

ZEISS - Open software ecosystem for data-centric model development

Dr. Sebastian Rhode



Software Architect - AI Solutions
Staff Expert

Product Center for Software

19.10.2022



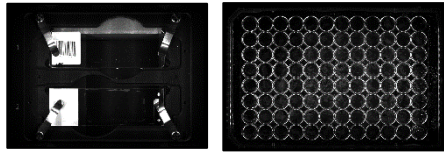
AI solutions @ZEISS microscopy



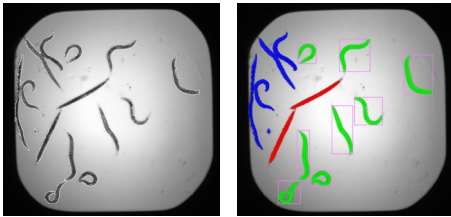
Image Analysis and Processing

Classification

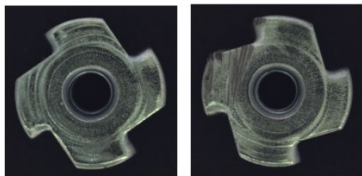
Classifying an entire image or individual objects



Recognize a Sample Carrier



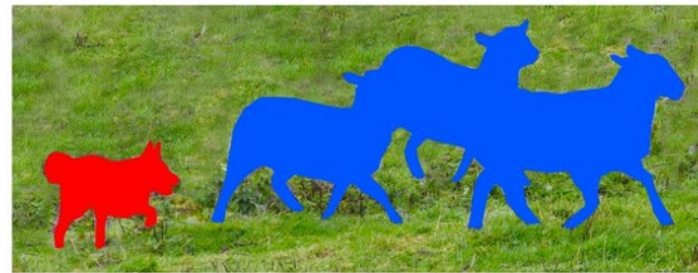
Classify objects in analyzed images



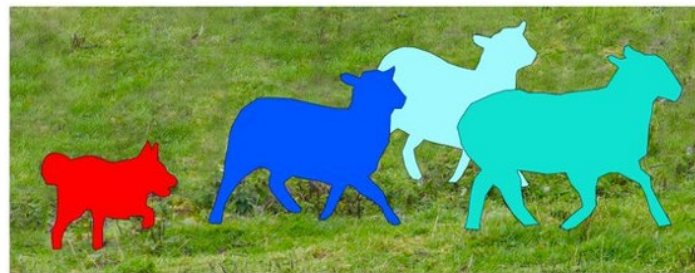
"Good" part vs "Anomaly"

Segmentation

Refers to classifying at a pixel level



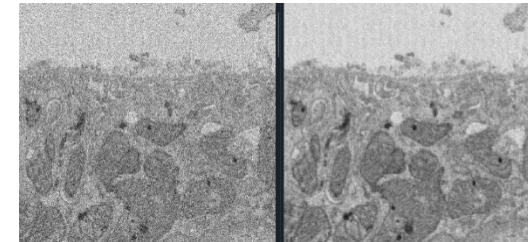
Semantic Segmentation



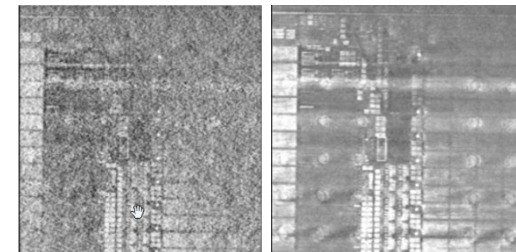
Instance Segmentation

Processing

Image corrections and enhancements



Denoising & Image2Image



Reconstruction



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

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

The **core message** when it comes to Image Analysis & AI solutions is:

“Better data beat better models”

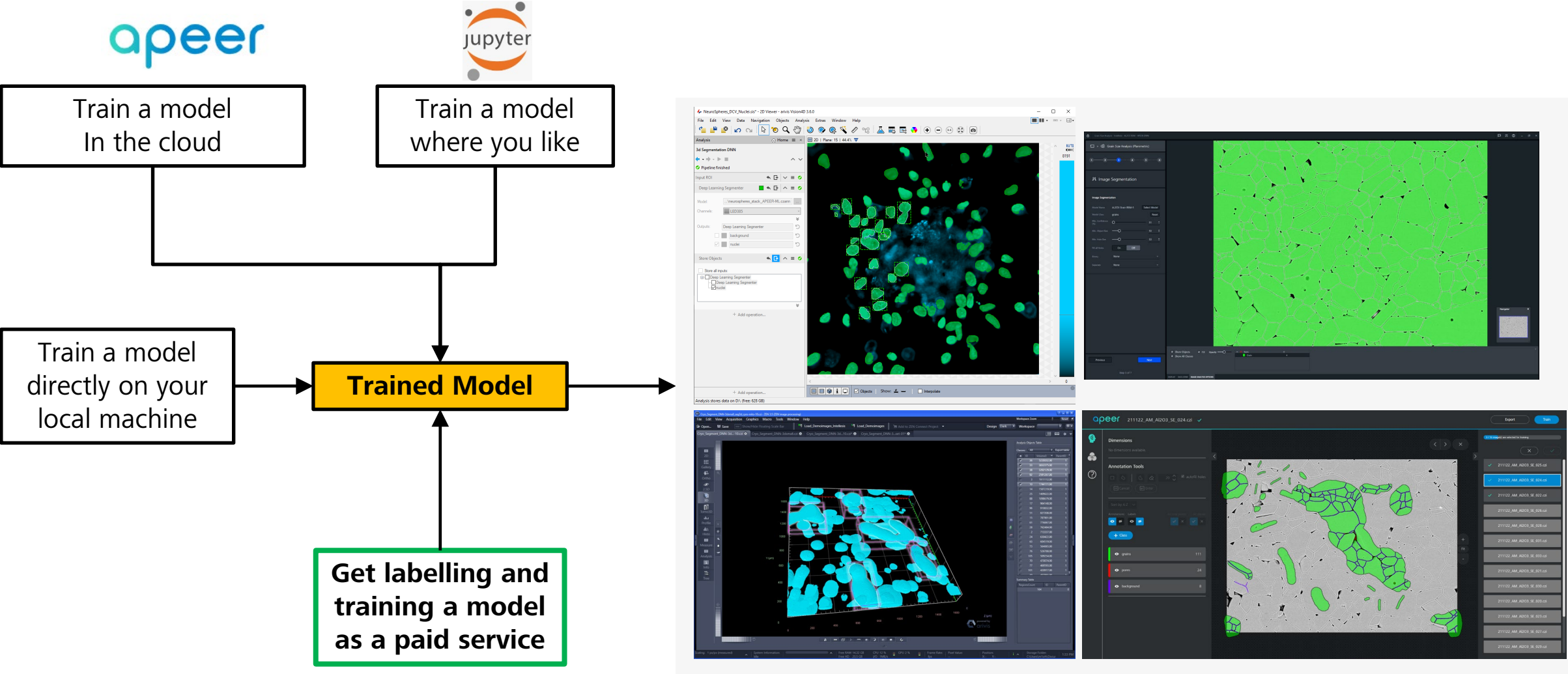
Align Requirements from Academia and Industry with Business Model



- 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 some users / customers do not want / are not allowed use cloud yet
- What tools should a we use? Should we develop our own (build or buy)



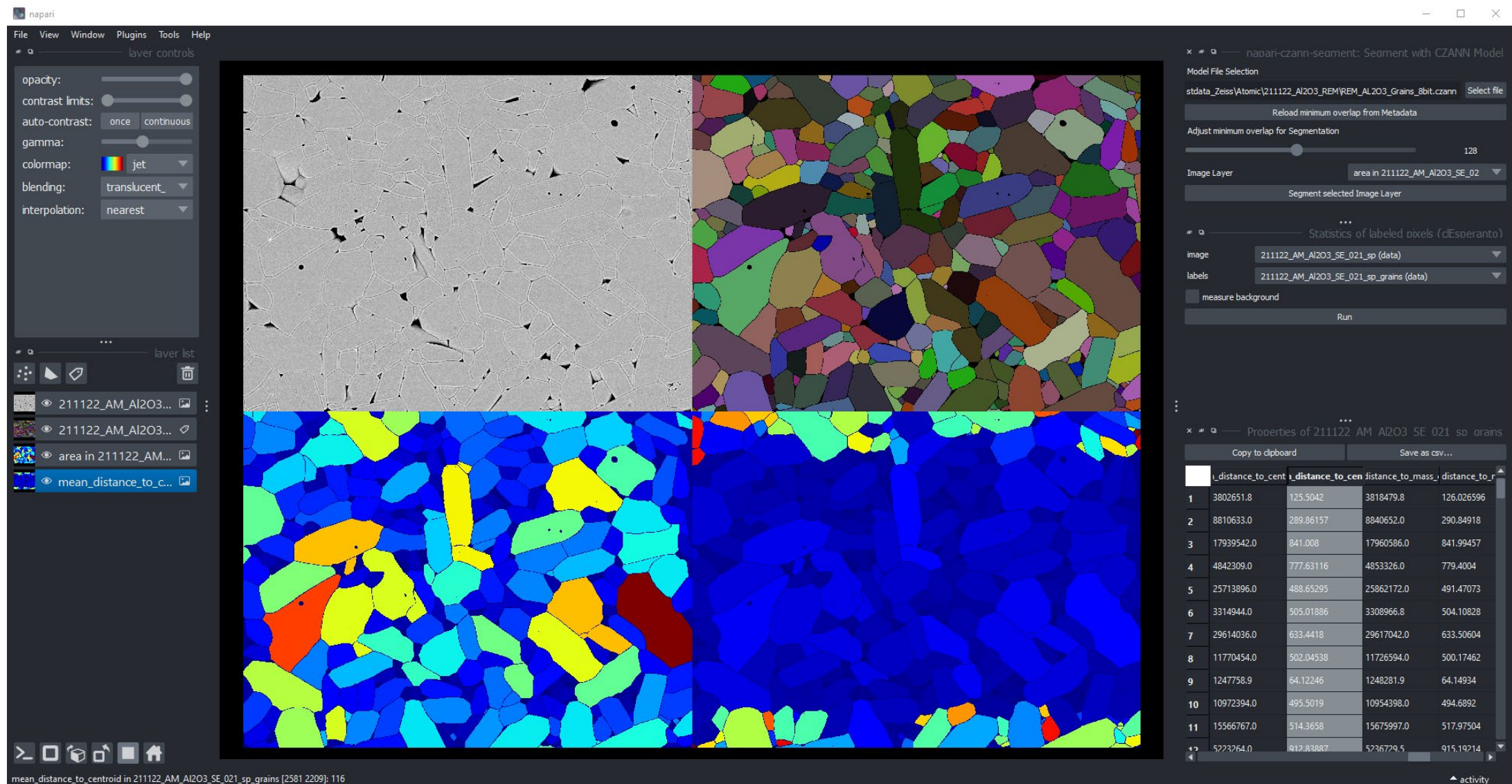
Concept: Train models “anywhere”



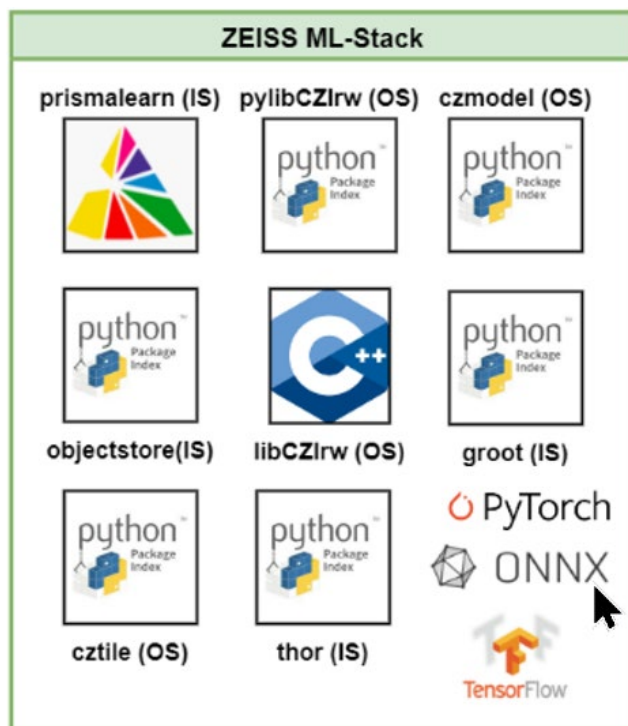
ZEN blue, ZEN core, vision4D and APEER

Open really means open – it also works inside “community software”

*.czann trained on APEER executed inside Napari



Our ecosystem - ZEISS Machine Learning Stack



ZEISS ML Stack

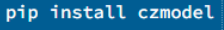
- mainly Python-based internal (IS) and open-source (OS) packages
- easy to deploy & use in different development teams @ZEISS
- clear rules that those packages must be the 1st choice when starting new projects
- test coverage and code-quality standards cannot be just be "nice thing to have" but are crucial



Open-Source python package czmodel

Store model along with metadata

czmodel 5.0.0

 `pip install czmodel`

Released: Sep 15, 2022

A conversion tool for TensorFlow or ONNX ANNs to CZANN

Navigation

- Project description
- Release history
- Download files

Project links

- ZEN Intellesis

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 >=3.7, <3.10

Maintainers

-  [cz-rms](#)

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
1.1.0	>= 3.5	>= 3.4
1.0.0	>= 3.5	>= 3.4
3.1.0 (legacy)	>= 3.4	>= 3.3
3.0.0 (legacy)	>= 3.2	>= 3.1

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

Structure of repo

This repo is divided into 3 separate packages -: core, tensorflow, pytorch.

- Core - Provides base functionality, no dependency on Tensorflow or Pytorch required.
- Tensorflow - Provides Tensorflow-specific functionalities, and converters based on Tensorflow-logics.
- PyTorch - Provides PyTorch-specific functionalities, and converters based on PyTorch-logics.

- 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 ONNX and TF2.SavedModels (legacy) models
- used by ZEN blue, ZEN core, APEER-ML and vision4D
- **allows external data scientists to integrate their own models in our tools**

Open-Source python package pylibCZIrw

Read and write CZI images format in your python environment



pylibCZIrw 3.2.0

`pip install pylibCZIrw`

Released: Aug 2, 2022

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

Navigation

- Project description
- Release history
- Download files

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:

[Open in Colab](#)

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

Installation

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, <4

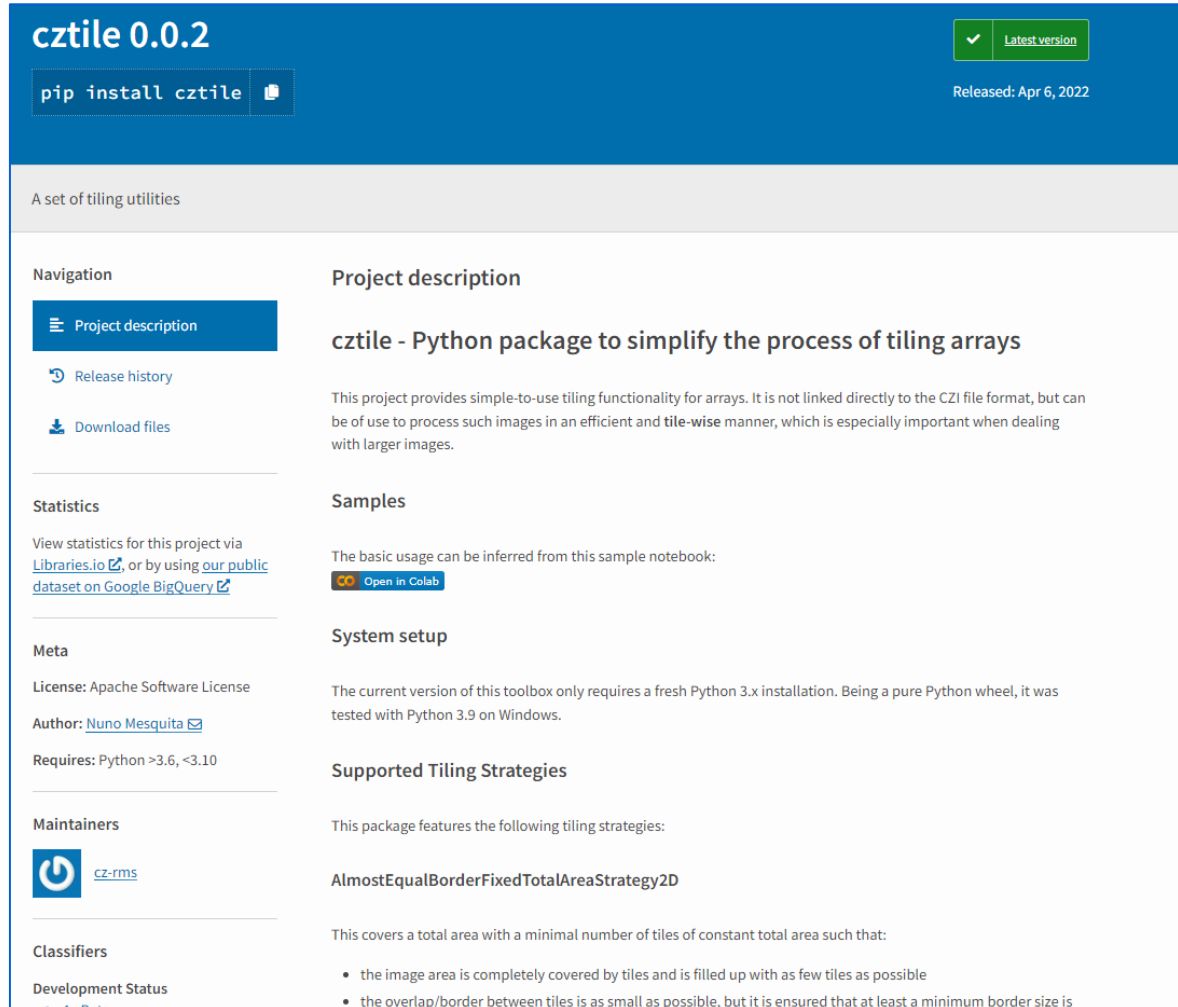
Maintainers

[cz-rms](#)


- easily read and write multi-dimensional image datasets as 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



The screenshot shows the PyPI page for the **cztile 0.0.2** package. The header is blue with the package name and version, a green 'Latest version' badge, and a 'pip install cztile' button. Below the header, a grey bar states 'A set of tiling utilities'. The main content area is divided into a left sidebar and a right main section. The sidebar contains navigation links (Project description, Release history, Download files), statistics, meta information (License: Apache Software License, Author: Nuno Mesquita, Requires: Python >3.6, <3.10), maintainers (cz-rms), and classifiers. The main section contains the project description, samples, system setup, supported tiling strategies, and development status.

cztile 0.0.2 ✓ Latest version
`pip install cztile`  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

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 identical tiles with overlap for an array
- not limited to CZI images or any specific format
- Can be directly used together with **pylibCZlrw** to read and write tiles
- focus on application code and not on re-inventing “tiling” over and over ...
- “unified” tiling algorithm gives consistent results

APEER-ML – Data-Driven Model Development

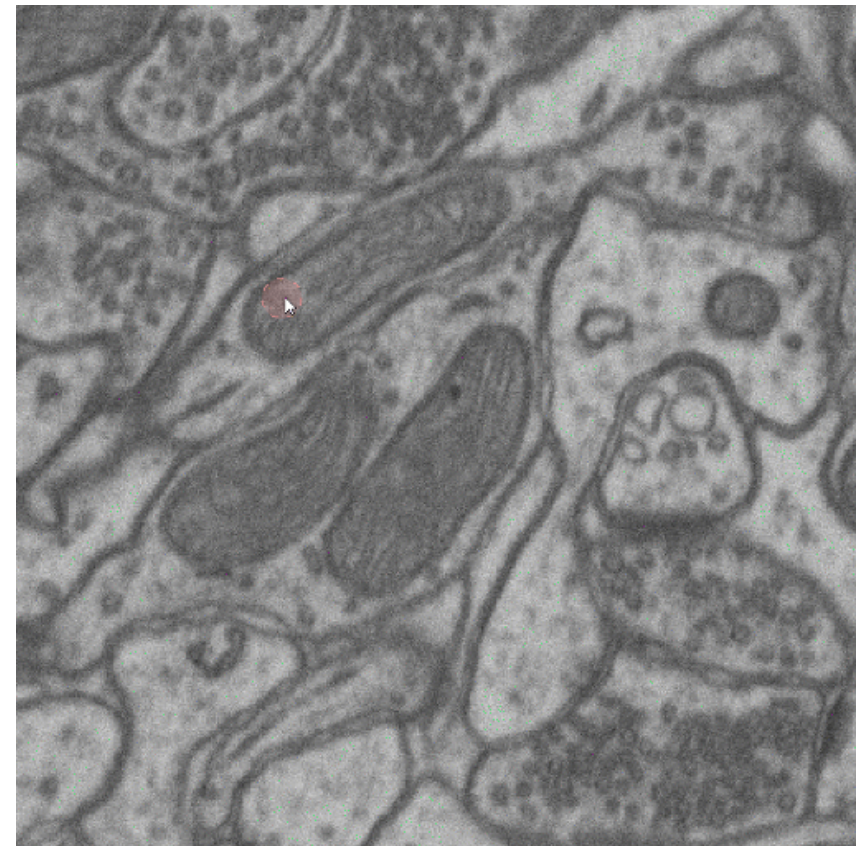
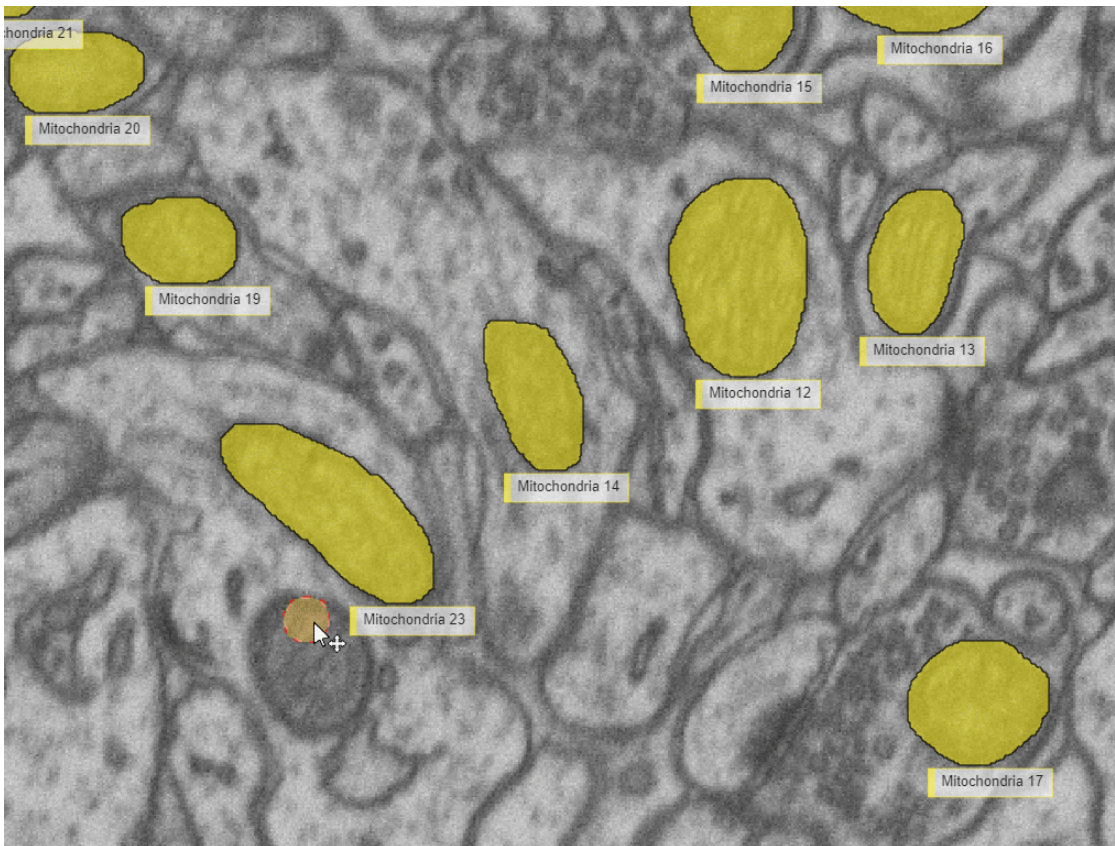
“Better Data beats better models”



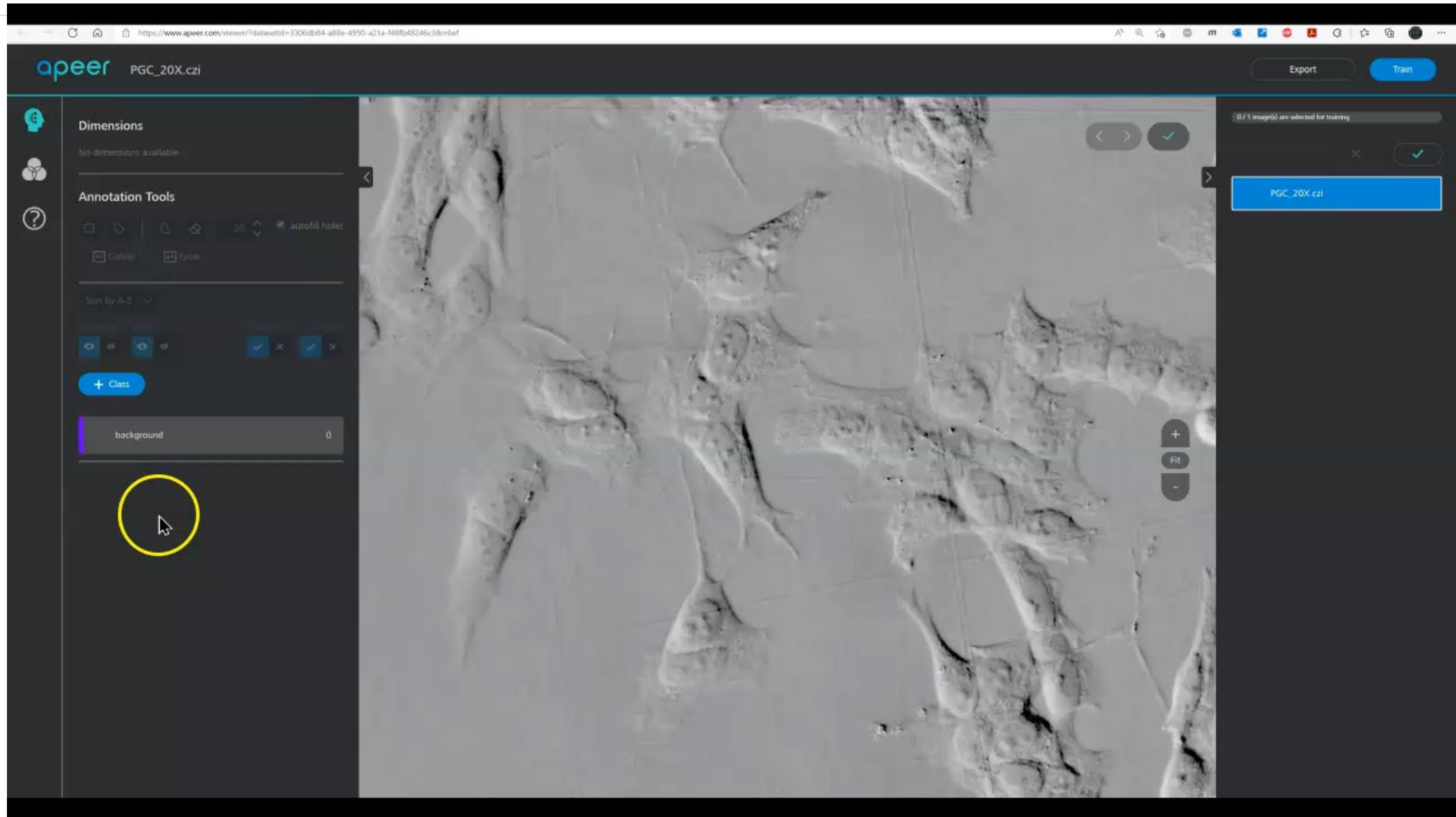
Annotate on APEER

Train your specific U-net

Download the model and use
in ZEN or V4D or Napari or ...



The ecosystem in action



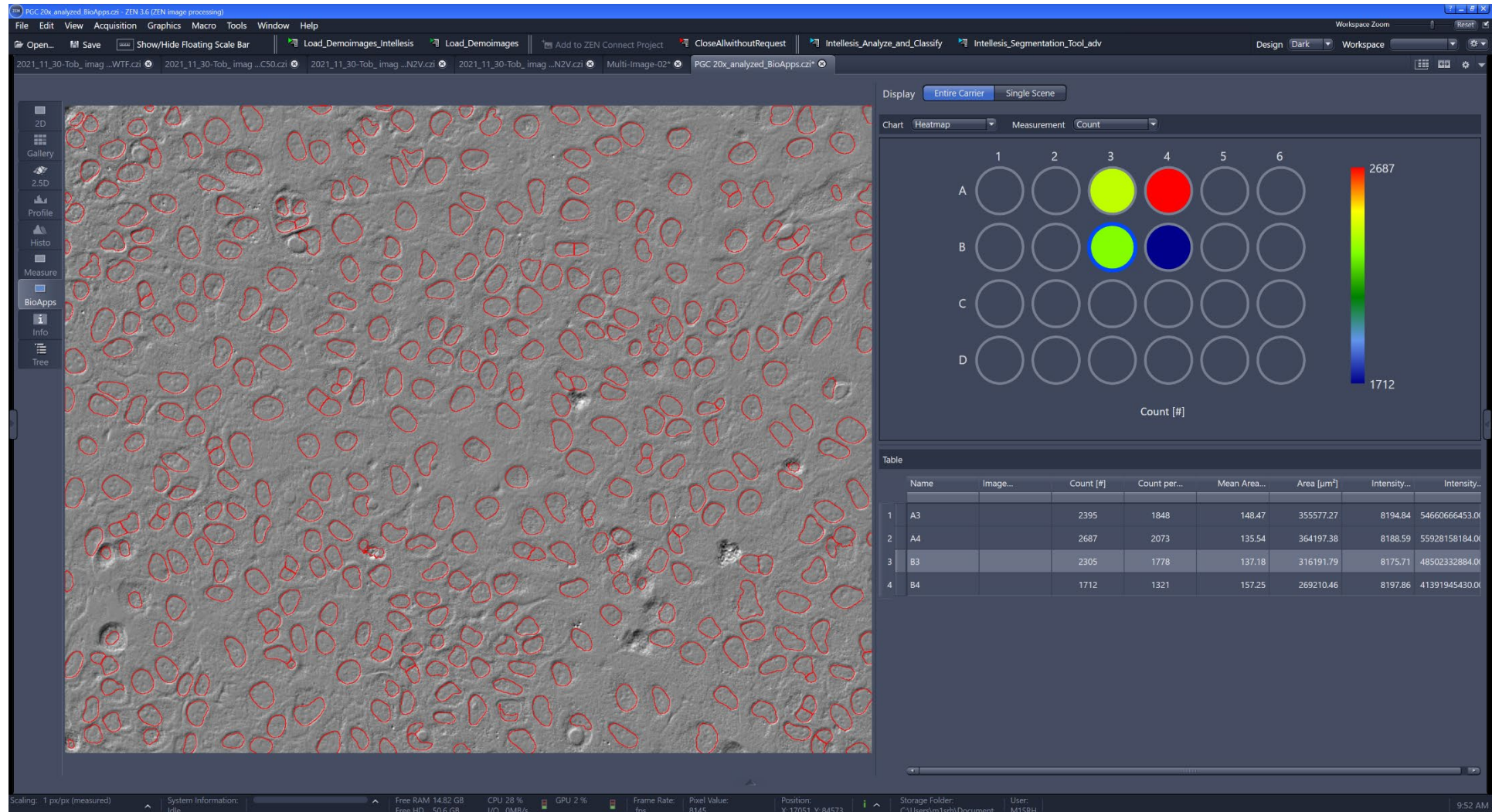
Fully automated Training pipeline

Download Model or continue training



Application Examples – Life Science

Label-Free Segmentation inside a ZEN BioApp



Python Stuff and APEER

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

<https://pypi.org/project/pylibCZlrw/>

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

<https://pypi.org/project/cztile/>

<https://www.napari-hub.org/plugins/napari-czann-segment> (experimental)

Additional Content

<https://www.zeiss.com/microscopy/int/website/landingpages/zen-intellessis.html>

[https://github.com/zeiss-microscopy/OAD/tree/master/Machine Learning](https://github.com/zeiss-microscopy/OAD/tree/master/Machine_Learning)

<https://github.com/ZEISS/libczi>



We make it visible.