DeepD3

Release 2023

Andreas M Kist, Martin H P Fernholz

Jan 31, 2023

USER GUIDE

With DeepD3, you are able to predict the presence and absence of dendritic spines and dendrites in 3D microscopy stacks. We evaluated DeepD3 on a variety of species, resolutions, dyes and microscopy types. In this documentation, you find information how to train your own DeepD3 model, how to use DeepD3 in inference mode and how to use the API to create custom scripts.

CHAPTER

ONE

DEEPD3 INFERENCE

Open the inference mode using `deepd3-inference`. Load your stack of choice (we currently support TIF stacks) and specify the XY and Z dimensions. Next, you can segment dendrites and dendritic spines using a DeepD3 model from the Model Zoo by clicking on `Analyze -> Segment dendrite and spines`. Afterwards, you may clean the predictions by clicking on `Analyze -> Cleaning`. Finally, you may build 2D or 3D ROIs using the respective functions in `Analyze`. To test the 3D ROI building, double click in the stack to a region of interest. A window opens that allows you to play with the hyperparameters and segments 3D ROIs in real-time.

All results can be exported to various file formats. For convenience, DeepD3 saves related data in its "proprietary" hdf5 file (that you can open using any hdf5 viewer/program/library). In particular, you may export the predictions as TIF files, the ROIs to ImageJ file format or a folder, the ROI map to a TIF file, or the ROI centroids to a file.

Most functions can be assessed using a batch command script located in `deepd3/inference/batch.py`.

CHAPTER

TWO

TRAINING YOUR OWN DEEPD3 MODEL

Use `deepd3-training` to start the GUI for generating training sets.

For each of your training set, please provide

- The original stack as e.g. TIF files
- The spine annotations (binary labels) as TIF or MASK files (the latter from [pipra](https://github.com/anki-xyz/pipra))
- The dendrite annotations as SWC file (only tested for SWC-files generated by [NeuTube](https://neutracing.com/download/))

2.1 Create training data

Click on the button "Create training data". For each of your stacks, import the stack, the spine annotation and the dendrite annotation file. If you dendrite annotation is a SWC file, it will create a 3D reconstruction of the SWC file, which will be stored for later use. If you reload the SWC, it will ask you if you want to keep the 3D reconstruction.

After importing all files, enter the metadata (resolution in x, y and z) and determine the region of interest using the bounding box and the sliders. Shortcuts are `B` for current plane is **z begin** and `E` for **z end**. You may enable or disable the cropping to the bounding box. If you are happy, save this region as `d3data`-file.

2.2 View training data

Click on the button "View training data" to re-visit any `d3data` files. You also are able to see and potentially manipulate the metadata associated the `d3data` file.

2.3 Arrange training data

For training, you need to create a `d3set`. This is an assembly of `d3data` files. Click on the button "Arrange training data". Then, simply load all relevant data using the "Add data to set" button and select appropriate `d3data` files. Clicking on "Create dataset" allows you to save your assembly as `d3set` file.

2.4 Actual training

We have prepared a Jupyter notebook in the folder `examples`. Follow the instructions to train your own deep neural network for DeepD3 use. For professionals, you also may utilize directly the files in `model` and `training` to allow highly individualized training. You only should ensure that your model allows arbitrary input and outputs two separate channels (dendrites and spines).

CHAPTER

THREE

DEEPD3 CORE

3.1 analysis

class deepd3.core.analysis.**ROI2D_Creator**(*dendrite_prediction*, *spine_prediction*, *threshold*)

clean(*maxD*, *minS*, *dendrite_threshold=0.7*)

Cleanes ROIs

Parameters

- **maxD** (*int*) – maximum distance to dendrite in px
- **minS** (*int*) – minimum size of ROIs in px
- **dendrite_threshold** (*float, optional*) – _description_. Defaults to 0.7.

Returns
old ROI count, new ROI count

Return type
tuple

create(*applyWatershed=False*, *maskSize=3*, *minDistance=3*)

Creates 2D ROIs

Parameters

- **applyWatershed** (*bool, optional*) – Apply Watershed algorithm. Defaults to False.
- **maskSize** (*int, optional*) – Size of the distance transform mask. Defaults to 3.
- **minDistance** (*int, optional*) – Minimum distance between ROIs in Watershed algorithm. Defaults to 3.

Returns
ROIs found

Return type
int

zSignal

class deepd3.core.analysis.**ROI3D_Creator**(*dendrite_prediction*, *spine_prediction*, *mode='floodfill'*, *areaThreshold=0.25*, *peakThreshold=0.8*, *seedDelta=0.1*, *distanceToSeed=10*, *dimensions={'xy': 0.094, 'z': 0.5}*)

create(*minPx*, *maxPx*, *minPlanes*, *applyWatershed=False*, *dilation=0*)

Create 3D ROIs

Parameters

- **minPx** (*int*) – only retain 3D ROIs containing at least *minPx* pixels
- **maxPx** (*int*) – only retain 3D ROIs containing at most *maxPx* pixels
- **minPlanes** (*int*) – only retain 3D ROIs spanning at least *minPlanes* planes
- **applyWatershed** (*bool, optional*) – Apply watershed algorithm to divide ROIs. Defaults to False.
- **dilation** (*int, optional*) – Dilate dendrite probability map. Defaults to 0.

Returns
number of retained ROIs

Return type
int

log

zSignal

class deepd3.core.analysis.**Stack**(*fn*, *pred_fn=None*, *dimensions={'xy': 0.094, 'z': 0.5}*)

cleanDendrite(*dendrite_threshold=0.7*, *min_dendrite_size=100*)

Cleaning dendrite

Parameters

- **dendrite_threshold** (*float, optional*) – Dendrite probability threshold. Defaults to 0.7.
- **min_dendrite_size** (*int, optional*) – Minimum dendrite size. Defaults to 100.

Returns
Cleaned dendrite prediction map

Return type
numpy.ndarray

cleanDendrite3D(*dendrite_threshold=0.7*, *min_dendrite_size=100*, *preview=False*)

Cleaning dendrites in 3D

Parameters

- **dendrite_threshold** (*float, optional*) – Dendrite semantic segmentation threshold. Defaults to 0.7.
- **min_dendrite_size** (*int, optional*) – Minimum dendrite size in px in 3D. Defaults to 100.
- **preview** (*bool, optional*) – Enable preview option. Defaults to False.

Returns
Cleaned dendrite

Return type
numpy.ndarray

cleanSpines(*dendrite_threshold=0.7*, *dendrite_dilation_iterations=12*, *preview=False*)

Cleaning spines in 2D

Parameters

- **dendrite_threshold** (*float, optional*) – Dendrite threshold for segmentation. Defaults to 0.7.
- **dendrite_dilation_iterations** (*int, optional*) – Iterations to enlarge dendrite. Defaults to 12.
- **preview** (*bool, optional*) – Enable preview option (not overwriting predictions). Defaults to False.

Returns
cleaned spines stack

Return type
numpy.ndarray

closing(*iterations=1*, *preview=False*)

Closing operation on dendrite prediction map

Parameters

- **iterations** (*int, optional*) – Iterations of closing operation. Defaults to 1.
- **preview** (*bool, optional*) – Enables preview mode. Defaults to False.

Returns
cleaned dendrite map

Return type
numpy.ndarray

predictFourFold(*model_fn*, *tile_size=128*, *inset_size=96*, *pad_op=<function mean>*, *zmin=None*, *zmax=None*)

Similar to *predictInset* (single tile prediction), but with four-way correction

Parameters

- **model_fn** (*str*) – path to Tensorflow/Keras model
- **tile_size** (*int, optional*) – Size of full tile. Defaults to 128.
- **inset_size** (*int, optional*) – Size of tile inset (probability map to be kept). Defaults to 96.
- **pad_op** (*_type_, optional*) – Padding operation. Defaults to np.mean.
- **zmin** (*_type_, optional*) – Z-index minimum. Defaults to None.
- **zmax** (*_type_, optional*) – Z-index maxmimum. Defaults to None.

Returns
operation was successful

Return type
bool

predictInset(*model_fn*, *tile_size=128*, *inset_size=96*, *pad_op=<function mean>*, *zmin=None*, *zmax=None*, *clean_dendrite=True*, *dendrite_threshold=0.7*)

Predict inset

Parameters

- **model_fn** (*str*) – path to Tensorflow/Keras model
- **tile_size** (*int, optional*) – Size of full tile. Defaults to 128.
- **inset_size** (*int, optional*) – Size of tile inset (probability map to be kept). Defaults to 96.
- **pad_op** (*_type_, optional*) – Padding operation. Defaults to np.mean.
- **zmin** (*_type_, optional*) – Z-index minimum. Defaults to None.
- **zmax** (*_type_, optional*) – Z-index maxmimum. Defaults to None.
- **clean_dendrite** (*bool, optional*) – Cleaning dendrite. Defaults to True.
- **dendrite_threshold** (*float, optional*) – Dendrite probability threshold. Defaults to 0.7.

Returns
operation was successful

Return type
bool

predictWholeImage(*model_fn*)

Predict whole image, plane by plane

Parameters
model_fn (*str*) – path to Tensorflow/Keras model file

Returns
operation was successful

Return type
bool

tileSignal

deepd3.core.analysis.**_distance_to_seed**(*seed*, *pos*, *delta_xy=1*, *delta_z=1*)

Computes the euclidean distance between seed pixel and current position *pos*

Parameters

- **seed** (*tuple*) – seed pixel coordinates (x,y,z)
- **pos** (*tuple*) – current position coordinates (x,y,z)

Returns
euclidean distance between seed and current position

Return type
float

deepd3.core.analysis.**_get_sorted_seeds**(*stack*, *threshold=0.8*)

Sort seeds according to their highest prediction value

Parameters

- **stack** (*numpy ndarray*) – The stack with the predictions
- **threshold** (*float, optional*) – The threshold for being a seed pixel. Defaults to 0.8.

Returns
seed coordinates sorted by prediction value

Return type
numpy.ndarray

deepd3.core.analysis.**_neighbours**(*x*, *y*, *z*)

Generates 26-connected neighbours

Parameters

- **x** (*int*) – x-value
- **y** (*int*) – y-value
- **z** (*int*) – z-value

Returns
neighbour indices of a given point (x,y,z)

Return type
list

deepd3.core.analysis.**centroid3D**(*im*)

Computes centroid from a 3D binary image

Parameters
im (*numpy.ndarray*) – binary image

Returns
centroid coordinates z,y,x

Return type
tuple

deepd3.core.analysis.**centroids3D_from_labels**(*labels*)

Computes the centroid for each label in an 3D stack containing image labels. 0 is background, 1…N are foreground labels. This function uses image moments to compute the centroid.

Parameters
labels (*numpy.ndarray*) – ROI labeled image (0…N)

Returns
Returns first-order moments, zero-order moments and covered planes

Return type
tuple(numpy.ndarray, numpy.ndarray, numpy.ndarray)

deepd3.core.analysis.**cleanLabels**(*labels*, *rois_to_delete*)

Cleans labels from label stack. Set labels in rois_to_delete to background.

Parameters

- **labels** (*[type]*) – [description]
- **rois_to_delete** (*[type]*) – [description]

Returns
[description]

Return type
[type]

deepd3.core.analysis.**connected_components_3d**(*prediction*, *seeds*, *delta*, *threshold*, *distance*, *dimensions*)

Computes connected components in 3D using various constraints. Each ROI is grown from a seed pixel. From there, in a 26-neighbour fashion more pixels are added iteratively. Each additional pixel needs to fulfill the following requirements:

- The new pixel's intensity needs to be in a given range relative to the seed intensity (*delta*)
- The new pixel's intensity needs to be above a given *threshold*
- The new pixel's position needs to be in the vicinity (*distance*) of the seed pixel

Each pixel can only be assigned to one ROI once.

Parameters

- **prediction** (*numpy.ndarray*) – prediction from deep neural network
- **seeds** (*numpy.ndarray*) – seed pixels
- **delta** (*float*) – difference to seed pixel intensity
- **threshold** (*float*) – threshold for pixel intensity
- **distance** (*int or float*) – maximum euclidean distance in microns to seed pixel
- **dimensions** (*dict(float, float)*) – xy and z pitch in microns

Returns
the labelled stack and the number of found ROIs

Return type
tuple(labels, N)

deepd3.core.analysis.**getROIsizes**(*labels*)

Get the ROI size for each label with one single stack pass

Parameters
labels (*numpy.ndarray*) – label map

Returns
the size of each ROI area

Return type
numpy.ndarray

deepd3.core.analysis.**minMaxProbability**(*labels*, *prediction*)

Computes the minimum and maximum probabilty of a prediction map given a label map

Parameters

- **labels** (*numpy.ndarray*) – labels
- **prediction** (*numpy.ndarray*) – prediction map with probabilities 0 … 1

Returns
for each label the minimum and maximum probability

Return type
numpy.ndarray

deepd3.core.analysis.**reid**(*labels*)

Relabel an existing label map to ensure continuous label ids

Parameters
labels (*numpy.ndarray*) – original label map

Returns
re-computed label map

Return type
numpy.ndarray

3.2 dendrite

class deepd3.core.dendrite.**DendriteSWC**(*spacing=[1, 1, 1]*)

_binarize_swc_w_spheres()

Binarizes SWC file in a given 3D stack with spheres

convert(*target_fn=None*)

Convert swc file to tif stack

Parameters
target_fn (*str, optional*) – Target path. Defaults to None.

Returns
save path

Return type
return

open(*swc_fn*, *ref_fn*)

Open and read the swc and the stack file.

Parameters

- **swc_fn** (*str*) – The file path to the swc file
- **ref_fn** (*str*) – The file path to the stakc file

deepd3.core.dendrite.**line_w_sphere**(*s*, *p0*, *p1*, *r0*, *r1*, *color=1*, *spacing=[1, 1, 1]*)

Draw a line with width in 3D space

Parameters

- **s** (*numpy.ndarray*) – the 3D stack
- **p0** (*tuple*) – point 0 (x, y, z)
- **p1** (*tuple*) – point 1 (x, y, z)
- **r0** (*float*) – radius for point 0
- **r1** (*float*) – radius for point 1
- **color** (*int, optional*) – Color for drawing, e.g. 255 for np.uint8 stack. Defaults to 1.
- **spacing** (*list, optional*) – Spacing in 3D (x,y,z). Defaults to [1, 1, 1].

deepd3.core.dendrite.**sphere**(*s*, *p0*, *d*, *spacing=[1, 1, 1]*, *color=255*, *debug=False*)

Draw a 3D sphere with given diameter d at point p0 in given color.

Parameters

- **s** (*numpy.ndarray*) – numpy 3D stack
- **p0** (*tuple*) – x, y, z tuple
- **d** (*float*) – diameter in 1 spacing unit
- **spacing** (*list, optional*) – x, y, z spacing; x and y spacing must be equal. Defaults to [1, 1, 1].
- **color** (*int, optional*) – Draw color, e.g. 255 for np.uint8 stack. Defaults to 255.
- **debug** (*bool, optional*) – if True prints plane related information. Defaults to False.

deepd3.core.dendrite.**xyzr**(*swc*, *i*)

returns xyz coordinates and radius as tuple from swc pandas DataFrame and loc i, actually it is y, x and z

Parameters

- **swc** (*pandas.DataFrame*) – List of traced coordinates
- **i** (*int*) – current location

Returns
y, x, z and r coordinates as integers

Return type
tuple

3.3 spines

class deepd3.core.spines.**Spines**

convert()

Loads and converts spine annotation files to TIFF stacks

Returns
Path to saved TIFF stack

Return type
str

open(*spines_fn: str*)

Saves path to object

Parameters
spines_fn (*str*) – path to spines annotation file

3.4 export

class deepd3.core.export.**ExportCentroids**(*roi_centroids*)

export(*fn*)

Exports ROIs to file

Parameters
fn (*str*) – target filename and location

class deepd3.core.export.**ExportFolder**(*rois*)

export(*fn*, *folder*)

Export ROIs to folder

Parameters

- **fn** (*str*) – file name
- **folder** (*str*) – target folder

class deepd3.core.export.**ExportImageJ**(*rois*)

export(*fn*)

Export ROIs to ImageJ ROI zip file

Parameters
fn (*str*) – path to zip file

class deepd3.core.export.**ExportPredictions**(*pred_spines*, *pred_dendrites*)

export(*fn*, *folder*)

Export predictions as tif files

Parameters

- **fn** (*str*) – file name
- **folder** (*str*) – target folder

class deepd3.core.export.**ExportROIMap**(*roi_map*, *binarize=False*)

export(*fn*)

Export predictions as tif files

Parameters

- **fn** (*str*) – file name
- **folder** (*str*) – target folder

3.5 distance

deepd3.core.distance.**_computeDistance**(*p1*, *p2*, *dxy=0.1*, *dz=0.5*)

compute euclidean distance of two points in space. Points are in (Z, Y, X) format

deepd3.core.distance.**_countOccurences**(*arr*) → dict

Count occurences in array

Parameters
arr (*numpy.ndarray*) – Array with non-unique numbers

Returns
Dictionary with unique numbers as keys and their occurence as value

Return type
dict

deepd3.core.distance.**_distanceMatrix**(*pt1*, *pt2*, *dxy=0.1*, *dz=0.5*) → ndarray

Compute distance matrix of points in 3D (Z, Y, X). Works only on 3D data

Parameters

- **pt1** (*numpy.ndarray*) – Points to be matched
- **pt2** (*numpy.ndarray*) – Points that can be matched
- **dxy** (*float, optional*) – Pitch in xy. Defaults to 0.1.
- **dz** (*float, optional*) – Pitch in z. Defaults to 0.5.

Returns
distance map from pt1 and pt2 points

Return type
numpy.ndarray

deepd3.core.distance.**distanceMatrix**(*pt1*, *pt2*, *dxy=0.1*, *dz=0.5*)

Compute distance matrix of points in 2D (Y, X) and 3D (Z, Y, X)

Parameters

- **pt1** (*numpy.ndarray*) – Points to be matched
- **pt2** (*numpy.ndarray*) – Points that can be matched
- **dxy** (*float, optional*) – Pitch in xy. Defaults to 0.1.
- **dz** (*float, optional*) – Pitch in z. Defaults to 0.5.

CHAPTER

FOUR

DEEPD3 INFERENCE

4.1 gui

class deepd3.inference.gui.**Cleaning**

close(*self*) → bool

class deepd3.inference.gui.**DoubleSlider**(*decimals=2*, **args*, ***kargs*)

setMaximum(*self*, *a0: int*)

setMinimum(*self*, *a0: int*)

setSingleStep(*self*, *a0: int*)

setValue(*self*, *a0: int*)

singleStep(*self*) → int

value(*self*) → int

class deepd3.inference.gui.**ImageView**(**args*, ***kwargs*)

mouseDoubleClickEvent(*self*, *a0: QMouseEvent*)

mousePressEvent(*self*, *a0: QMouseEvent*)

class deepd3.inference.gui.**Interface**(*fn*, *pred_fn*, *rois_fn*, *logs_fn*, *dimensions={'xy': 0.094, 'z': 0.5}*)

_changeOverlay(*z*)

Hook for z-slider

Parameters
z (*int*) – current z index

changeOverlay(*z*, *preview=False*)

When the z-slider is changed, update the overlay image (i.e. the prediction)

Parameters
z (*int*) – current z-location in stack

getSelection()

Sets the current row selection in table and updates ROIs, because the selected ROI has a different color.

keyPressEvent(*self*, *a0: QKeyEvent*)

populateTable()

Populates ROI table

roiSelection(*xy*)

Highlight selected ROI due to left click

Parameters
xy (*QPoint*) – Clicking location

saveSettingsROI3D(*settings*)

Save test settings to global settings

Parameters
settings (*dict*) – 3D ROI settings

testROIbuilding(*xy*)

Test ROI building using dedicated interface. Interface is opened at particular stack location where user double-clicked

Parameters
xy (*QPoint*) – XY Location of pointer during click

updateProgress(*pval*, *pmax*)

updates progress bar

Parameters

- **pval** (*int*) – current value
- **pmax** (*int*) – target value

class deepd3.inference.gui.**Main**

cleaning()

Clean the prediction using user-specified settings

exportImageJ()

Export ROIs to ImageJ

exportPredictions()

Export neural network prediction as TIFF stacks

exportRoiCentroids()

Export ROI centroids as CSV file

exportRoiMap()

Export ROI map as TIFF stack

exportToFolderStructure()

Export ROIs as folder structure

importAnnotations()

Import annotations to visualize those on the central widget

open()

Open a z-stack for inference.

If a prediction and/or ROIs already exist, do load these as well.

previewCleaning(*settings*)

Preview cleaning settings to specify the settings

Parameters
settings (*dict*) – cleaning settings

roi2d()

Create ROIs from segmentation

roi3d()

Create ROIs from segmentation in 3D

save()

Save segmentation predictions and ROIs

segment()

Segment stack using user-defined settings

setDimensions()

Set dimensions for z-stack to ensure proper functionality (e.g. distance measures)

setShowLabels()

Toggles the visualization of labels on central widget

setShowMaxProjection()

Shows maximum projection of stack and prediction in central widget

setShowROIs()

Toggle ROIs on central widget

setShowSegmentation()

Toggle the segmentation visualization on central widget

zprojection()

Show a maximum and summed intensity z-projection for the full stack in separate windows

class deepd3.inference.gui.**QHLine**

class deepd3.inference.gui.**ROI2D**

close(*self*) → bool

class deepd3.inference.gui.**ROI3D**(*settings=None*)

close(*self*) → bool

class deepd3.inference.gui.**Segment**(*model_fn=None*)

close(*self*) → bool

findModel()

Find TensorFlow/Keras model on file system

class deepd3.inference.gui.**askDimensions**(*xy=0.094*, *z=0.5*)

dimensions()

Returns dictionary containing xy and z dimensions in µm

Returns
returns xy and z dimensions

Return type
dict

class deepd3.inference.gui.**testROI**(*stack*, *d*, *s*, *settings=None*)

do()

Actually generating ROIs

CHAPTER

FIVE

DEEPD3 MODEL

5.1 builder

deepd3.model.builder.**DeepD3_Model**(*filters=32*, *input_shape=(128, 128, 1)*, *layers=4*, *activation='swish'*)

DeepD3 TensorFlow Keras Model. It defines the architecture, together with the single encoder and dual decoders.

Parameters

- **filters** (*int, optional*) – Base filter multiplier. Defaults to 32.
- **input_shape** (*tuple, optional*) – Image shape for training. Defaults to (128, 128, 1).
- **layers** (*int, optional*) – Network depth layers. Defaults to 4.
- **activation** (*str, optional*) – Activation function used in convolutional layers. Defaults to "swish".

Returns
function TensorFlow/Keras model

Return type
Model

deepd3.model.builder.**convlayer**(*x*, *filters*, *activation*, *name*, *residual=None*, *use_batchnorm=True*)

Convolutional layer with normalization and residual connection

Parameters

- **x** (*Keras.layer*) – input layer
- **filters** (*int*) – filters used in convolutional layer
- **activation** (*str*) – Activation function
- **name** (*str*) – Description of layer
- **residual** (*Keras.layer, optional*) – Residual layer. Defaults to None.
- **use_batchnorm** (*bool, optional*) – Use of batch normalization. Defaults to True.

Returns
Full convolutional procedure

Return type
Keras.layer

deepd3.model.builder.**decoder**(*x*, *filters*, *layers*, *to_concat*, *name*, *activation*)

Decoder for neural network.

Parameters

- **x** (*Keras layer*) – Start of decoder, normally the latent space
- **filters** (*int*) – The filter multiplier
- **layers** (*int*) – Depth layers to be used for upsampling
- **to_concat** (*list*) – Encoder layers to be concatenated
- **name** (*str*) – Description of the decoder
- **activation** (*str*) – Activation function used in Decoder

Returns
Full decoder across layers

Return type
Keras layer

deepd3.model.builder.**identity**(*x*, *filters*, *name*)

Identity layer for residual layers

Parameters

- **x** (*Keras.layer*) – Keras layer
- **filters** (*int*) – Used filters
- **name** (*str*) – Layer description

Returns
Identity layer

Return type
Keras.layer

CHAPTER

SIX

DEEPD3 TRAINING

6.1 generator

class deepd3.training.generator.**Arrange**

addData()

Add selected d3data files

createSet()

Create d3set from selected d3data files.

keyPressEvent(*self*, *a0: QKeyEvent*)

removeSelection()

Remove selected d3data sets

class deepd3.training.generator.**ImageView**(**args*, ***kwargs*)

keyPressEvent(*self*, *a0: QKeyEvent*)

class deepd3.training.generator.**Selector**

arrangeTrainingData()

Arrange training data (d3data files) in a d3set

createTrainingData()

Create d3data set

viewTrainingData()

View training data

class deepd3.training.generator.**Viewer**(*fn*)

plane()

Overlay current annotation plane

save()

Save d3data set

class deepd3.training.generator.**addStackWidget**

changeOverlay()

Show the dendrite and spine annotations as overlay in addition to the original stack

keyPressEvent(*a0: QKeyEvent*) → None

Key press event to enable shortcuts

Parameters
a0 (*QKeyEvent*) – Key event

save()

Save a d3data set

selectDendrite()

Select dendrite annotation file

selectSpines()

Select a spine annotation

selectStack()

Select a microscopy stack

updateProgress(*a*, *b*)

Update progress bar

Parameters

- **a** (*int, float*) – maximum of progress bar
- **b** (*int, float*) – current value of progress bar

updateROI()

Updates the ROI chosen as dataset

updateZ(*a*, *b*)

Updates z-level in stack

Parameters

- **a** (*int*) – z-stack begin, first plane
- **b** (*int*) – z-stack end, last plane

class deepd3.training.generator.**askSpacing**

spacing()

Converts spacing

Returns
spacing in µm in x, y and z

Return type
tuple(float, float, float)

deepd3.training.generator.**main**()

Main entry point to GUI

6.2 stream

class deepd3.training.stream.**DataGeneratorStream**(*fn*, *batch_size*, *samples_per_epoch=50000*, *size=(1, 128, 128)*, *target_resolution=None*, *augment=True*, *shuffle=True*, *seed=42*, *normalize=[-1, 1]*, *min_content=0.0*)

_getSample(*squeeze=True*)

Retrieves a sample

Parameters
squeeze (*bool, optional*) – Squeezes return shape. Defaults to True.

Returns
Tuple of stack (X), dendrite (Y0) and spines (Y1)

Return type
tuple

_get_augmenter()

Defines used augmentations

getSample(*squeeze=True*)

Get a sample from the provided data

Parameters
squeeze (*bool, optional*) – if plane is 2D, skip 3D. Defaults to True.

Returns
stack image with respective labels

Return type
list(np.ndarray, np.ndarray, np.ndarray)

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX

Symbols

A

C

D

E

F

G

I

K

L

M

O

P

Q

R

S

T

U

V

X

Z