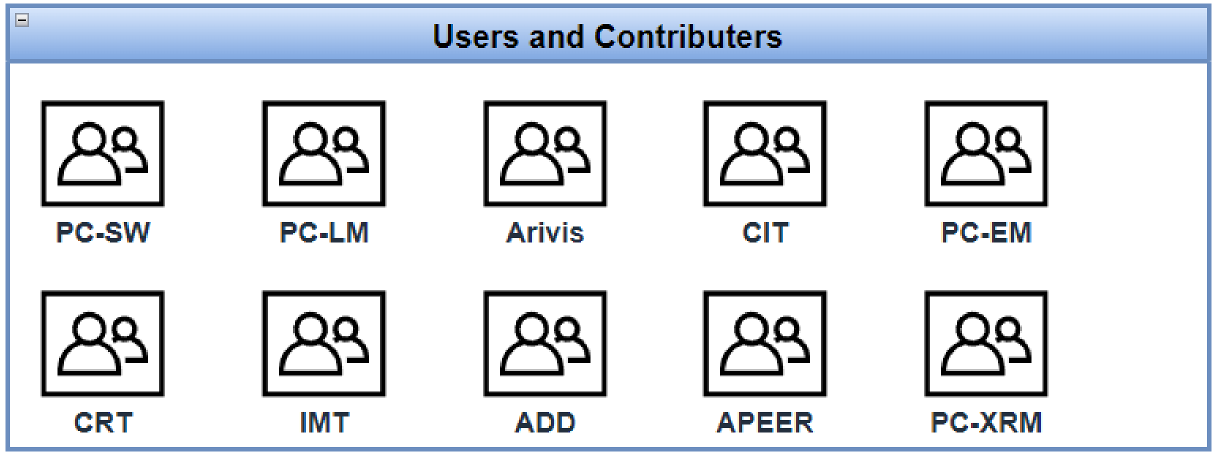

Denoising & Image2Image


Reconstruction

Creating such AIs tools in a company is a bit complicated ☺

@ZEISS there are many AI stakeholders that need to be aligned …

… and creating AI tools and solutions is fun (mostly) = everybody is tempted to create their own solutions ☺

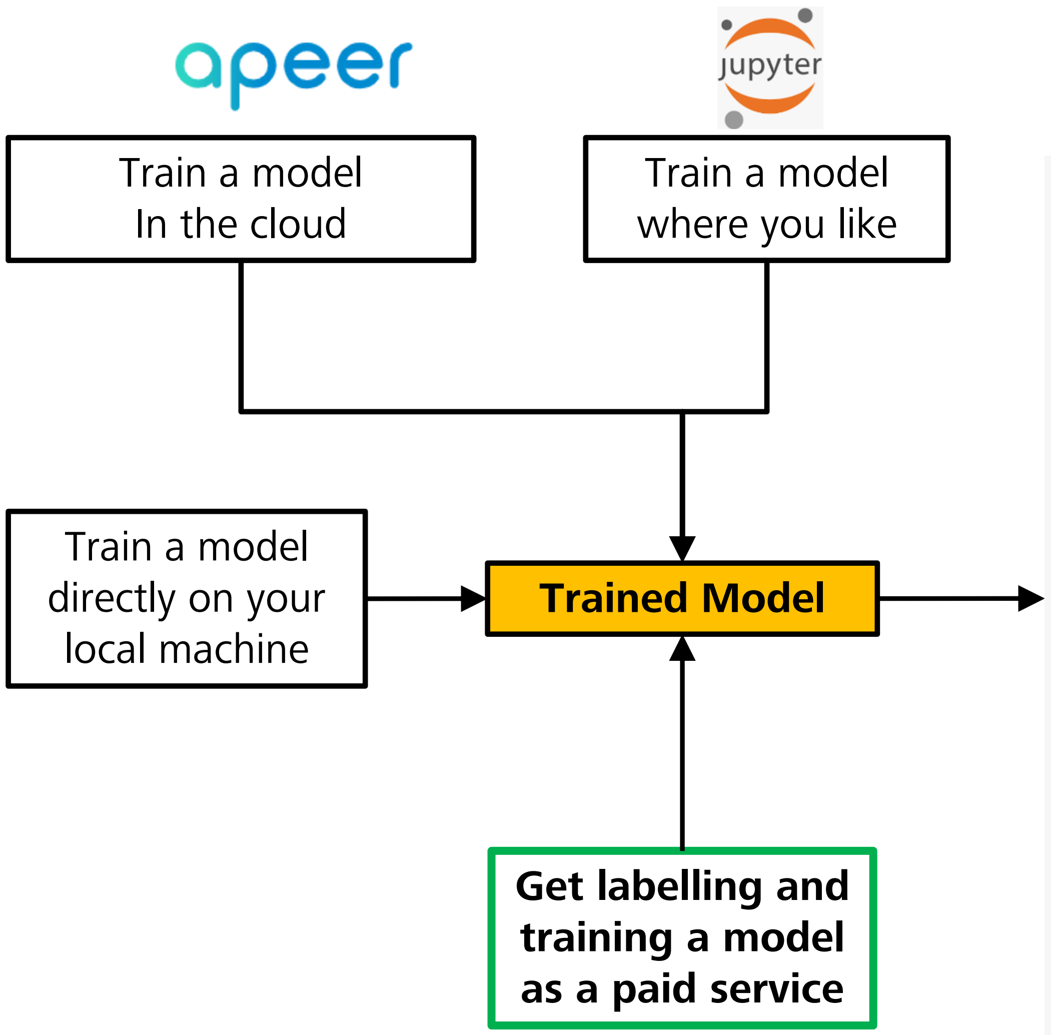


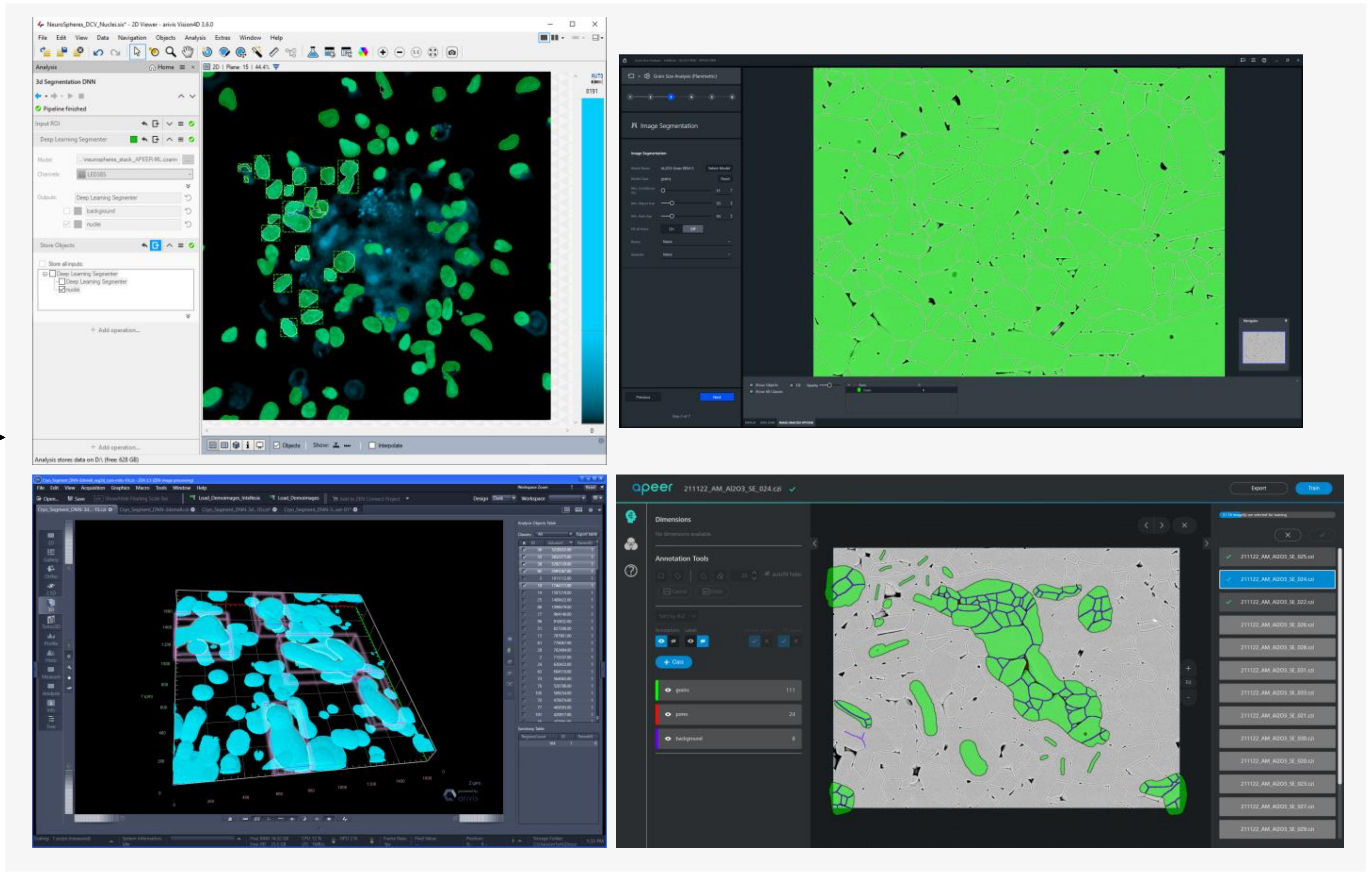
- PC-SW/LM/EM/XRM = Product Center for Software, Light, Electron and X-ray Microscopy
- CRT = Corporate Research and Technology
- ADD = Advanced Development
- … and more

Align Requirements from Academia and Industry with Business Model

- Academia wants open and flexible solutions and does not like to be locked in
- Industry often wants “Streamlined and Integrated” tools and “one-Button-Solutions” for a specific task
- various of open-source software (OSS) tools that offer specific solutions → what is our business model
- “cloud-computing” is trending, but many users do not want / are not allowed use cloud yet
- What tools should a we use? Should we develop our own?

Concept: Train models “anywhere”


apeer
jupyter
Train a model In the cloud
Train a model where you like
Train a model directly on your local machine
Trained Model
Get labelling and training a model as a paid service


ZEN blue, ZEN core, vision4D and APEER

Train Segmentation models “anywhere” and use them in ZEN

How do we implement such an **Open Ecosystem** without driving everybody in R & D crazy?

What components & tools do we need for that?

What is the best way to align everybody and create “maintainable” code and create real value?

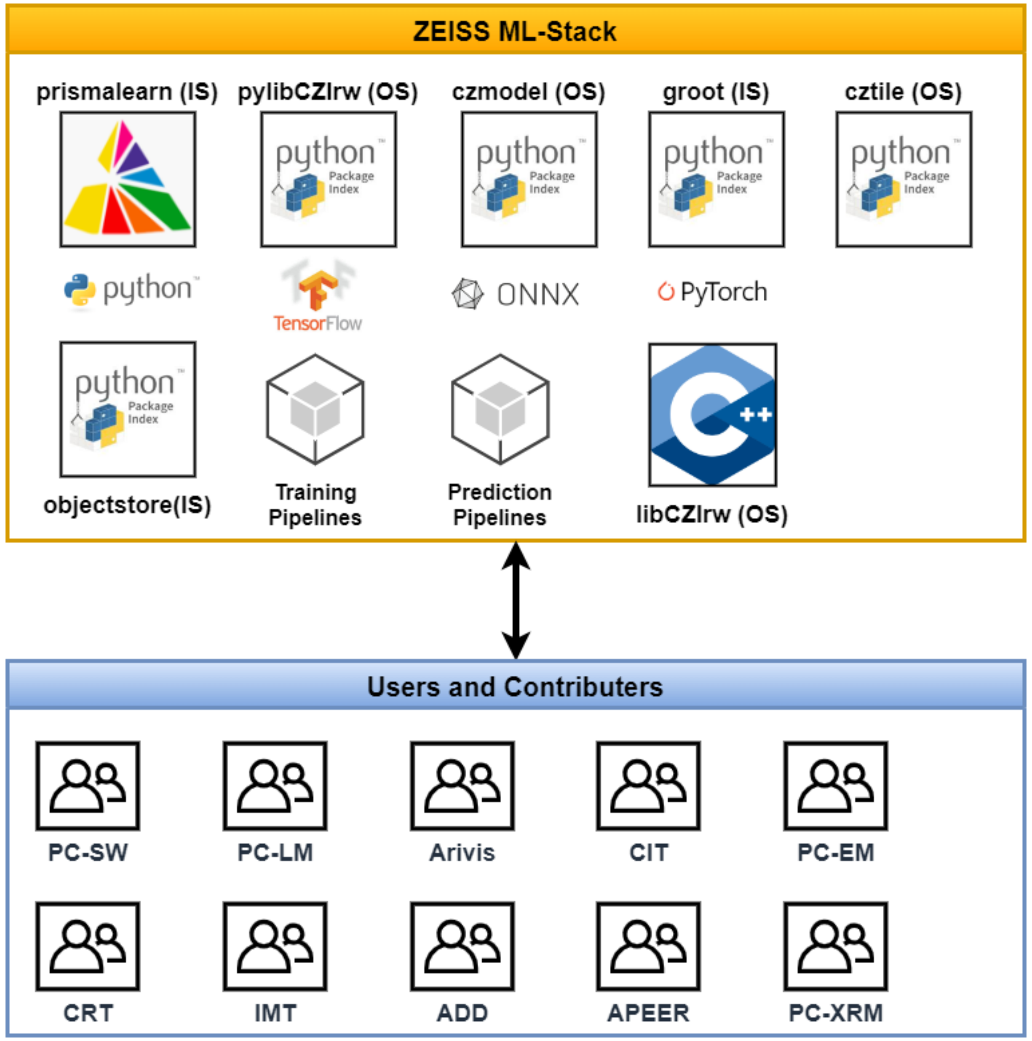
The aligment issue in bigger organizations

One nice way to align different "parties" in SW development is …

… to provide useful tools and APIs that make the life of people easier → then they will use it without telling them to do so ☺

This is especially true inside a companies and therefore we decided to create what we call the "ZEISS ML-Stack"

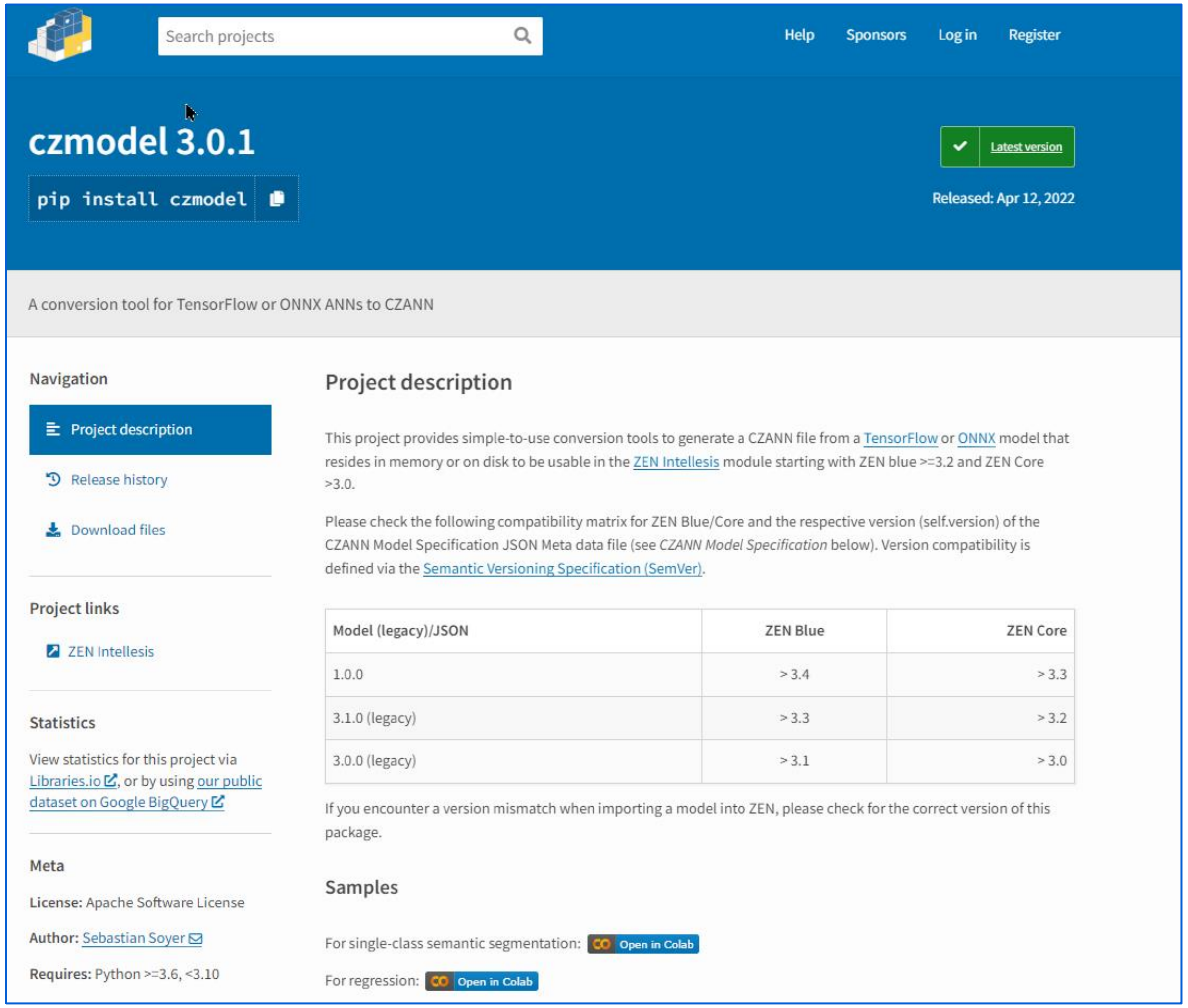
Our ecosystem - ZEISS Machine Learning Stack



ZEISS ML Stack

- mainly Python-based internal (IS) and open-source (OS) packages
- easy to use via doing pip install XYZ in different development teams @ZEISS
- clear rules that those packages must the 1st choice when starting new projects
- test coverage and code-quality standards cannot be just be “nice thing to have” but are crucial to make this fly ☺

Open-Source python package czmodel
Store model along with metadata



czmodel 3.0.1

```
pip install czmodel
```

Released: Apr 12, 2022

A conversion tool for TensorFlow or ONNX ANNs to CZANN

Statistics

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

Meta

License: Apache Software License

Author: Sebastian Soyer

Requires: Python >=3.6, <3.10

Project description

This project provides simple-to-use conversion tools to generate a CZANN file from a TensorFlow or ONNX 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.

Please check the following compatibility matrix for ZEN Blue/Core and the respective version (self.version) of the CZANN Model Specification JSON Meta data file (see *CZANN Model Specification* below). Version compatibility is defined via the Semantic Versioning Specification (SemVer).

Model (legacy)/JSON	ZEN Blue	ZEN Core
1.0.0	> 3.4	> 3.3
3.1.0 (legacy)	> 3.3	> 3.2
3.0.0 (legacy)	> 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.

Samples

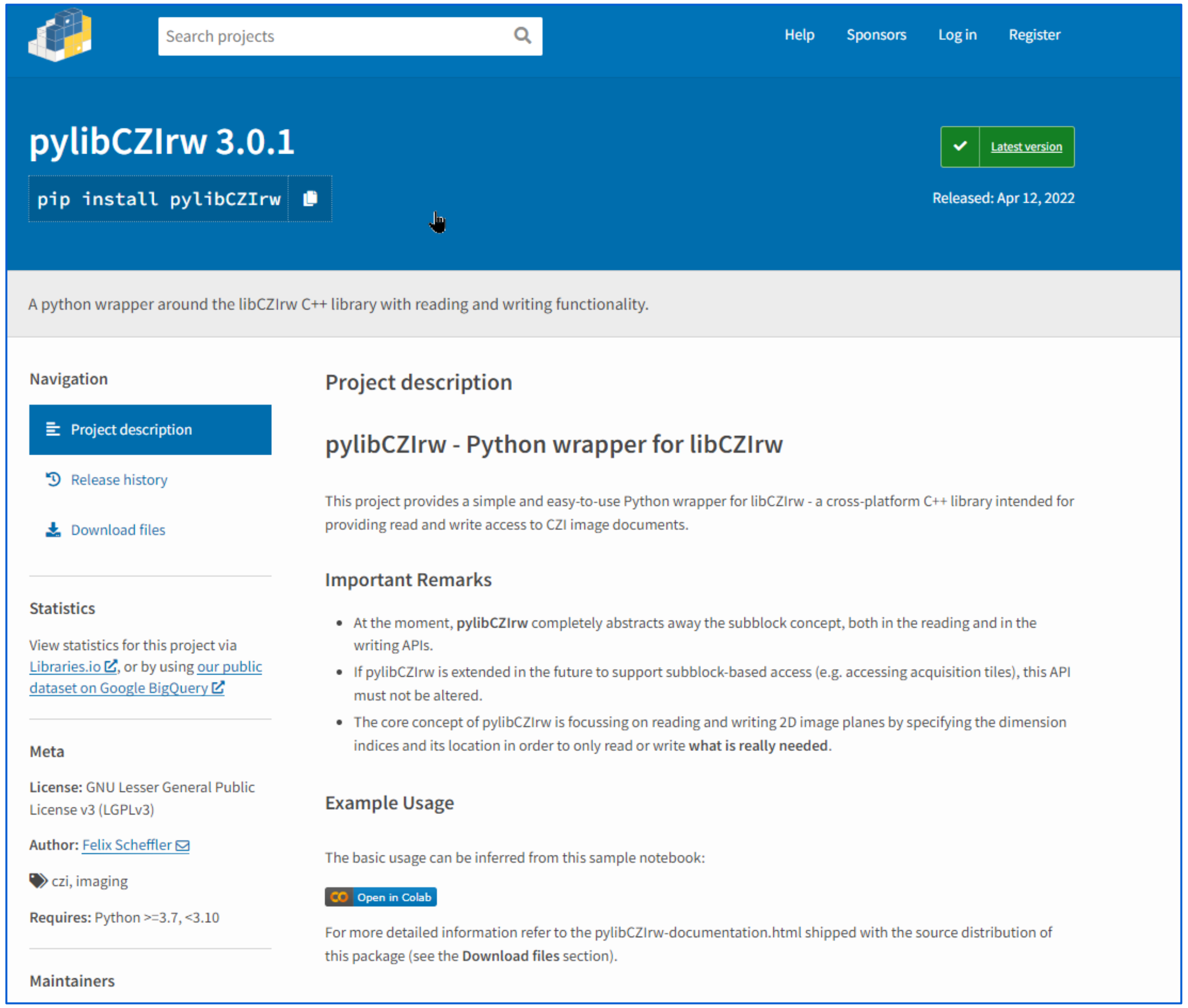
For single-class semantic segmentation: Open in Colab

For regression: Open in Colab

- Open and standardized “container” to store ML models and metadata
- is the “glue” between SW tools for Machine-Learning at ZEISS
- no new model format
- support for TF2.SavedModels (legacy) and ONNX models
- used by ZEN blue, ZEN core, APEER-ML and vision4D
- **allows external data scientists to integrate their own models our tools**

Open-Source python package pylibCZIrw

Read and write CZI images format in your python environment



pylibCZIrw 3.0.1

```
pip install pylibCZIrw
```

Released: Apr 12, 2022

A python wrapper around the libCZIrw C++ library with reading and writing functionality.

Statistics

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

Meta

License: GNU Lesser General Public License v3 (LGPLv3)

Author: Felix Scheffler

czi, imaging

Requires: Python >=3.7, <3.10

Maintainers

Project description

pylibCZIrw - Python wrapper for libCZIrw

This project provides a simple and easy-to-use Python wrapper for libCZIrw - a cross-platform C++ library intended for providing read and write access to CZI image documents.

Important Remarks

- At the moment, **pylibCZIrw** completely abstracts away the subblock concept, both in the reading and in the writing APIs.
- If pylibCZIrw is extended in the future to support subblock-based access (e.g. accessing acquisition tiles), this API must not be altered.
- The core concept of pylibCZIrw is focussing on reading and writing 2D image planes by specifying the dimension indices and its location in order to only read or write **what is really needed**.

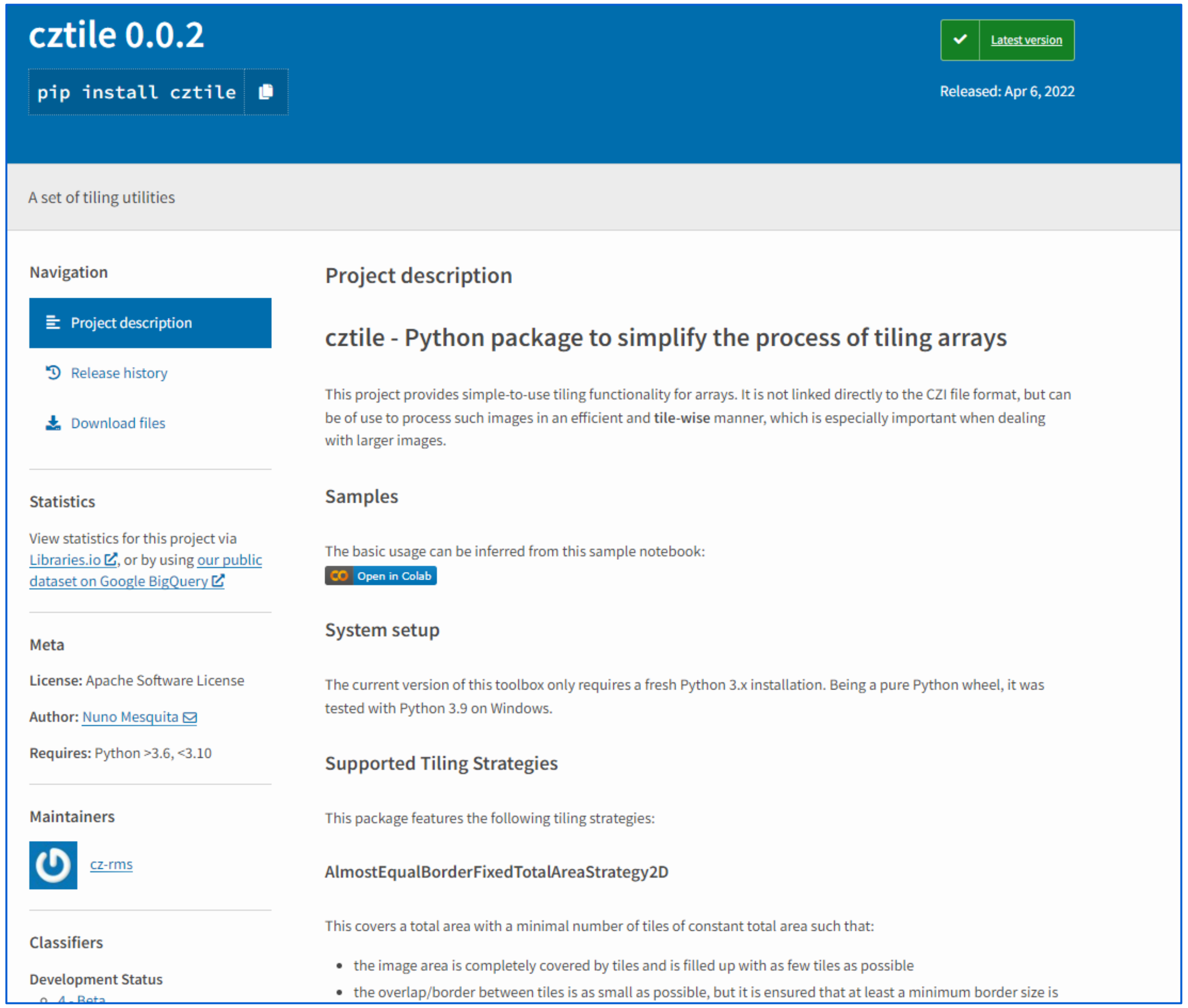
Example Usage

The basic usage can be inferred from this sample notebook:

For more detailed information refer to the pylibCZIrw-documentation.html shipped with the source distribution of this package (see the **Download files** section).

- easily read and write CZIs from Python using a simple API
- based on libCZIrw (C++) library (to be published soon)
- focus on the real application code and not an DataIO … ☺
- allows reading and writing parts of an CZI image
- ensure that the output works in ZEISS tools

Use a tiling method to process (big) arrays
Open-Source python package cztile



cztile 0.0.2

```
pip install cztile
```

Latest version

Released: Apr 6, 2022

A set of tiling utilities

Navigation

- Project description
- Release history
- Download files

Statistics

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

Meta

License: Apache Software License

Author: Nuno Mesquita

Requires: Python >3.6, <3.10

Maintainers

cz-rms

Classifiers

Development Status

- 4 - Beta

Project description

cztile - Python package to simplify the process of tiling arrays

This project provides simple-to-use tiling functionality for arrays. It is not linked directly to the CZI file format, but can be of use to process such images in an efficient and **tile-wise** manner, which is especially important when dealing with larger images.

Samples

The basic usage can be inferred from this sample notebook:

Open in Colab

System setup

The current version of this toolbox only requires a fresh Python 3.x installation. Being a pure Python wheel, it was tested with Python 3.9 on Windows.

Supported Tiling Strategies

This package features the following tiling strategies:

AlmostEqualBorderFixedTotalAreaStrategy2D

This covers a total area with a minimal number of tiles of constant total area such that:

- the image area is completely covered by tiles and is filled up with as few tiles as possible
- the overlap/border between tiles is as small as possible, but it is ensured that at least a minimum border size is

- General library to create tiles with overlap for an array
- Is not limited to CZI images or any specific format
- Can be directly used together with **pylibCZIrw** to read and write tiles
- focus on the real application code and not on re-inventing “tiling” over and over … ☺
- “unified” tiling algorithm gives consistent results

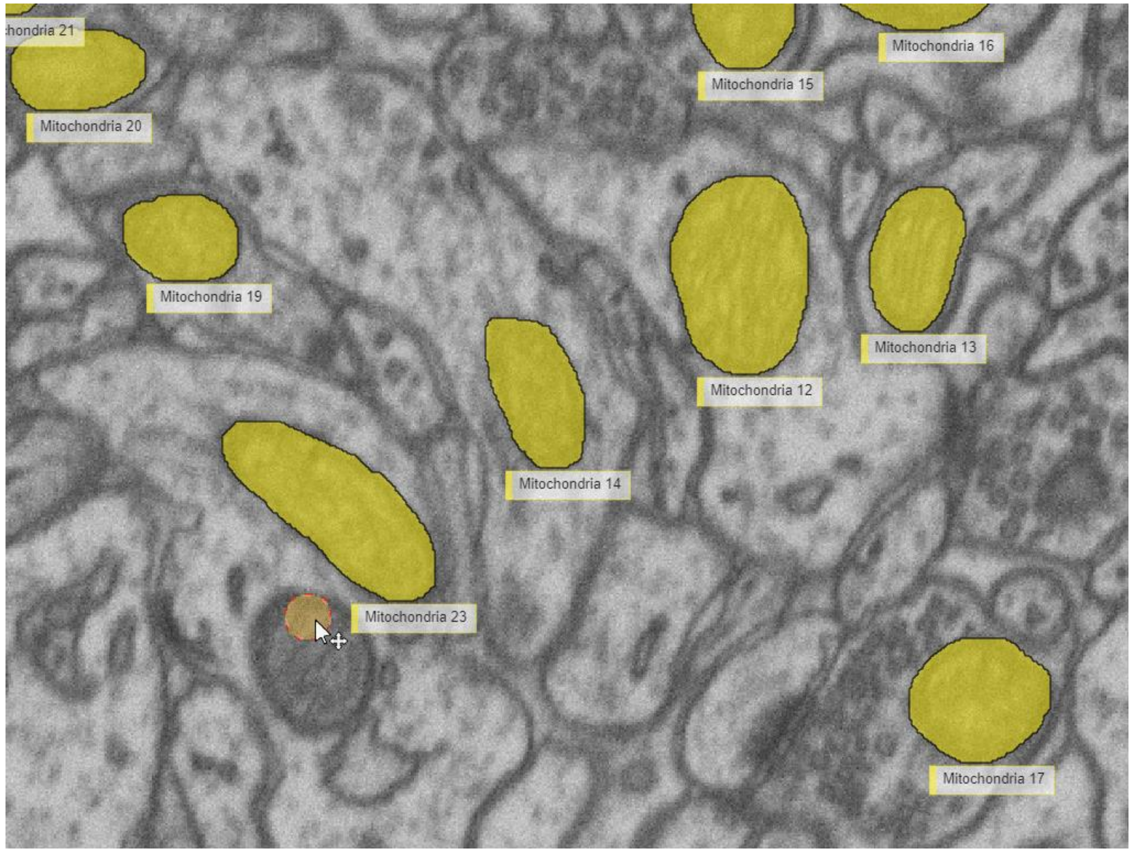
The ecosystem in action

What does the user see?

APEER-ML – Data-Driven Model Development

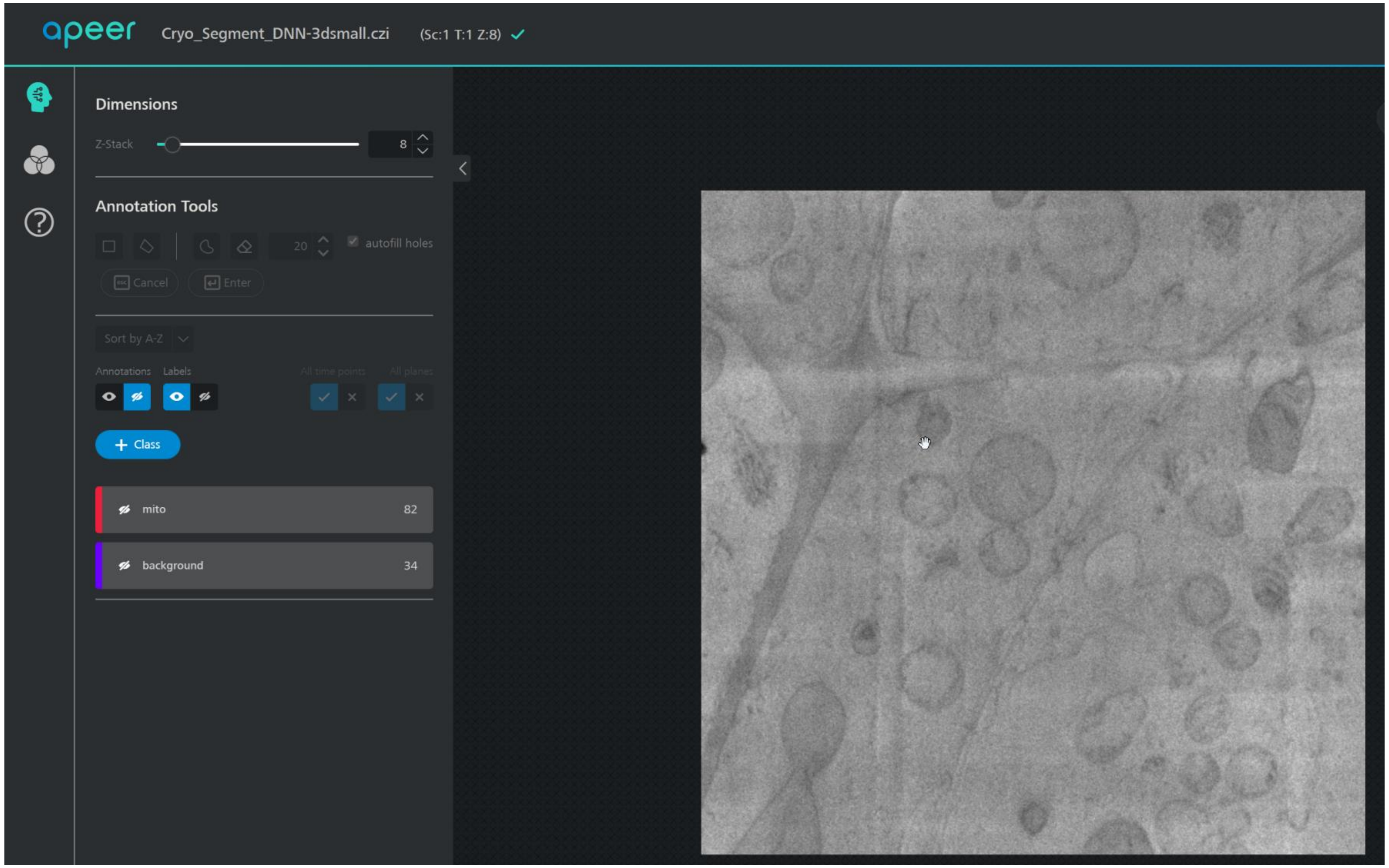
“Better Data beats better models”

Annotate on APEER → **Train** your specific U-net → **Download** the model and use in ZEN Intellesis Segmentation



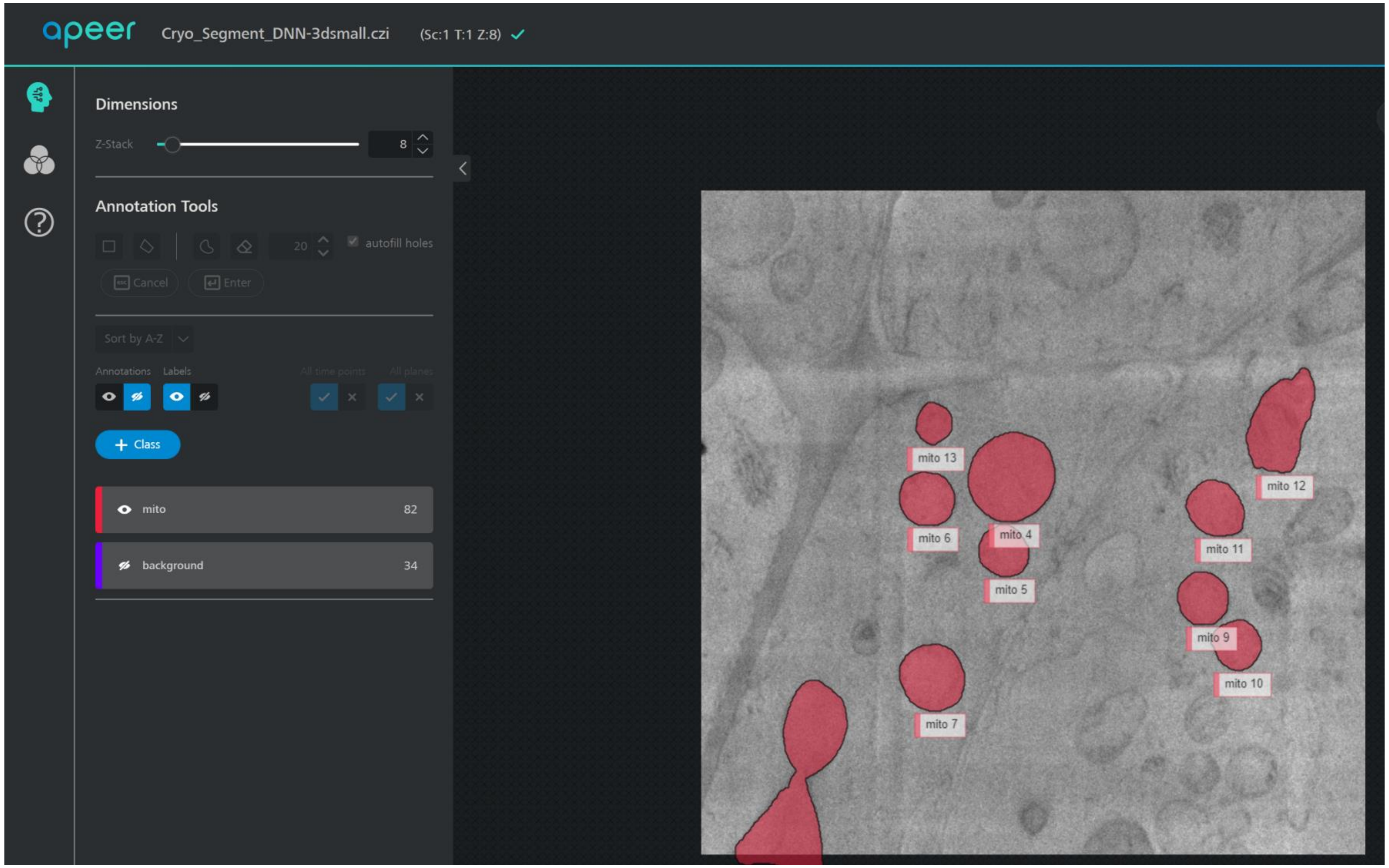
Partial Labeling is the key

Focus on areas where the model can "really learn new things"



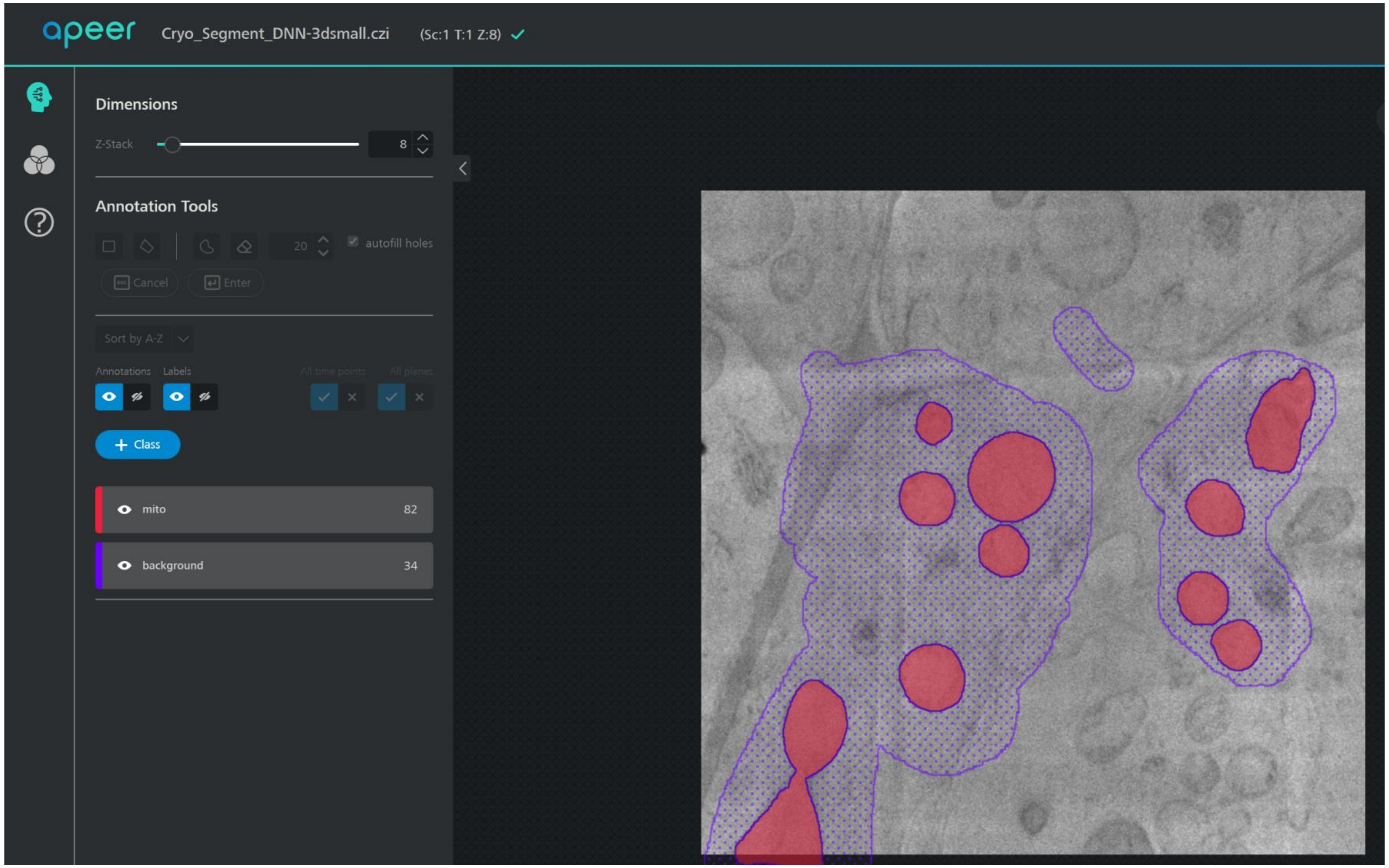
Partial Labeling is the key

Label a few objects – no need to label the complete frame



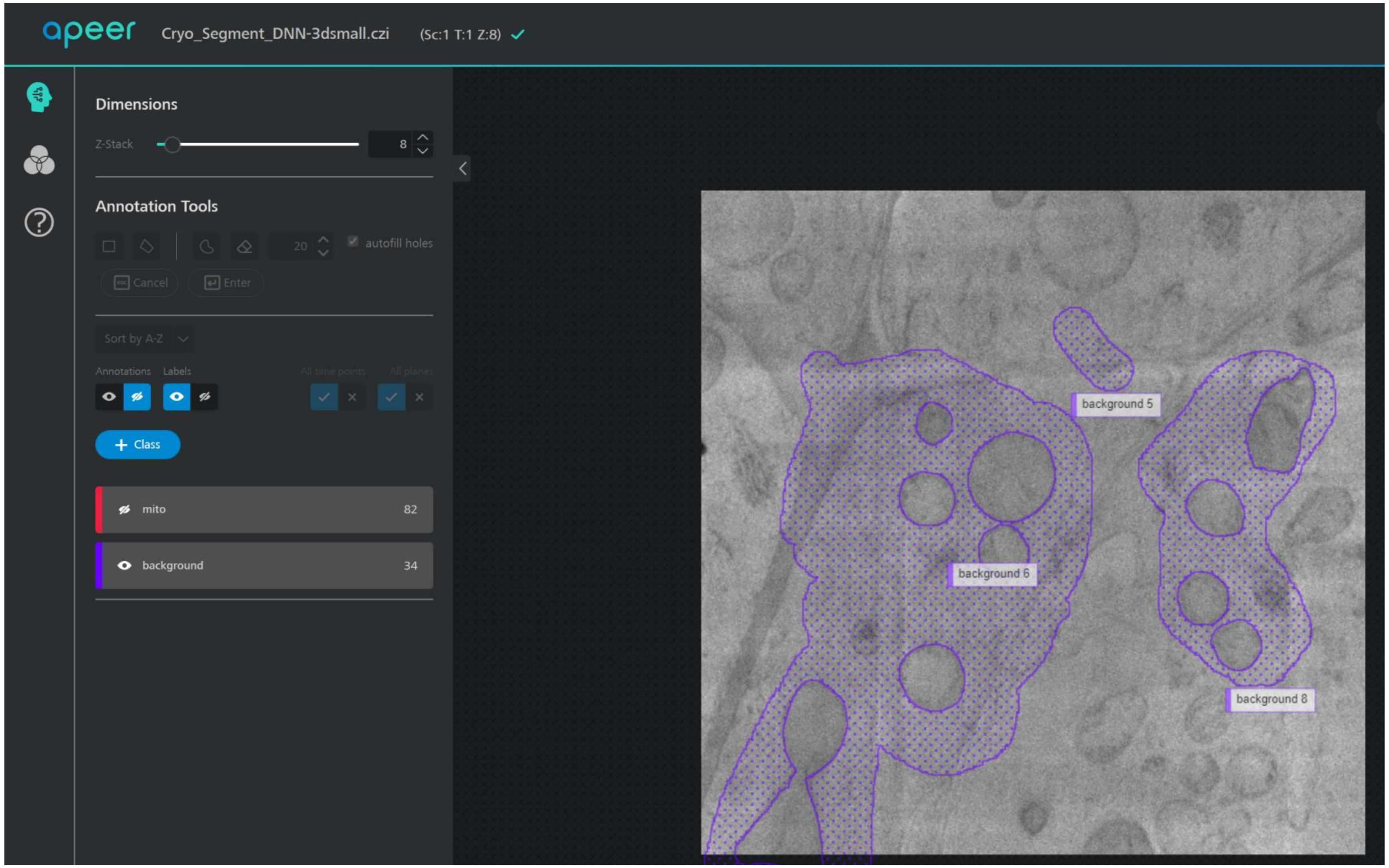
Partial Labeling is the key

Draw background around the objects



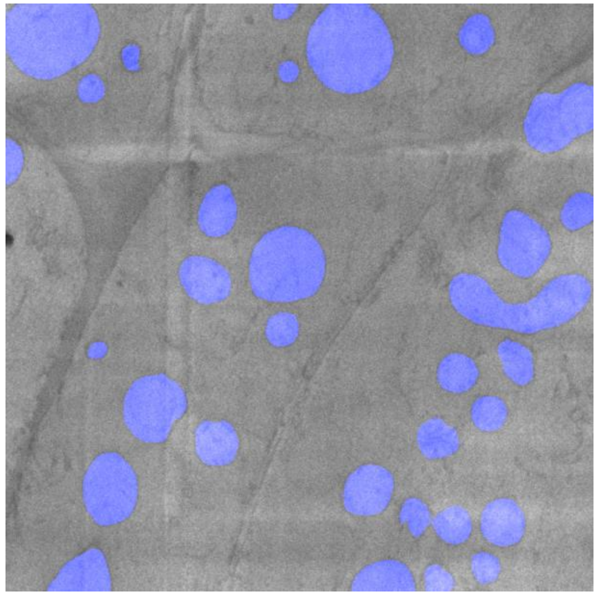
Partial Labeling is the key

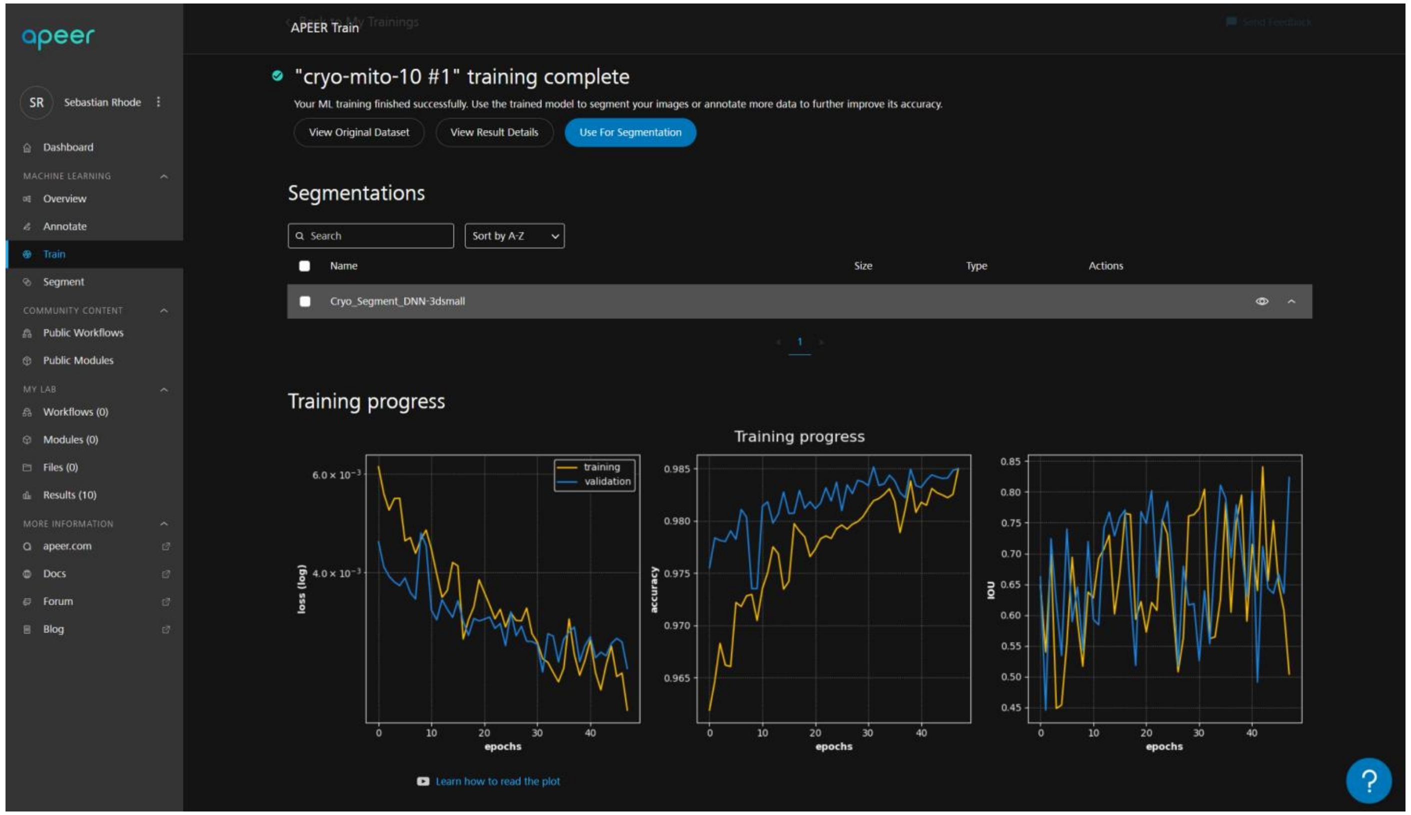
Automatic cutout – edges are labeled precisely



Make it easy to iterate a model with „better data" is key

- U-Net (EfficientNet) trained on APEER-ML platform using partial annotations
- **Fully automated training process (no coding required)**
- Import of trained model and segmentation in ZEN incl. integration into Image Analysis





"cryo-mito-10 #1" training complete

Your ML training finished successfully. Use the trained model to segment your images or annotate more data to further improve its accuracy.

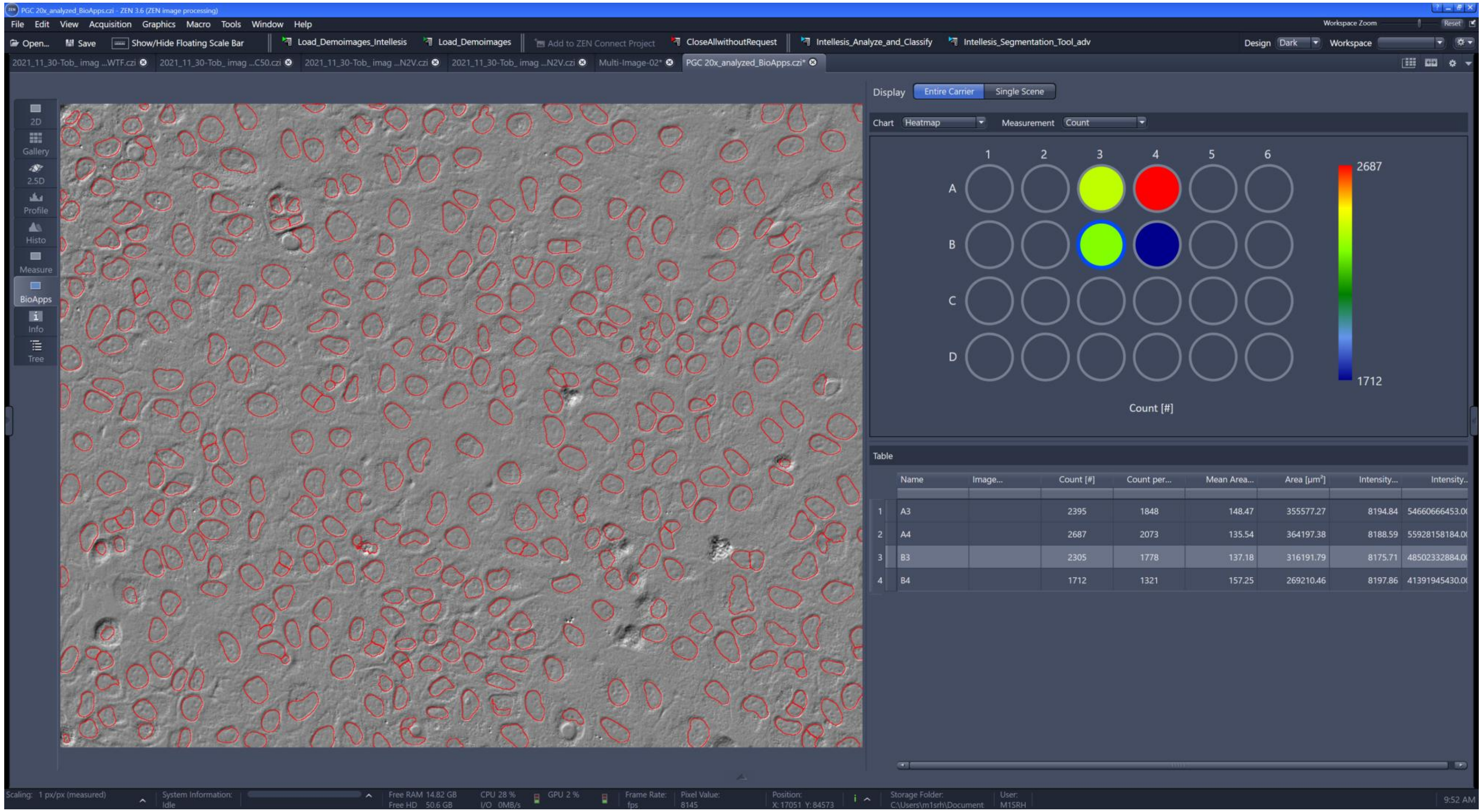
Segmentations

Name	Size	Type	Actions
Cryo_Segment_DNN-3dsmall			

Training progress

Application Examples – Life Science

Label-Free Segmentation inside a BioApp supercharged with a DNN


PGC 20x_analyzed_BioApps.czi - ZEN 3.6 (ZEN image processing)
File
Edit
View
Acquisition
Graphics
Macro
Tools
Window
Help
Open...
Save
Show/Hide Floating Scale Bar
Load_Demoimages_Intellesis
Load_Demoimages
Add to ZEN Connect Project
CloseAllwithoutRequest
Intellesis_Analyze_and_Classify
Intellesis_Segmentation_Tool_adv
2D
Gallery
2.5D
Profile
Histo
Measure
BioApps
Info
Tree
1
2
3
4
5
6
A
B
C
D
2687
1712
Count [#]

Design: Dark

Display: Entire Carrier

Chart: Heatmap

Measurement: Count

Table

	Name	Image...	Count [#]	Count per...	Mean Area...	Area [µm²]	Intensity...	Intensity...
1	A3		2395	1848	148.47	355577.27	8194.84	54660666453.0
2	A4		2687	2073	135.54	364197.38	8188.59	55928158184.0
3	B3		2305	1778	137.18	316191.79	8175.71	48502332884.0
4	B4		1712	1321	157.25	269210.46	8197.86	41391945430.0

Scaling: 1 px/px (measured)

System Information: Idle

Free RAM 14.82 GB

Free HD 50.6 GB

CPU 28 %

I/O 0MB/s

GPU 2 %

Frame Rate: fps

Pixel Value: 8145

Position: X: 17051 Y: 84573

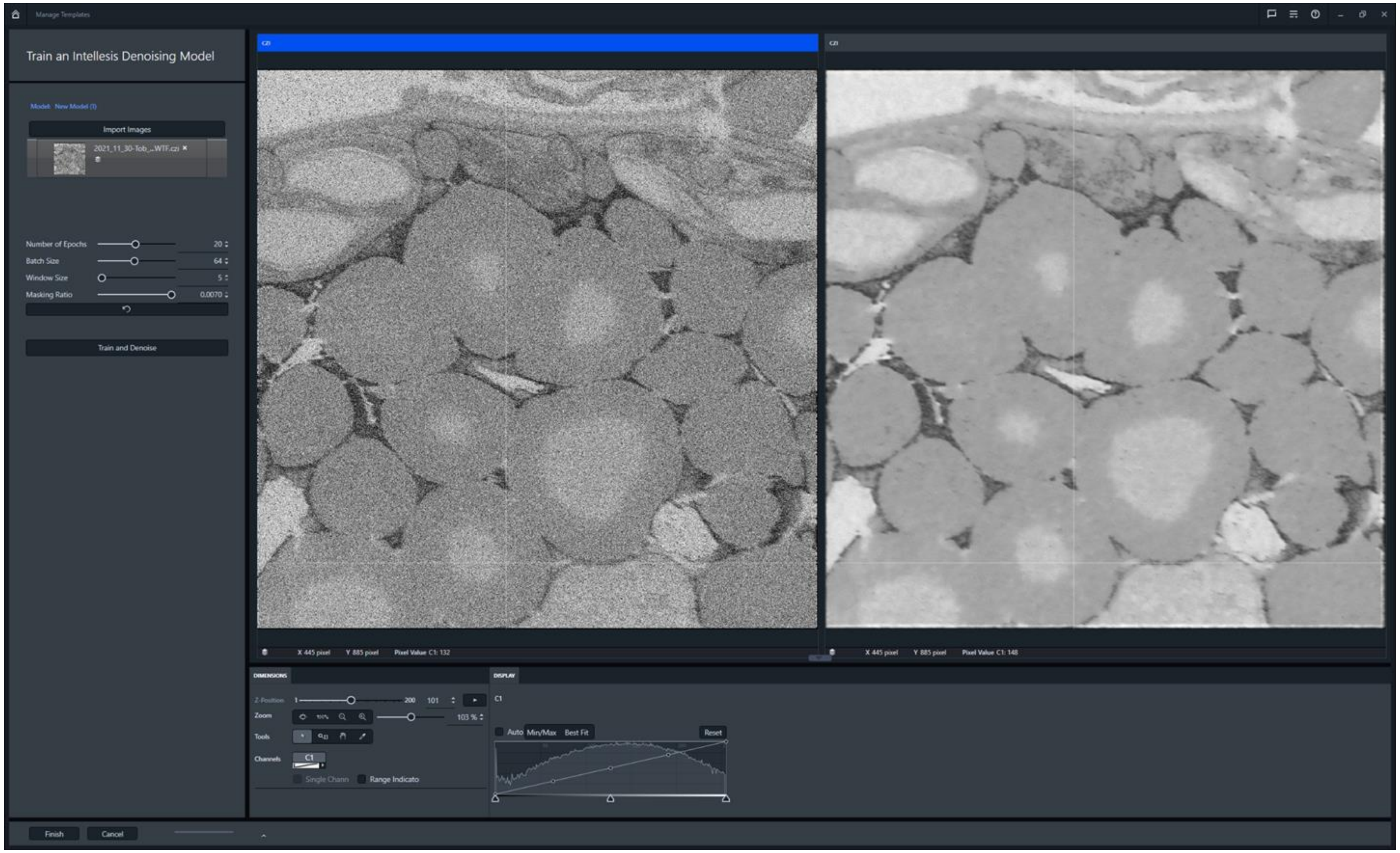
Storage Folder: C:\Users\m1srh\Document

User: M1SRH

9:52 AM

ZEN Intellesis Denoising based on Noise2Void

Simple training UI



Train an Intellesis Denoising Model

Model: New Model (1)

2021_11_30-Tob_...WTF.czi

Number of Epochs	20
Batch Size	64
Window Size	5
Masking Ratio	0.0070

X 445 pixel Y 885 pixel Pixel Value C1: 132

X 445 pixel Y 885 pixel Pixel Value C1: 148

Open ML Ecosystem - Summary

- Our ML ecosystem @ZEISS is build mainly on a python stack
- Many teams use and contribute to this stack
- A very good way to address the "alignment issues" is to incentivize the teams to create reusable python packages that make the life of others "easier"
- Our idea to keep our system as open as possible to make some of those packages public
- it "forces" ourselves @ZEISS to adhere to transparent software quality standards
- by staying "open" our ecosystem can also benefit more easily from new developments

Open Ecosystem for integrated Machine-Learning Workflows

https://www.zeiss.com/microscopy/int/website/landingpages/zen-intellesis.html

Get the trial license and try it out!

-- APEER --

https://www.apeer.com/app/machine-learning/overview

https://pypi.org/project/czmodel/

-- ZEISS GITHUB --

Additional Content on GitHub:

https://github.com/zeiss-microscopy/OAD/tree/master/Machine_Learning

We make it visible.