ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

CAiM Computer-assisted Applications in Medicine

MiNTiF is Networks Tool in Fiji: Documentation of the Plugin

Luca Widmer, Alvaro Gomariz, Orcun Goksel
Computer-assisted Applications in Medicine, Computer Vision Lab, ETH Zurich, Switzerland

2021
Technical Report

Contents

Setup

1.1 GitHub Repository

The MiNTiF Github Repository can be found here .

1.2 Videos

A collection of example video walk-troughs and can be found here.

1.3 Installation

1. Download and install Fiji.
2. Download the MiNTiF git repository and place the folder `"MiNTiF"` in the Fiji sub-folder:

   ```
   {Userspecific}\Fiji.app\plugins\Scripts\Plugins
   ```

3. Place the folder `"MiNTiF_Utils"` in the Fiji sub-folder:

   ```
   {Userspecific}\Fiji.app
   ```

4. Before you use MiNTiF please install & update Anaconda.
5. Restart Fiji. Find the MiNTiF Plugin in Fiji under Plugins >MiNTiF.
   Install the necessary packages by running the command
   MiNTiF >Utilities >Install Environment

Introduction

MiNTiF ("MiNTiF is Networks Tool in Fiji") aims to provide user friendly way to train TensorFlow models and predict on new data using these models. It is based on work by Gomariz et al.[1] and was developed as a Master Thesis by Luca Widmer. It assumes users have no previous experience with Deep Neural Network methods and aims to provide a simple but flexible way to apply these methods to images from within FIJI. Currently MiNTiF models have been implemented for semantic segmentation and detection. MiNTiF is built on a Python framework that is used to call TensorFlow libraries.

The MiNTiF plugin is made up of four main Modules described below. These Modules provide functionality for users to setup, train and deploy DNN models without needing to leave the Fiji User interface.

Users provide images containing marker (input) channels and label (output) channels or coordinate files. For example, marker channels might be fluorescence microscopy channels and labels might be an segmentation mask of a certain tissue. MiNTiF converts these images into a custom datafile that implements a tiling strategy to make it possible to analyze large images. These datafiles can subsequently be used to train a TensorFlow model on the user's machine. The aim of this training is to learn a way to predict the label channels from the marker channels.

Once a model is trained, it can be used to predict labels on previously unseen data using the same or a subset of the same marker channels. After prediction the datafile can then be reconstructed back into an image that can be loaded in ImageJ (Fig. 2.1).

[1] Alvaro Gomariz, Tiziano Portenier, Patrick M Helbling, Stephan Isringhausen, Ute Suessbier, César Nombela-Arrieta, and Orcun Goksel. Modality attention and sampling enables deep learning with heterogeneous marker combinations in fluorescence microscopy. *arXiv preprint arXiv:2008.12380*, 2020

2.1 Summary

- **Define Model:** Here you will be able to define the parameters necessary to construct one of the predefined Convolutional Neural Networks (CNN) . The parameters are saved as a model.json file that is required for the further steps.
- **Crate Dataset:** With this module you can create a new custom MiNTiF dataset from an Image currently loaded in Fiji, or you can append the Image to an already existing MiNTiF file. These files will be used to train models and predict the labels of the images.
- **Deploy Model:** This module will let you train a model on training datasets or let you predict the labels of test datasets.
  - **Training:** Train a model to accomplish a task (e.g. segmentation) by providing annotated data and letting the network determine the relevant features.
  - **Prediction:** Predict labels on new datasets. The network will use the trained parameters from the previous step to predict labels on previously unseen data.
- **Reconstruct:** This module will let you reconstruct the image and labels from MiNTiF Files created within this module after prediction. The reconstructed Images will contain the predicted labels as separate image channels.

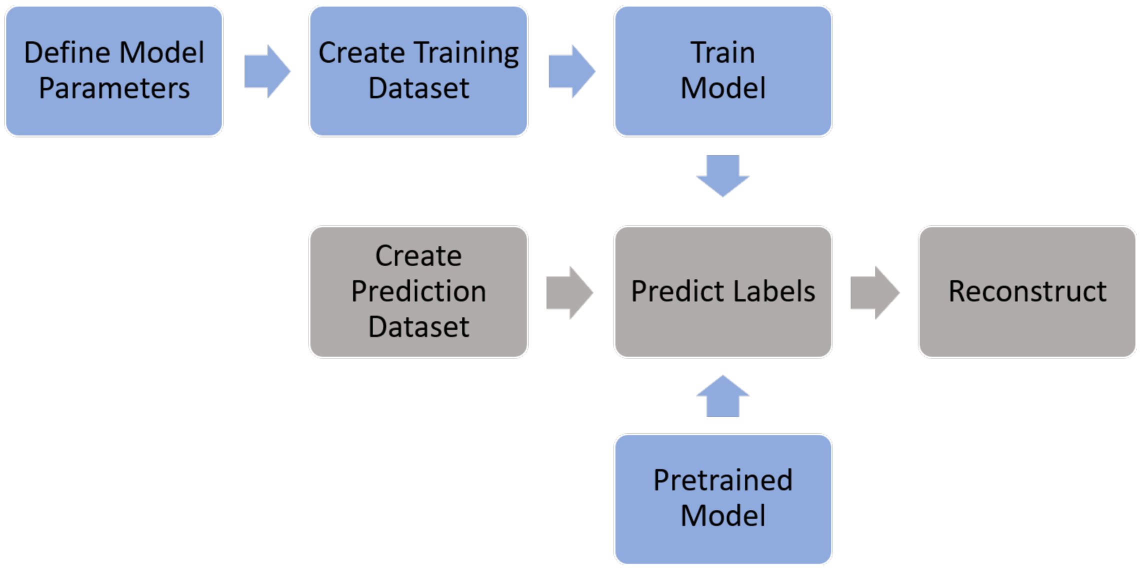
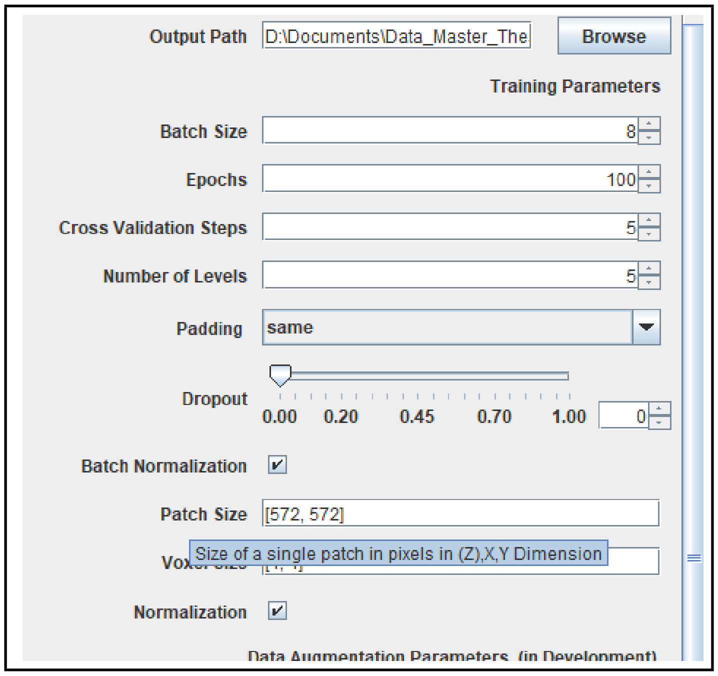


Figure 2.1: **Machine Learning Workflow**: For most basic machine learning applications, a model needs to be defined (for example using TensorFlow) and trained with training data. The training data contains the inputs and training targets (or labels) for the classifier. After a Model has been sufficiently trained, this trained model can be applied to predict on new data. The initial preparation of Datasets oftentimes includes some prepossessing and is therefore included as a separate step.

Example of Application

This section will give a short walk through of the MiNTiF plugin. Screenshots will show step by step what the user should see. All of these Steps are also documented as videos. Parameters as well as a more details on the pipeline are explained in detail in the Chapter **Modules** and there is an **Troubleshooting & FAQ** section at the end of the document.

1. **Define a Model:** To define a model you will need to open Plugins >MiNTiF>Create Model Here you can enter all values necessary to define the model. If you hover over any filed name, a short description will pop up. Again, you can find more details on all parameters in chapter 4.1.



Output Path D:\Documents\Data_Master_The

Training Parameters

Batch Size 8

Epochs 100

Cross Validation Steps 5

Number of Levels 5

Padding same

Dropout 0

Batch Normalization ✓

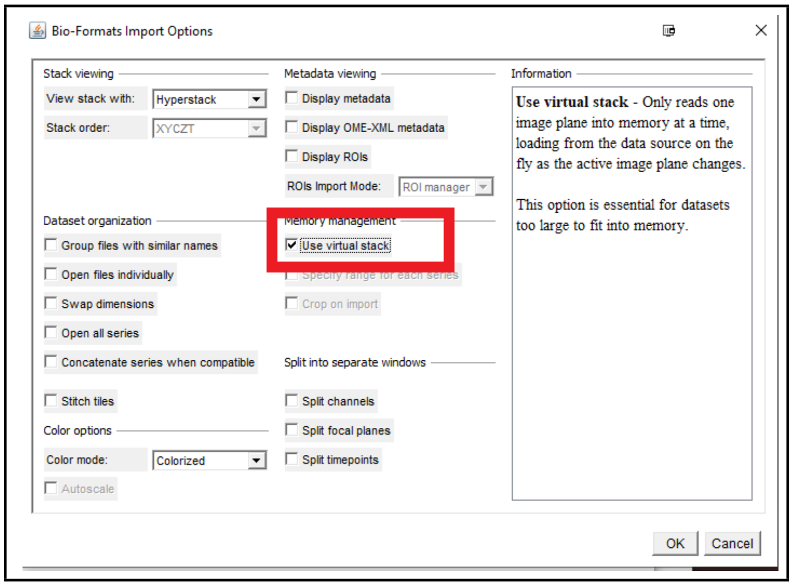
Patch Size [572, 572]

Size of a single patch in pixels in (Z),X,Y Dimension

Normalization ✓

Data Augmentation Parameters (in Development)

2. **Load an image into ImageJ**: Here we use an imaris file as an example, other formats may need different import modules. If your image can be loaded in Fiji, It is recommended you do not select this option, as the conversion will be faster. If you cannot load the image, since it is to large for the RAM of your computer, select "Use Virtual stack". BIO-Formats Importer:


Bio-Formats Import Options

Stack viewing

View stack with: Hyperstack

Stack order: XYCZT

Metadata viewing

☐ Display metadata

☐ Display OME-XML metadata

☐ Display ROIs

ROIs Import Mode: ROI manager

Information

Use virtual stack - Only reads one image plane into memory at a time, loading from the data source on the fly as the active image plane changes.

This option is essential for datasets too large to fit into memory.

Dataset organization

☐ Group files with similar names

☐ Open files individually

☐ Swap dimensions

☐ Open all series

☐ Concatenate series when compatible

☐ Stitch tiles

Memory management

☑ Use virtual stack

☐ Specify range for each series

☐ Crop on import

Split into separate windows

☐ Split channels

☐ Split focal planes

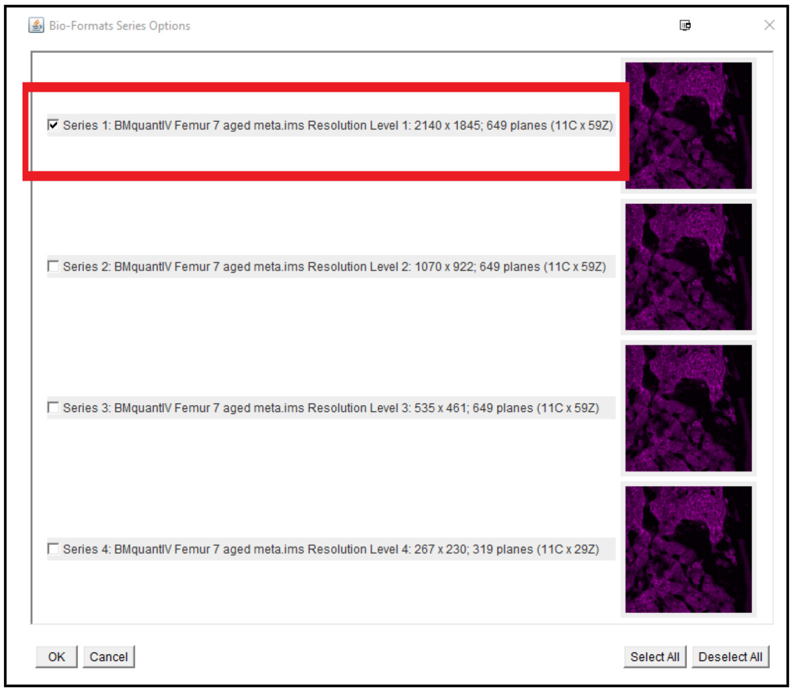
☐ Split timepoints

Color options

Color mode: Colorized

☐ Autoscale

Note: If your image file contains multiple resolution levels, load only the highest resolution level.


Bio-Formats Series Options

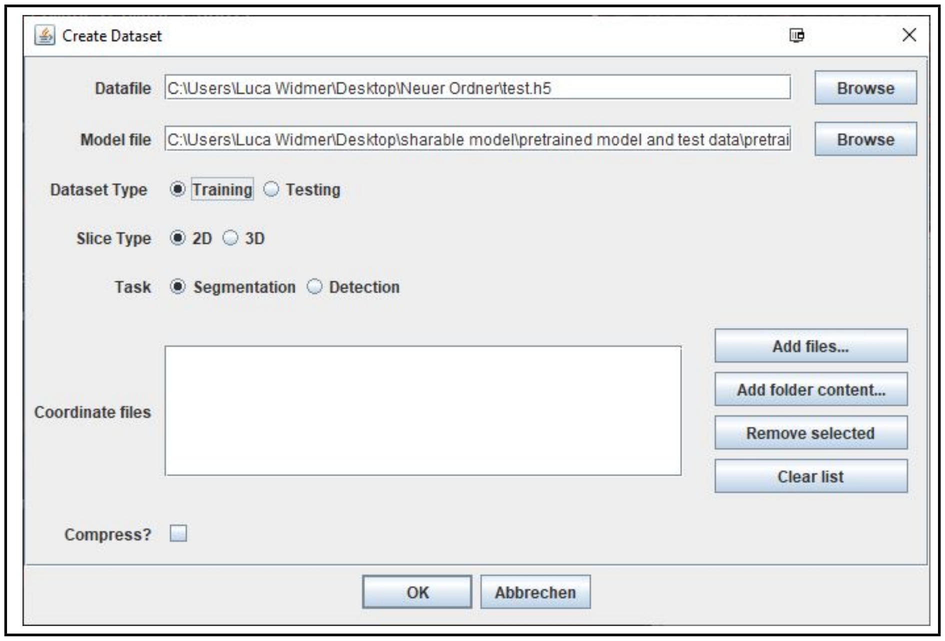
☑ Series 1: BMquantIV Femur 7 aged meta.ims Resolution Level 1: 2140 x 1845; 649 planes (11C x 59Z)

☐ Series 2: BMquantIV Femur 7 aged meta.ims Resolution Level 2: 1070 x 922; 649 planes (11C x 59Z)

☐ Series 3: BMquantIV Femur 7 aged meta.ims Resolution Level 3: 535 x 461; 649 planes (11C x 59Z)

☐ Series 4: BMquantIV Femur 7 aged meta.ims Resolution Level 4: 267 x 230; 319 planes (11C x 29Z)

3. **Create A Training Dataset:** Call Plugins >MiNTiF >Create Dataset and enter the information about the dataset. Make sure you select “Training” as Dataset Type:



Create Dataset

Datafile: C:\Users\Luca Widmer\Desktop\Neuer Ordner\test.h5

Model file: C:\Users\Luca Widmer\Desktop\sharable model\pretrained model and test data\pretrai

Dataset Type: [✓ Training] Testing

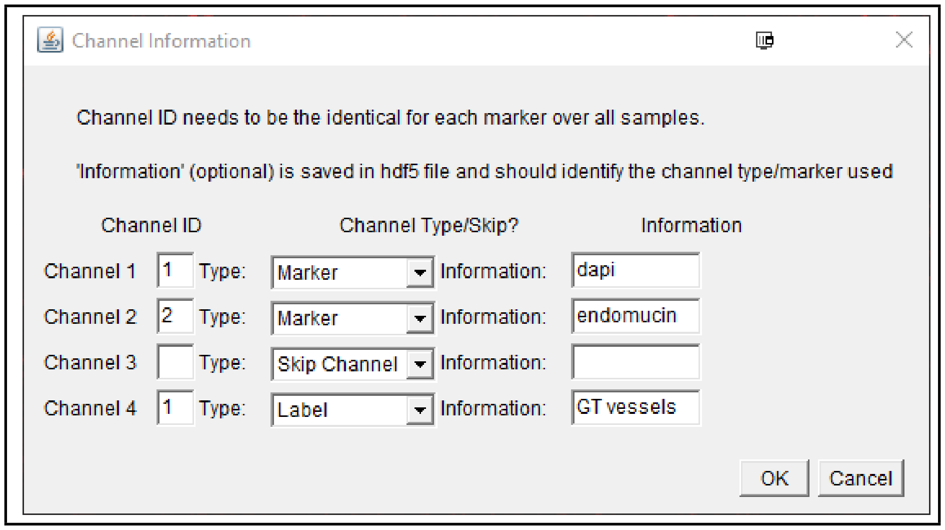
Slice Type: [✓ 2D] 3D

Task: [✓ Segmentation] Detection

Coordinate files

Compress? ☐

Next, enter information about each channel:


Channel Information

Channel ID needs to be the identical for each marker over all samples.

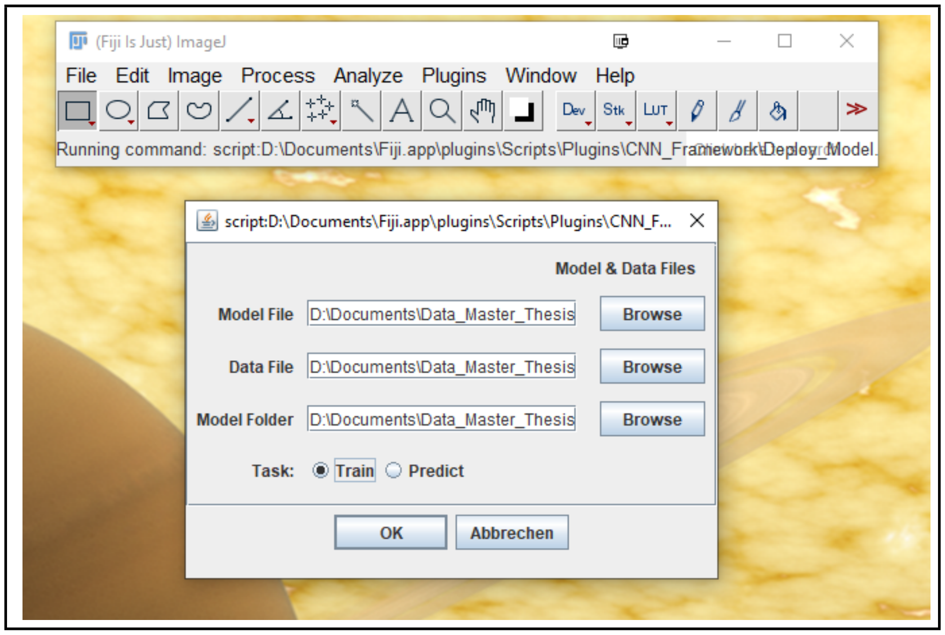
'Information' (optional) is saved in hdf5 file and should identify the channel type/marker used

	Channel ID	Channel Type/Skip?	Information
Channel 1	1	Type: Marker	Information: dapi
Channel 2	2	Type: Marker	Information: endomucin
Channel 3		Type: Skip Channel	Information:
Channel 4	1	Type: Label	Information: GT vessels

After you click OK, MiNTiF will convert the image to an MiNTiF data file at the path you provided. This can take multiple minutes, the plugin will output "done!" when it is finished. **Repeat this step with at least two more datasets and append them to the same file.**

At this point you should have a model.json file, as well as a single MiNTiF datafile (.h5 file-ending), containing multiple images.

4. **Train The Model:** Call Plugins >MiNTiF >Deploy Model and choose the Task "Train". Training is the most time intensive step and can take many hours. You can supervise the training in Tensorboard (Chapter 4.3.5).



Running command: script:D:\Documents\Fiji.app\plugins\Scripts\Plugins\CNN_Framework\Deploy_Model.

script:D:\Documents\Fiji.app\plugins\Scripts\Plugins\CNN_F...

Model & Data Files

Model File D:\Documents\Data_Master_Thesis Browse

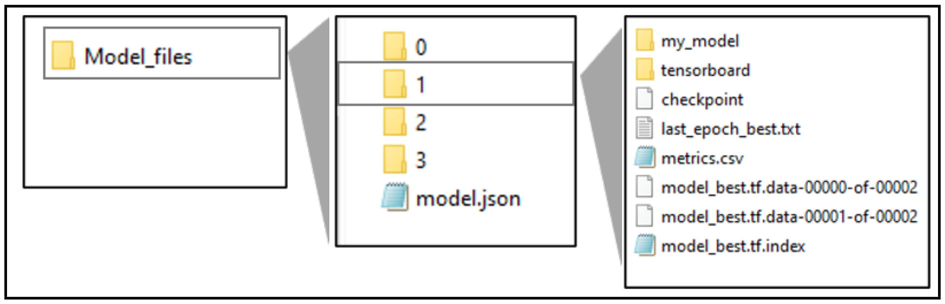
Data File D:\Documents\Data_Master_Thesis Browse

Model Folder D:\Documents\Data_Master_Thesis Browse

Task: [✓ Train] Predict

OK Abbrechen

The training will produce various files, the output folder might look similar to this , depending on the model parameters:



Model_files

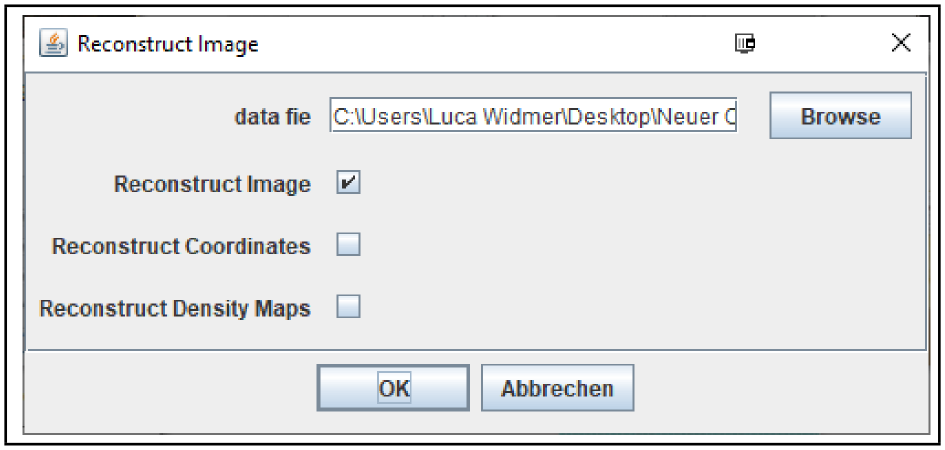
0
1
2
3
model.json

my_model
tensorboard
checkpoint
last_epoch_best.txt
metrics.csv
model_best.tf.data-00000-of-00002
model_best.tf.data-00001-of-00002
model_best.tf.index

5. **Create A Test Dataset:** Analogous to point 3. but choose the dataset type "Test". A single image is sufficient to predict.

6. **Predict on test Dataset:** Analogous to point 4. but choose the task "Predict". Give the path to the model folder containing the files of the trained model. Give the path of the dataset you would like to predict labels for. Prediction can take multiple minutes up to an hour, depending on the size of the prediction dataset.

7. **Reconstruct the Test Dataset:** Call Plugins >MiNTiF >Reconstruct Image and choose a file you want to reconstruct. MiNTiF will reconstruct all Images contained in this file in the same folder. This process can take multiple minutes.



Reconstruct Image

data fie C:\Users\Luca Widmer\Desktop\Neuer C

Reconstruct Image ✓

Reconstruct Coordinates ☐

Reconstruct Density Maps ☐

The MiNTiF Modules

4.1 Define Model

This Module is used to define model parameters. Most values will provide defaults and are described below. Recommended ranges are given for most values. These are intended to be general guidelines but should be evaluated individually for each use case. Parameters are saved to a file called “model.json” that is used in the other modules.

- **Outpath**: Folder where model files will be saved.
- **Batch Size:** The number of samples that will be passed to the network for training at one time. In general larger batch sizes are preferable, as long as the computer has sufficient RAM. *Recommended Range: 1-10.*
- **Levels**: Number of convolutional Layers. More layers allow the network to learn more high level/complex features, but increase the complexity of the model. This limits the minimum patch size (or formulated differently, for small patch sizes, users might need to reduce this parameter).
  Recommended Range: 3-6
- **Epochs**: Number of optimization passes to perform during training. More epochs will improve the quality of the model up to the point of convergence, but will also linearly increase training time.
  Recommended Range: 50-200

- **Padding:** Type of padding strategy to employ. Padding is necessary to avoid loss of information. Convolution operations reduce the output size of a layer, meaning some information on the edges of patches would get lost otherwise.
  - **Valid:** The input image is not padded. This means that the filter window always stays inside the input image. Only the valid and original elements of the input image are considered. This can lead to loss of information on the edges of images. Therefore, MiNTiF adds an separate padding to edge patches to avoid this.
  - **Same**: Pad input patches with zeros to ensure outputs have the same shape as the input.
- **Dropout:** Dropout Rate to use during training. At each training stage, individual nodes are either dropped out of the net with the selected probability, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. This may reduce over-fitting to certain features.
  Recommended Range: 0-0.5
- **Batch Normalization:** Perform Batch Normalization by re-centering and re-scaling the input layer. Reduces the impact of outlying large weights in training and can improve training speed. In general it is recommended to apply this option.
- **Patch Size:** Size of the input of the network. Larger Patches require more memory to run. Smaller patches limit the number of levels the network can have, as each convolution operation reduces the size of its patch. This parameter should be 2D (e.g. [500, 500]) or 3D (e.g. [50, 150, 150]) depending on the type of model you would like to train.
  Recommended Range: 100x100 - 1000x1000
- **Voxel Size:** Voxel size of the data. The all images will be resampled to match these dimensions during conversion. Can be 2D or 3D.
- **Cross Validation Steps**: Number of cross validation steps to perform. For each CV step the network training is repeated with a different subset of the data allocated to training, validation & testing. This allows for more accurate assessment of the metrics of the prediction. This value is overwritten by training/validation/test indices. Choose low values if you want to minimize training time, choose higher values if you want to maximize the estimation accuracy of the metrics of your model.
  Recommended Range: 1-10
- **Normalization**: If set to True data will be normalized using standard normalization before saving as a patched datafile.
- **Data Augmentation:** Perform standard data augmentation strategies on the training data. This can improve the generalizability of the trained network.

- **Model Type:** Which model architecture should be employed. The list of available models will be expanded in future releases.
  - UNet2D: Unet Network for 2D slices.
  - UNet3D: Unet Network for 3D slices.
  - MarkerSampling_MarkerExcite: Modification to Unet2D that allows prediction on a subset of markers used at training time.[1])
- **Channel Indices:** The ID of the channels that should be used as input.
- **Label Indices**: The ID of the channels that should be used as output (i.e. training targets).

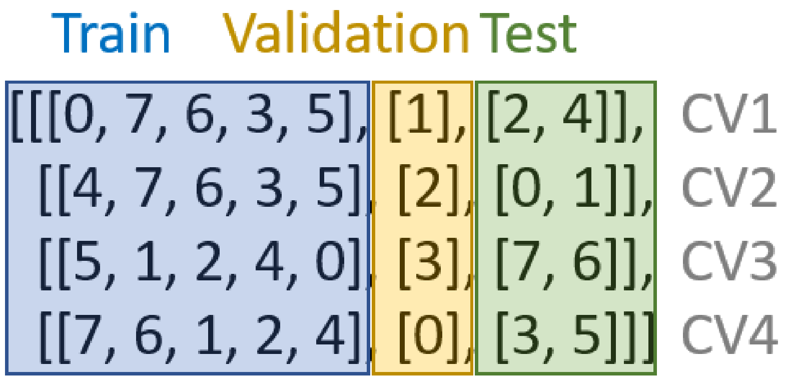
  Note: Channel & Label indices are arbitrary but need to be consistent throughout the experiment, meaning a given channel (e.g. a specific fluorescence marker, segmentation label or coordinates for a type of object) needs to always have the same ID (e.g. 1) in all datasets. Channel and label indices are independent of each other, (e.g. there can be a Channel 1 & Label 1)

[1]Alvaro Gomariz, Tiziano Portenier, Patrick M Helbling, Stephan Isringhausen, Ute Suessbier, César Nombela-Arrieta, and Orcun Goksel. Modality attention and sampling enables deep learning with heterogeneous marker combinations in fluorescence microscopy. *arXiv preprint arXiv:2008.12380*, 2020

- **Training / Validation / Test Indices:** For training datasets with multiple samples, these indices can be entered by users to manually determine which samples are used for training, validation and testing in each cross validation iteration.This can be useful to ensure reproducibility. If this is left empty, the training algorithm will attempt to distribute the samples. The indices should be given as a List of Lists, i.e.

  ```
  [[[Train_CV1], [Val_CV1],[Test_CV1]],
  [[Train_CV2], [Val_CV2], [Test_CV2]],...]
  ```

  The following example illustrates the expected format:

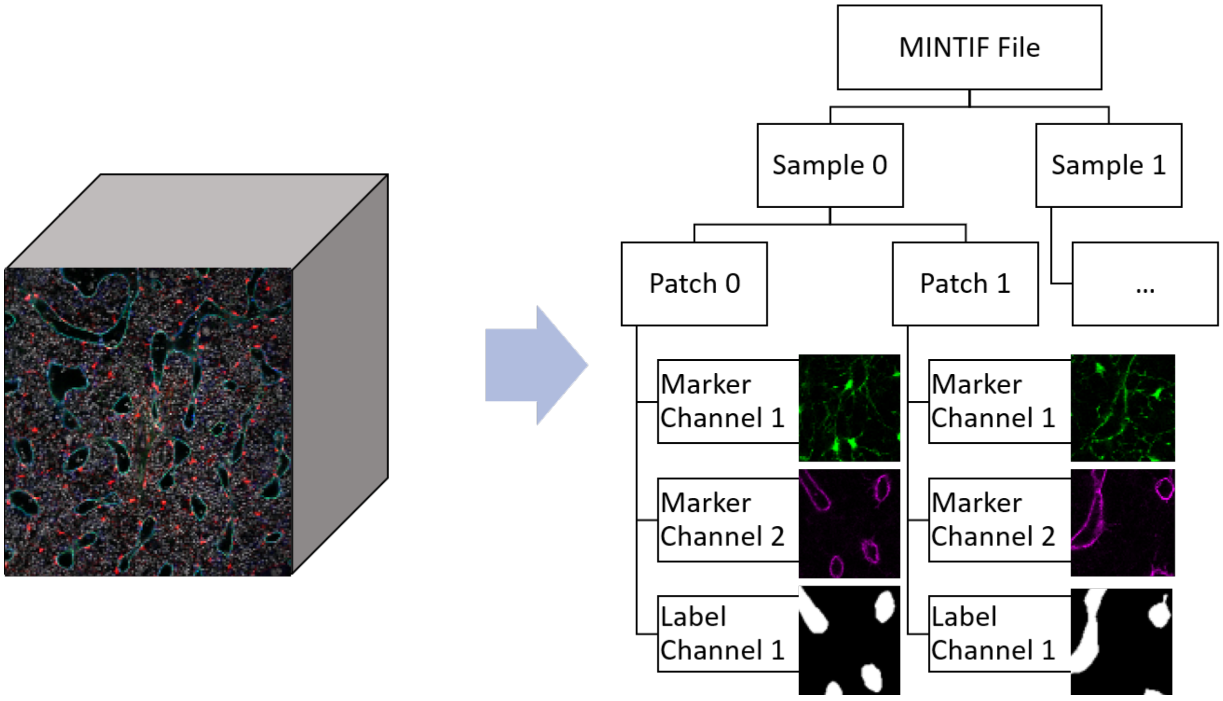
  ```
  [[[0, 7, 6, 3, 5], [1], [2, 4]],
    [[4, 7, 6, 3, 5], [2], [0, 1]],
    [[5, 1, 2, 4, 0], [3], [7, 6]],
    [[7, 6, 1, 2, 4], [0], [3, 5]]]
  ```

  Note: These indices overwrite the “Cross Validation Steps” parameter. E.g if 4 sets of indices are given, then the training will perform 4 cross validation steps independent of the value of “Cross Validation Steps”.

4.2 Create Dataset

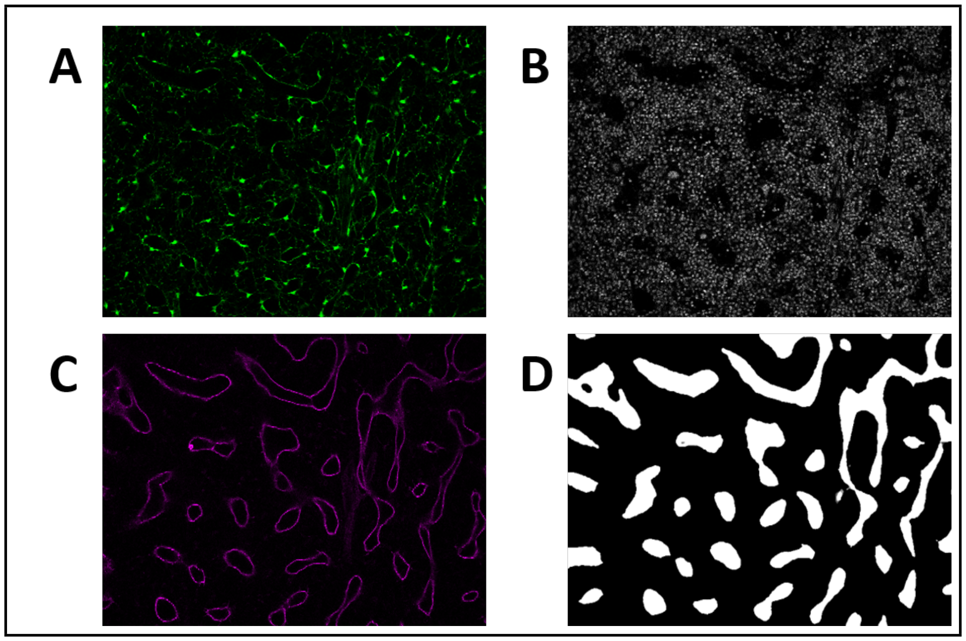
This Module is used to convert an arbitrary image loaded in Fiji to the custom data format used in MiNTiF. This is necessary for multiple reasons. First it assures the data is in a format MiNTiF can read. Second, MiNTiF also implements a tiling strategy, meaning the input image is decomposed in multiple small patches. This is necessary as large inputs significantly increase the computation requirements when using (convolutional) neural networks.

Users provide their data as multi-channel images loaded in Fiji that contain both markers and (in the case of Segmentation training data) labels as separate channels. A finished MiNTiF data file might contain several samples (images), each containing multiple patches (parts of the image), which in turn can contain multiple marker and label channels:



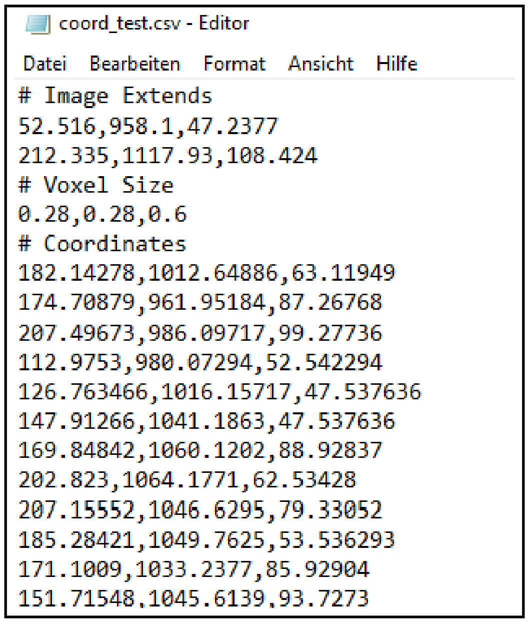
4.2.1 Segmentation Data

To train a segmentation network in MiNTiF, users will need to provide training data in the form of multi-channel images that contain segmentations as separate channels (e.g. an image that contains 3 (A-C) channels with input data, such as microscopy channels and a 4th channel (D) that contains a segmentation mask)



4.2.2 Detection Data

MiNTiF includes functionality to train models for **object detection**. At the moment only Unet Models for detection of a single class of objects are supported. To train a model for detection, users will need to provide coordinates of the objects in a **coordinate .csv** file (instead of label channels containing annotations for each class, as in segmentation).



```
# Image Extends
52.516,958.1,47.2377
212.335,1117.93,108.424
# Voxel Size
0.28,0.28,0.6
# Coordinates
182.14278,1012.64886,63.11949
174.70879,961.95184,87.26768
207.49673,986.09717,99.27736
112.9753,980.07294,52.542294
126.763466,1016.15717,47.537636
147.91266,1041.1863,47.537636
169.84842,1060.1202,88.92837
202.823,1064.1771,62.53428
207.15552,1046.6295,79.33052
185.28421,1049.7625,53.536293
171.1009,1033.2377,85.92904
151.71548,1045.6139,93.7273
```

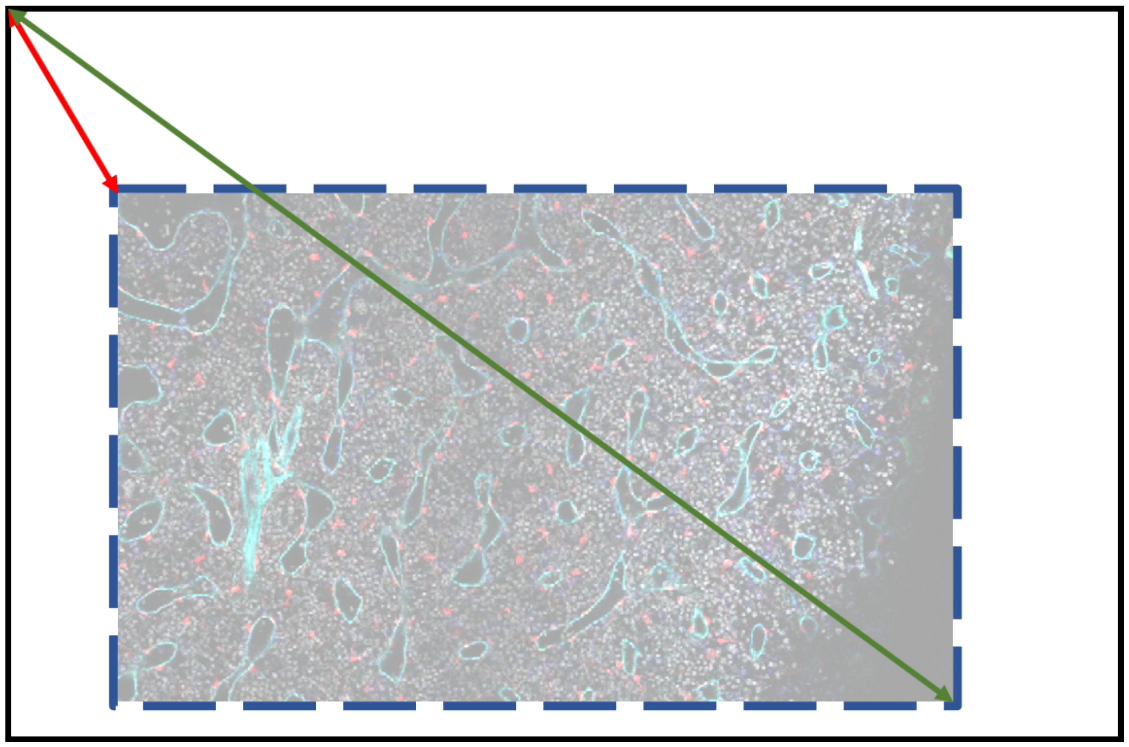


Figure 4.1: **Image Extends**: The minimum extent (red arrow) measures the distance in micrometer form the top left corner of the real world microscopy framework (black solid line) to the top left corner of the image(blue striped line). The maximum extent (green arrow) measures the distance in micrometer form the top left corner of the real world microscopy framework bottom right corner of the image

Coordinate File

The coordinate file contains the 3D coordinates of each object belonging to a class and is used as Training targets for training a detection model. each file should contain the coordinates of a single class and have the format described below. When coordinates are recorded using the "Export Points" tool, they will be in the correct format automatically. The coordinate .csv file contains the following information:

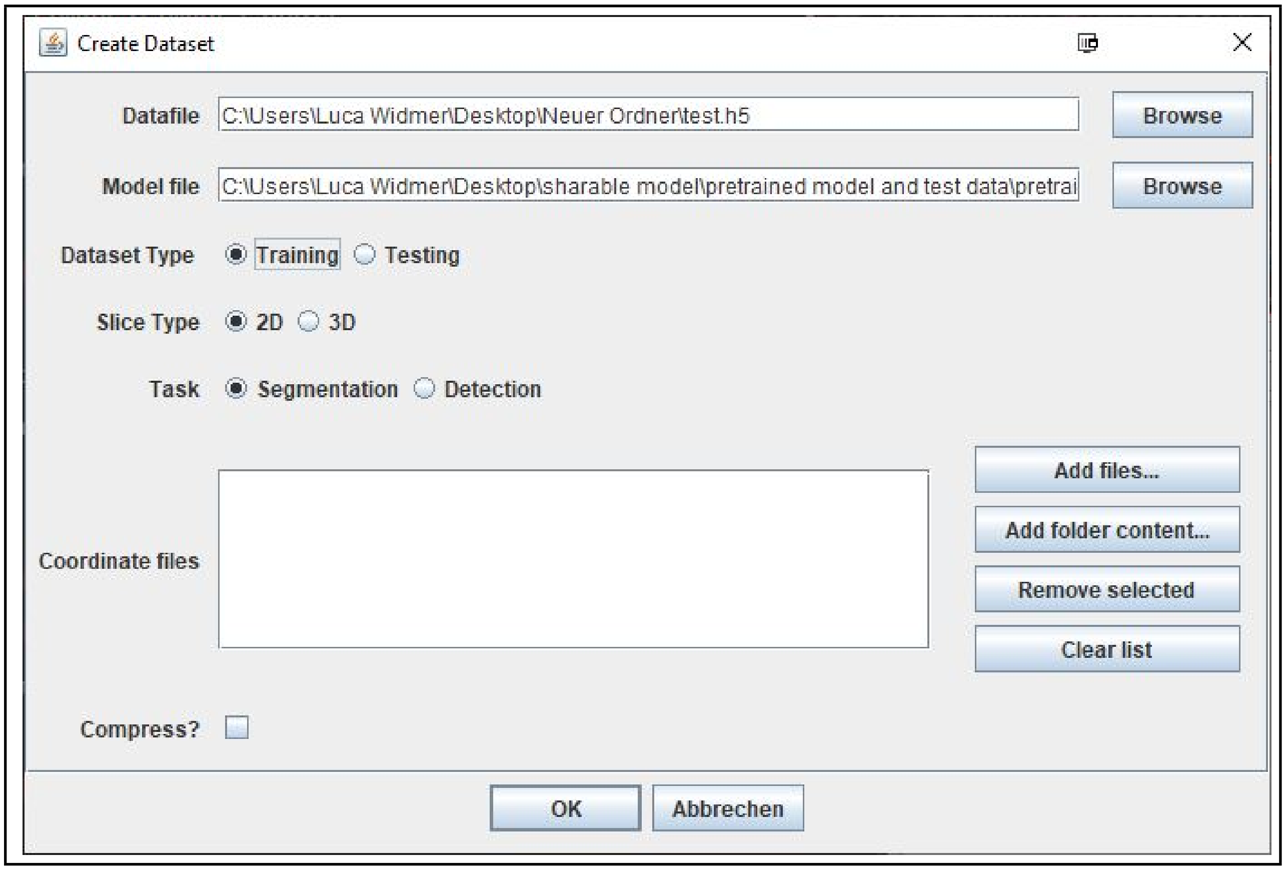
- **Image extends:** Two lines containing first the minimum then the maximum extend of the image in the real life framework of the microscope in micrometer (Fig. 4.1).
- **Voxel Size:** Voxel size in micrometer in ZXY (depth, height, width).
- **Coordinates:** In ZXY (depth, height, width) in micrometer in the real life framework of the microscope.

The three header sections have to start with a "#" and are expected in this Order (Image Extends, Voxel Size, Coordinates), the actual text in these lines is ignored. If a particular pipeline produces Coordinates that are already 0 centered on the image (meaning coordinate (0,0,0) lies on the first pixel of the image), simply set the image extends to [[0,0,0] , [max(z) ,max(x), max(y)]], again in micrometers and proceed normally.

Creating a Dataset

To start the Module, select MiNTiF >Create Dataset, while you have an Image loaded in Fiji. You will be asked to enter the following information:

- **Data File**: Path including the desired filename and file-ending (.h5) of the MiNTiF output file. If this file does not exist yet it is created in the process of conversion. If the path of an existing file is chosen, the image is instead appended to the existing file as a separate sample.
- **Model File**: Path to the previously defined model.json file. It contains information that is used in decomposing the image into patches.
- **Dataset Type**: Will this file be used for training a model or to predict labels on the contained images. This is only relevant for 2D segmentation Training datasets, as in this case, only slices that contain annotations for all labels are saved to the MiNTiF file. This significantly reduces the file size and run times.
- **Slice Type**: Are Patches 3D or 2D.
- **Task:** What task is the model intended for that will be used on this data. Currently only Segmantic Segmentation and Detection are implemented.
- **Coordinate Files**: List of paths to coordinate .csv files. These are only used in training of detection models and are expected to be in a format described in detail below. These are not needed for segmentation.
  Note: Multiple files can be passed and converted to MiNTiF files and will be handled as separate classes. But at the moment only a single class per detection model is supported for training.
- **Compress:** Should the output file be compressed. If selected, this option reduces the resulting file size but will slightly increase the time needed to convert and later load the file.


Create Dataset

Datafile: C:\Users\Luca Widmer\Desktop\Neuer Ordner\test.h5

Model file: C:\Users\Luca Widmer\Desktop\sharable model\pretrained model and test data\pretrai

Dataset Type: [● Training] ○ Testing

Slice Type: [● 2D] ○ 3D

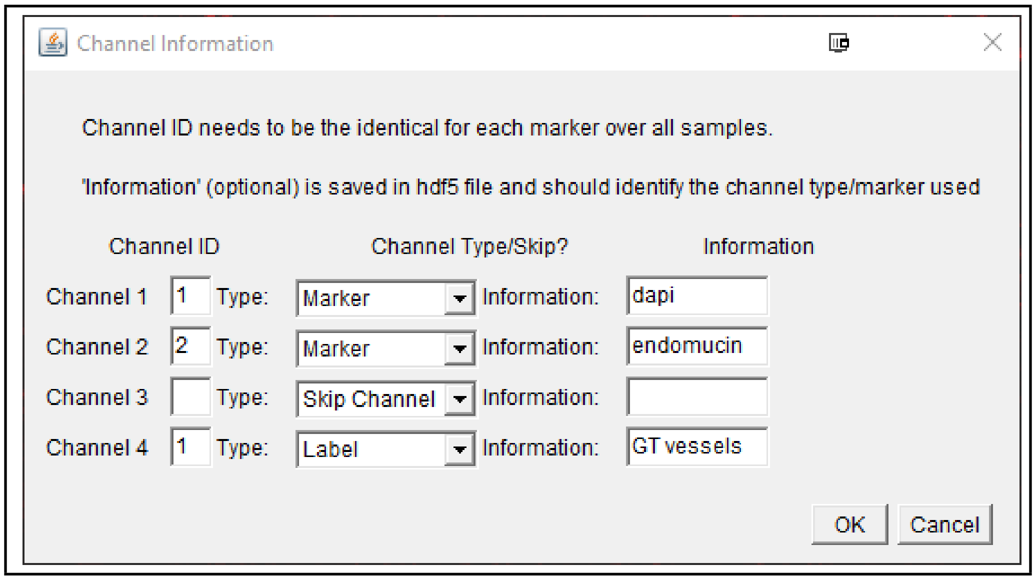
Task: [● Segmentation] ○ Detection

Coordinate files

Compress? ☐

When you click OK, a new window will appear asking for information on each channel in the image:

- **Channel ID:** which numerical ID should be assigned to this channel. This channel ID is used for the internal representation in MiNTiF, make sure that it is consistent across samples. (e.g. a specific marker/label channel should always have the same channel ID)
- **Channel Type:**
  - **Marker**: will be used as inputs in the network if the channel ID matches the marker channel IDs defined in the model.
  - **Label**: will be used as a training target (output) in the model during training if the channel ID matches the label channel IDs defined in the model. Generally used for Segmentation models *Note: you can save channels with indices that do not match indices in the model.json file, but they will be ignored by the model.*
  - **Skip Channel**: this channel will be discarded and not saved in the final file.
- **Information**: The contents of this box are saved as metadata in the final MiNTiF file and in the information.txt file, but have no function in the plugin at the moment. The intention is that this information should make it possible to replicate the generation of the dataset by other users.


Channel Information

Channel ID needs to be the identical for each marker over all samples.

'Information' (optional) is saved in hdf5 file and should identify the channel type/marker used

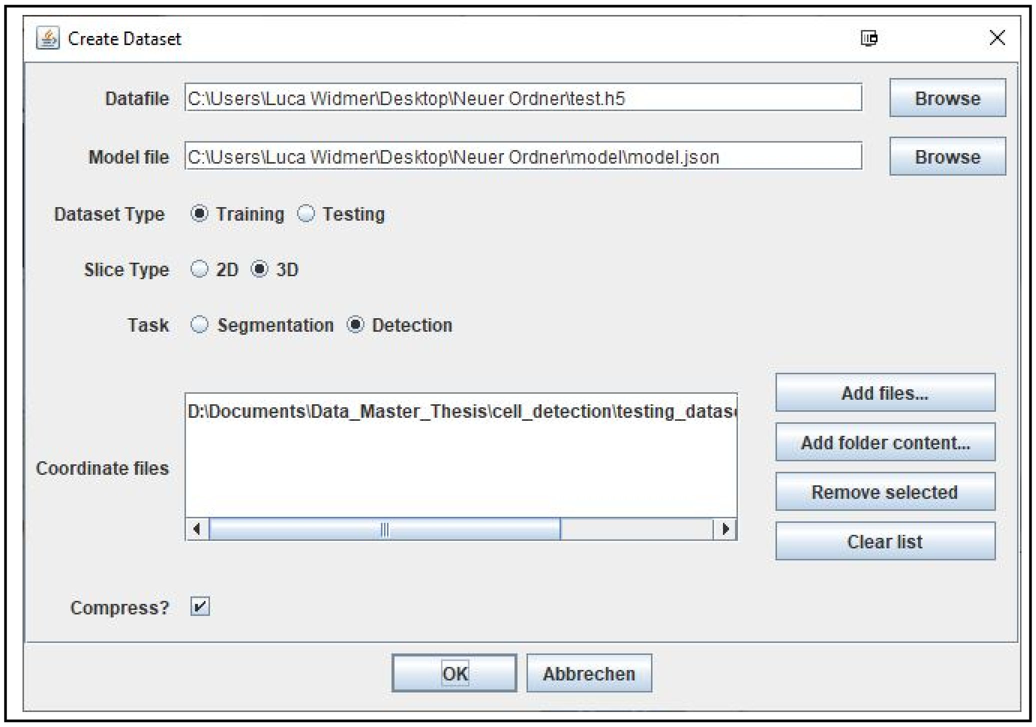
	Channel ID		Channel Type/Skip?		Information
Channel 1	1	Type:	Marker	Information:	dapi
Channel 2	2	Type:	Marker	Information:	endomucin
Channel 3		Type:	Skip Channel	Information:	
Channel 4	1	Type:	Label	Information:	GT vessels

After you click OK, the image will be resized to the desired voxel dimensions and saved first as intermediary tiff files and then appended to an MiNTiF file together with necessary metadata. This process can take minutes up to an hour for very large datafiles.

Note: Do not rename any temporary files while the conversion is running.

4.2.3 Dataset Creation for Detection

The process is similar to the one described for segmentation, with the difference that you will need to provide one or more csv file(s) for the coordinate ground truth data if you want to train a detection model.



Create Dataset

Datafile: C:\Users\Luca Widmer\Desktop\Neuer Ordner\test.h5

Model file: C:\Users\Luca Widmer\Desktop\Neuer Ordner\model\model.json

Dataset Type: [● Training] ○ Testing

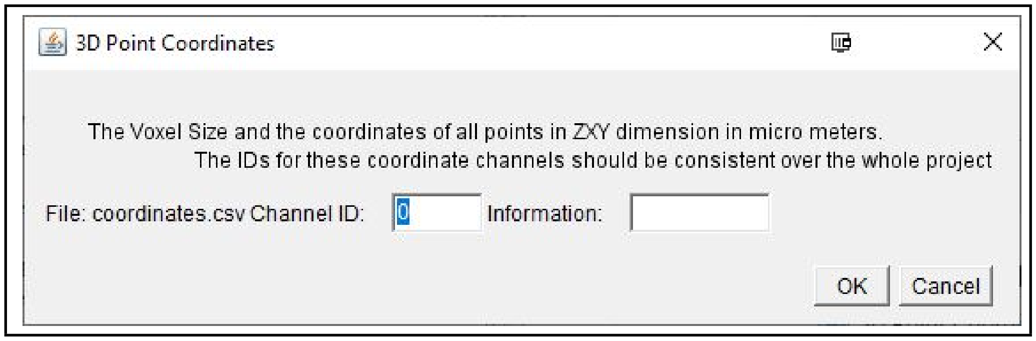
Slice Type: ○ 2D [● 3D]

Task: ○ Segmentation [● Detection]

Coordinate files: D:\Documents\Data_Master_Thesis\cell_detection\testing_datas

Compress? [✓]

Now provide information on the coordinate labels the same way you would for segmentation labels.(i.e give it a consistent index over the whole project and note information to make it reproducible).



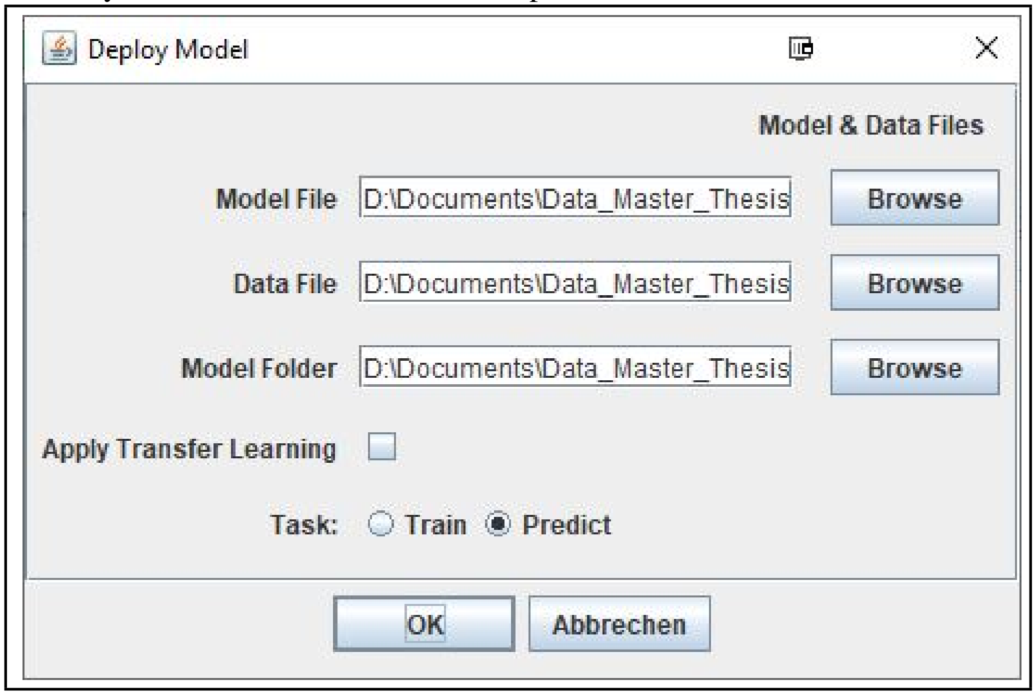
3D Point Coordinates

The Voxel Size and the coordinates of all points in ZXY dimension in micro meters.
The IDs for these coordinate channels should be consistent over the whole project

File: coordinates.csv Channel ID: 0 Information:

4.3 Deploy Model

This Module uses a Python tensorflow framework to train a model and/or predict on MiNTiF data files using a trained model. You typically will want to call the script first as training on training data. Later you can use this same Module to predict labels on new test data.



Deploy Model

Model & Data Files

Model File: D:\Documents\Data_Master_Thesis

Data File: D:\Documents\Data_Master_Thesis

Model Folder: D:\Documents\Data_Master_Thesis

Apply Transfer Learning: ☐

Task: Train [✓ Predict]

4.3.1 Training a Model

This step can be very time intensive. Training a model on a Desktop PC without dedicated GPU hardware can take multiple hours to multiple days, depending on the model. However, if necessary, the training can be aborted and continued later by calling the training module on the same files.
To train a model on your data you will need:

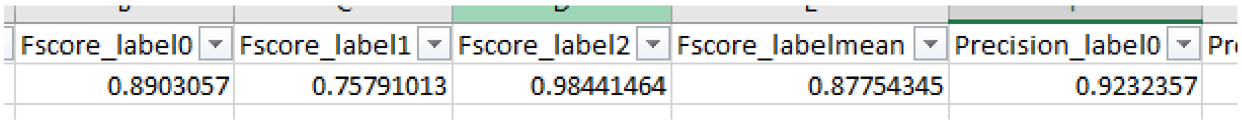
- **Model File:** A model file created with "Create Model"
- **Data File:** A Data File created with "Create Dataset"
  - The Samples in the Dataset should contain marker and label channels/coordinates with indices matching the indices in the model file.
- **Model Folder:** The Folder where files for the trained model will be saved.
- **Apply Transfer Learning:** Choosing this option will train a model using the provided model weights as starting point. This can be used to continue training an existing model or to fine tune a model using new / additional data.

By calling the module "Deploy Model" with the option "Task: Train", A TensorFlow model will be defined using the parameters saved in the model file and trained on the data in the data file. The training progress can be supervised using Tensorboard as described below.

4.3.2 Testing the Model & Interpreting Metrics Files

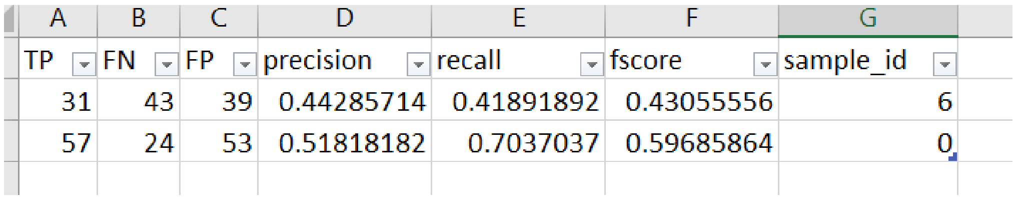
To evaluate the performance of a trained model, MiNTiF calculates some test metrics and saves them in a `metrics.csv` file. Metrics files are generated for every cross validation step after training has finished. it contains various metrics to evaluate the performance of the model by testing it on part oft the data that was not used for training (the samples selected as 'Test' data for this cross validation step, either manually by the user or automatically by MiNTiF). The metrics used to test a Model depend on the application and implementation, at the moment there are two metrics calculations implemented, one for semantic segmentation models and one for Detection. For a good intuitive explanation on most of these metrics we recommend this article.

- **The semantic segmentation metrics.csv** file contains the metrics of the given cross validation step as one line, aggregated over all test samples. The metrics are calculated separately for each class $(0 - n)$ and the background class $(n + 1)$, i.e. the group of pixels belonging to no annotated class, as well as a score mean over all classes (including background).



Fscore_label0	Fscore_label1	Fscore_label2	Fscore_labelmean	Precision_label0	Pr
0.8903057	0.75791013	0.98441464	0.87754345	0.9232357	

- **The detection metrics.csv** file contains metrics separated by sample:

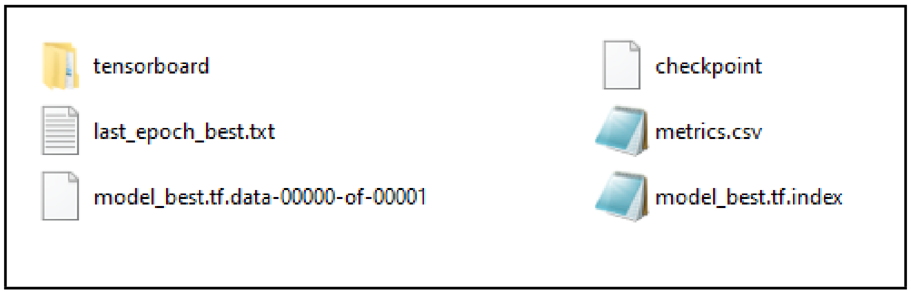


TP	FN	FP	precision	recall	fscore	sample_id
31	43	39	0.44285714	0.41891892	0.43055556	6
57	24	53	0.51818182	0.7037037	0.59685864	0

4.3.3 Predicting on new data

After a model has been trained, it can be used to predict labels of new images. This can take multiple minutes per image. To train a model on data you will need:

- **Model File**: A model file created with "Create Model".
- **Data File**: A Data File created with "Create Dataset"
- **Model Folder**: The Folder where files for the trained model have been saved. The contents of the folder should look similar to this:



Note: if a model was trained with multiple cross validation steps, the user needs to provide the path of the numbered subfolder of the cross validation set they want to use for prediction.

The Samples in the dataset should contain marker and label channels with IDs matching the indices in the model file. Certain models can handle missing markers, but others will need the same input marker channels that were used to train the models. Label channels are unnecessary for this step and will be ignored.

4.3.4 Prediction using a pre-trained or imported model

To use a pretrained model to predict, you will need the following:

- A datafile which contains marker channels using the same channels & indices the model was trained on. This information needs to be provided by the creator of the model. When defining & exporting a model from MINITF, this information is included in the `info_{imagename}.txt` file.
- The tensorflow saved model files of the pretrained models. When exporting a model from MINITF, these files are located in the `my_model` folder.
- The model.json file which describes the model parameters.

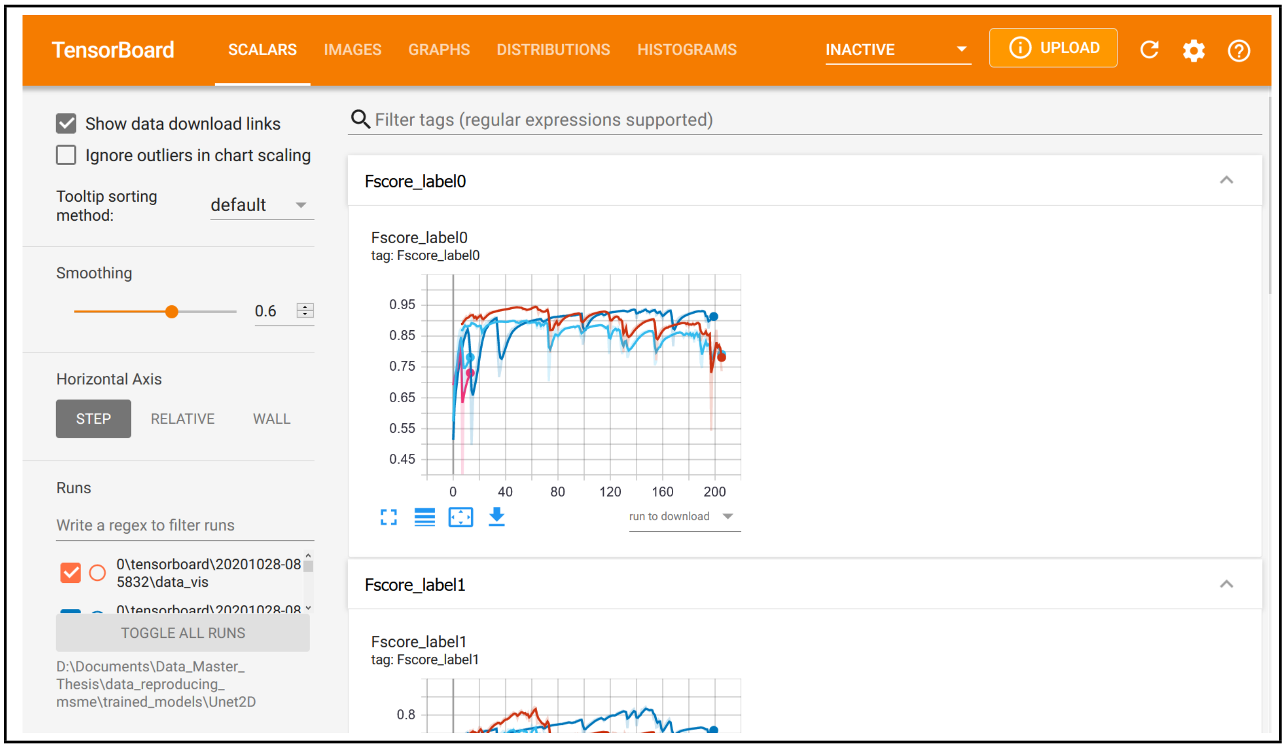
Call the Module "Deploy Plugin" and choose the folder containing the trained model files as "model folder" parameter.

4.3.5 Tensorboard

Tensorboard can be used to visualize and explore training progress as well as many other aspects of the model. Tensorboard is part of the environment that was installed during setup. To access TensorFlow, open a command prompt and run the following command:

```
conda activate cnnframework & tensorboard --logdir {logdir}
```

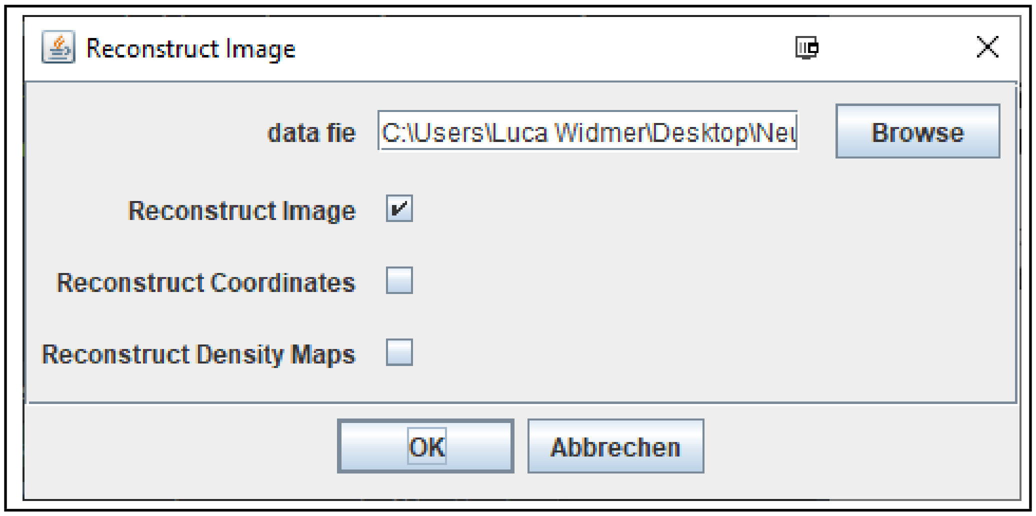
where `{logdir}` is the path to the model folder. This command is automatically generated with the matching `{logdir}` path during training and can be copied directly from the Fiji console. A link will be created to open Tensorboard in a browser. Tensorboard offers a wide variety of information. But most importantly it displays various training metrics in real time. This is also currently the only way to monitor the progress of the model training. If the training process is aborted for whatever reason, simply start the training again using the same files. The training will continue at the last checkpoint.



☑ Show data download links
☐ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis [✓ STEP] RELATIVE WALL
Runs
Write a regex to filter runs
☑ 0\tensorboard\20201028-085832\data_vis
0\tensorboard\20201028-08
D:\Documents\Data_Master_Thesis\data_reproducing_msme\trained_models\Unet2D

4.4 Reconstruct Images

This module reconstructs all images contained in an MiNTiF file into tiff images that can be loaded in Fiji. If coordinates were predicted on this Data, it also creates coordinate files containing the predicted coordinates of objects. The images will be reconstructed in the same folder as the MiNTiF file. The reconstructed images follow the naming convention `Sample_number#{image_name}_reconstructed.tiff`, where `Sample_#` is the name of the sample in the MiNTiF File and `{image_name}` is the original filename of the image. The Reconstruction of the image and the coordinates can be selected independently. Additionally, if you reconstruct the image of a Detection dataset, the Density Maps can also be reconstructed. Density maps are the internal representation of the coordinates in MiNTiF and are used to predict the coordinates.


Reconstruct Image

data fie: C:\Users\Luca Widmer\Desktop\Neu

Reconstruct Image [✓]

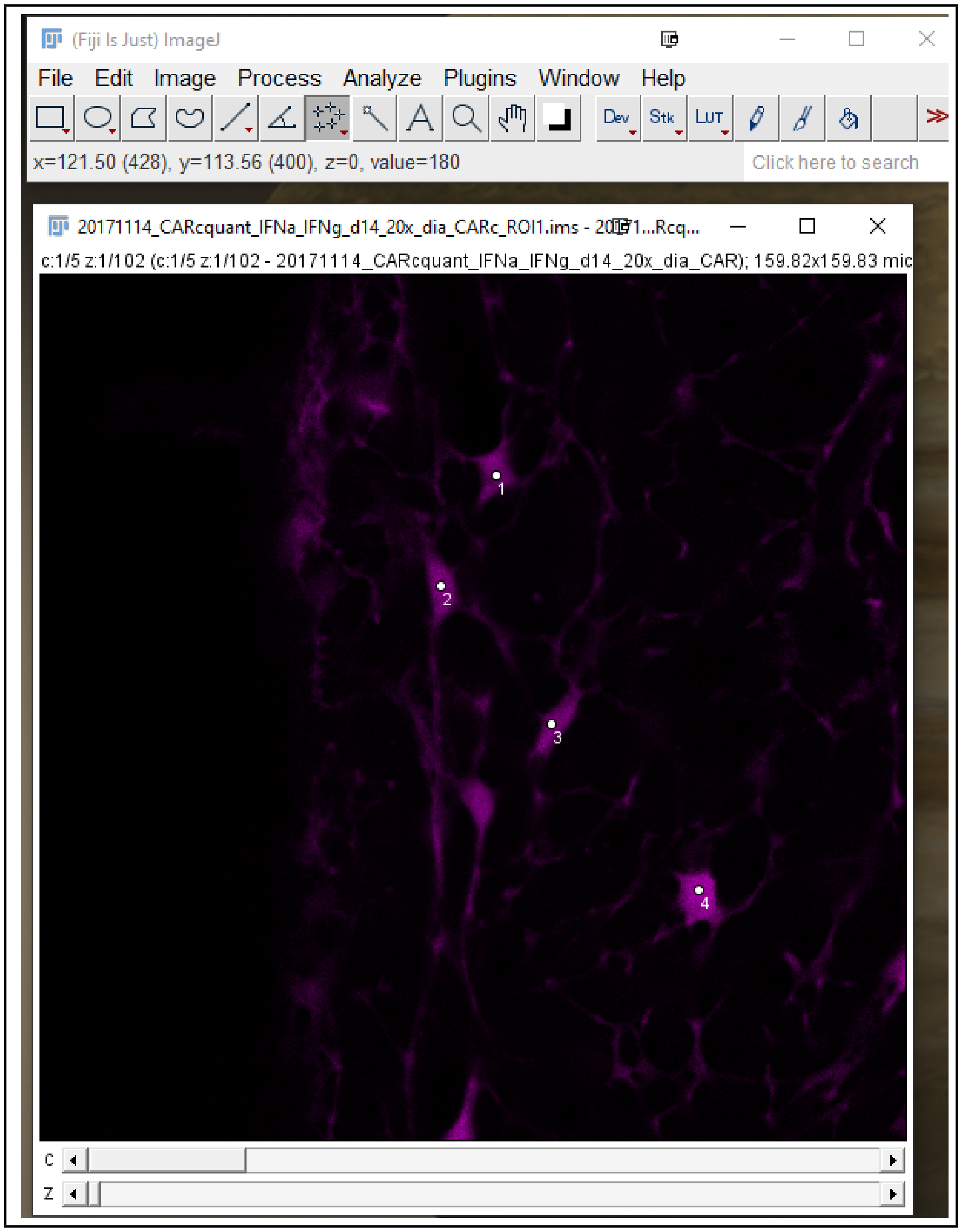
Reconstruct Coordinates [ ]

Reconstruct Density Maps [ ]

4.5 Import & Export Points

These Modules make it easy to import & export point annotations into/from coordinate csv files used for MiNTiF Detection. These csv files are used as during dataset creation for detection tasks and define the training targets. It is assumed that each class (training target) is defined in a separate coordinate file.

Export: Using the Multi Point Tool, or using other Fiji plugins, create a multi-point Selection on the image you would like to annotate:


(Fiji Is Just) ImageJ
File Edit Image Process Analyze Plugins Window Help
Dev Stk LUT
x=121.50 (428), y=113.56 (400), z=0, value=180
Click here to search
20171114_CARcquant_IFNa_IFNg_d14_20x_dia_CARc_ROI1.ims - 2017...Rcq...
c:1/5 z:1/102 (c:1/5 z:1/102 - 20171114_CARcquant_IFNa_IFNg_d14_20x_dia_CAR); 159.82x159.83 mic
1
2
3
4
c
z

Next, select Plugins >MiNTiF >Utilities >Export Points and select the output path for the coordinate file. Click OK and the points will be converted to a MiNTiF coordinate csv file.

Import: Select Plugins >MiNTiF >Utilities >Import Points and choose a MiNTiF coordinate csv file you want to import. The points will be loaded and displayed in Fiji automatically. The imported coordinates will be adapted to the current image size.

Troubleshooting & FAQ

- **Problem: The environment installation (Setup Step 5) Fails:** It is possible that the conda environment already exists (if you have ran this command before), try running the same command () With the option "overwrite" set to True. Otherwise you can try to manually install the environment like this:
  Open a command line interface and typing the following command :

  ```
      conda env create -f
      {Userspecific}\Fiji.app\MINITIF_utils\environment.yml
  ```

  Here you should give the full path to the environment.yml file including the user specific part.

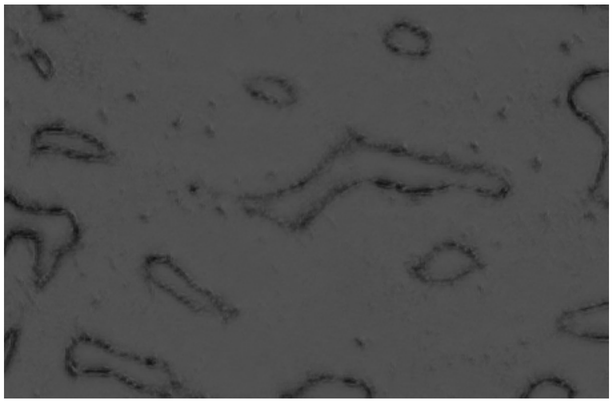
- **Problem: Fiji gives the following or similar error when running MiNTiF:**

  ```
  INFO:fiji:writing parameters in
  C:\Users\lucaw\Desktop\temp_model.json
  The command "conda" was not found
  ```

  The "conda" command is not accessible to the subprocess used in the backend of MiNTiF. Make sure the the path `\{Usrespecific}\anaconda3\condabin` is added to the PATH Environment variable and conda is initialized as described here (for Windows) or follow the troubleshooting steps in this helpful article (For Mac OS).
  A simpler (but possibly more destructive fix) might be to reinstall anaconda and check "Add Anaconda to my PATH environment variable" during installation NOTE: this might delete older environments you might have used for previous projects.

  Note: Multiple Anaconda Installations (such as Miniconda & regular anaconda on the same machine) may cause issues in successfully calling the backend scripts for MiNTiF. If possible, only have a single instance of anaconda active on this machine or ensure that this environment is installed in the default anaconda installation.

- **Problem: when predicting on an image the results look random and not clearly defined**:



  There might be multiple explanations. First, make sure you selected the correct files when running the prediction (e.g. for `Model Folder` you selected the directory that containes the `model_best.tf.data` files of the model you want to use for prediction).
  Another possibility is that the model was not trained successfully, for example because the training was not run correctly (e.g. the wrong files were selected, or the indices of channels were mixed up between different samples). In this case, the model performance should be very bad, as measured in the metrics file. If this is case try to rerun the training step (with the corrected data) and see if there are improvements.

- **Problem: I would like to view the contents of the data files used in MiNTiF** There is currently no way to directly view the contents of these files in Fiji, but you can use external programs such as **HDFView** to to visualize them.

- **Problem: When I reconstruct a training dataset, there are many empty slices in the resulting tiff file.** Only annotated slices (slices that contain labeled pixels for all label channels) are saved to the MiNTiF data file. In the reconstruction, the un-annotated slices are reconstructed as empty slices.

- **Problem: Fiji gives the following or similar error when running MiNTiF to train a model:**

```
[...]
line 599, in eval_step for data_batch in ds:
TypeError: 'NoneType' object is not iterable
```

  There may not be enough samples in the dataset. If possible, try adding multiple labeled samples to the training dataset.

- **Problem: Reconstructed Coordinates are missing some points Possible explanations:** Points at the edge of patches (within one cell diameter of edge) are discarded, consider lowering the cell diameter parameter

- **Problem: Fiji gives the following or similar error when converting a multichannel colour image :**

  ```
  [...]
  line 78, in extract_patch_from_img
  ret[ww[0]:ww[1], ww[2]:ww[3], ww[4]:ww[5]] = insert
  ValueError: could not broadcast input array
  from shape (1,480,0) into shape (1,480,279)
  ```

  Colour channels are not supported for conversion. Try to split the colour channel into multiple grey value channels before conversion.

- **Problem: Fiji gives the following or similar error when converting Data for a detection Dataset**

  ```
  [...]File "h5py\h5o.pyx", line 190, in h5py.h5o.open

  KeyError: "Unable to open object (object 'label_channel_2'
  doesn't exist)"
  ```

  The plugin expects a label channel of a certain name (here “2”) to appear in the dataset, because it was defined in the model.json file. Either rerun the dataset creation and add a channel with the expected channel ID or redefine the model file to no longer expect this channel ID.

- **Problem: Fiji gives the following or similar Warning reconstructing a dataset**

  ```
  WARNING, found NaN values in reconstructed
  image and filled with 0

  Slices:
   [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
   15 16 17 18 19 20 22 23 24 25 26 27 28 29 30 31
   32 33 34 35 36 37 38 39 40 42 43 44 45 46 47 48 49
   50 51 52 53 54 55 56 57 58]

  Saving image to D:\Documents\Data_Master_Thesis\test\
  Sample_0_test2_BMquantIV_Femur_7_aged_meta_reconstructed.tiff
  done
  ```

  In general this warning is displayed when some parts of the image were not possible to reconstruct. This can sometimes happen if the patch size is significantly larger than the image, though the plugin attempts to handle this case, or if a 2D training dataset is reconstructed, as in that specific case, only image slices with annotation in all label channels are actually saved to the MiNTiF File. During reconstruction, these slices are placed at their original position and any other slices are filled with 0s.

Bibliography

Alvaro Gomariz, Tiziano Portenier, Patrick M Helbling, Stephan Isringhausen, Ute Suessbier, César Nombela-Arrieta, and Orcun Goksel. Modality attention and sampling enables deep learning with heterogeneous marker combinations in fluorescence microscopy. *arXiv preprint arXiv:2008.12380*, 2020.