

Annotat3D

Tutorial

Scientific Computing Group (GCC)

November/2019

Documentation History

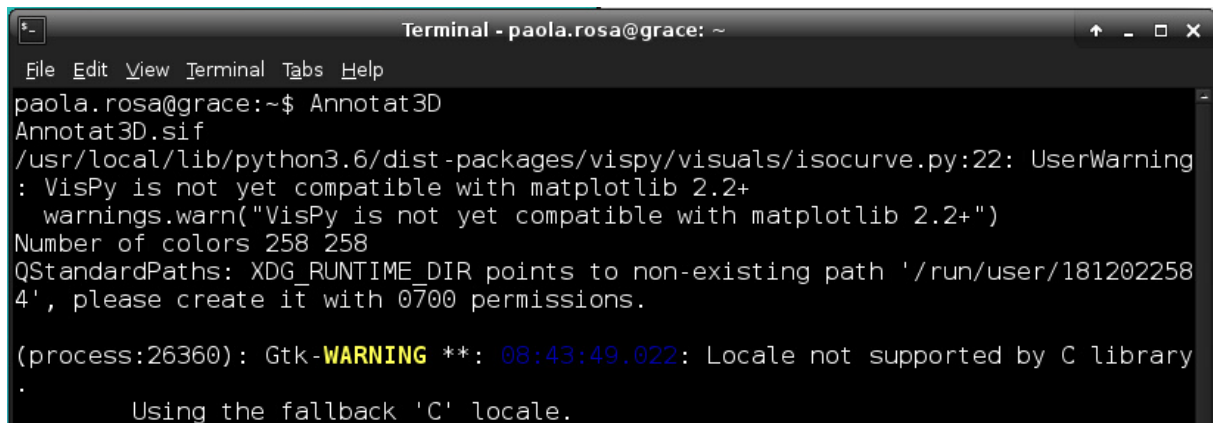
Date	Revision	Description	Author
30/01/2018	1	Version based on Annotat3D 1.	Giovanna Antonieti
30/08/2019	2	Version based on Annotat3D 2.	Paola R. R. Rosa
03/11/2019	3	Version based on Annotat3D - DeepSirius	Paola R. R. Rosa

Summary

I. Opening Annotat3D and Loading files	1
II. Menus	2
II. I File	2
II. II Edit	2
II. III View	3
III. Visualization Options	4
IV. Annotation Options	9
V. Segmentation Module and Classification Menu	11
V. I Feature Extraction	12
V. II Superpixels	13
VI. Segmentation	15
VI. I First Segmentation	15
VI. II Loading and editing a previous label	18
VII. Keyboard Shortcuts	19
VIII. Deep Learning Menu	19
VIII. I What is Deep Learning	19
VIII. II What you need before using Deep Learning	20
VIII. III Workspace Submenu	20
VIII. IV Dataset Submenu	21
VIII. IV. I Sampling	21
VIII. IV. II Augmentation	25
VIII. V Network Submenu	30
VIII. VI Batch Inference Submenu	35

9I. Opening Annotat3D and Loading files

After logging into <<tarsila or ada>> via *TurboVNC*, open a terminal and type “Annotat3D”.



```
Terminal - paola.rosa@grace: ~
File Edit View Terminal Tabs Help
paola.rosa@grace:~$ Annotat3D
Annotat3D.sif
/usr/local/lib/python3.6/dist-packages/vispy/visuals/isocurve.py:22: UserWarning
: VisPy is not yet compatible with matplotlib 2.2+
  warnings.warn("VisPy is not yet compatible with matplotlib 2.2+")
Number of colors 258 258
QStandardPaths: XDG_RUNTIME_DIR points to non-existing path '/run/user/181202258
4', please create it with 0700 permissions.

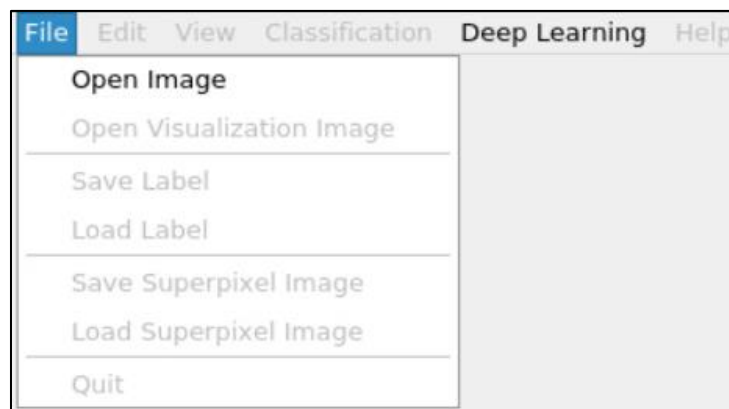
(process:26360): Gtk-WARNING **: 08:43:49.022: Locale not supported by C library
.
Using the fallback 'C' locale.
```

Alternative paths:

/ddn/GCC/apps/Annotat3D

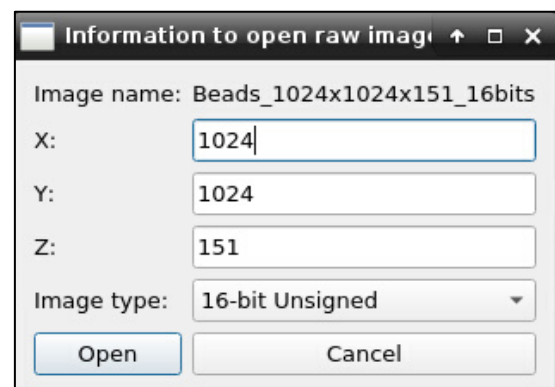
/ssd/apps/Annotat3D

First, load an image into the application. Select the image in **Menu>File>Open Image** located in the upper left corner.

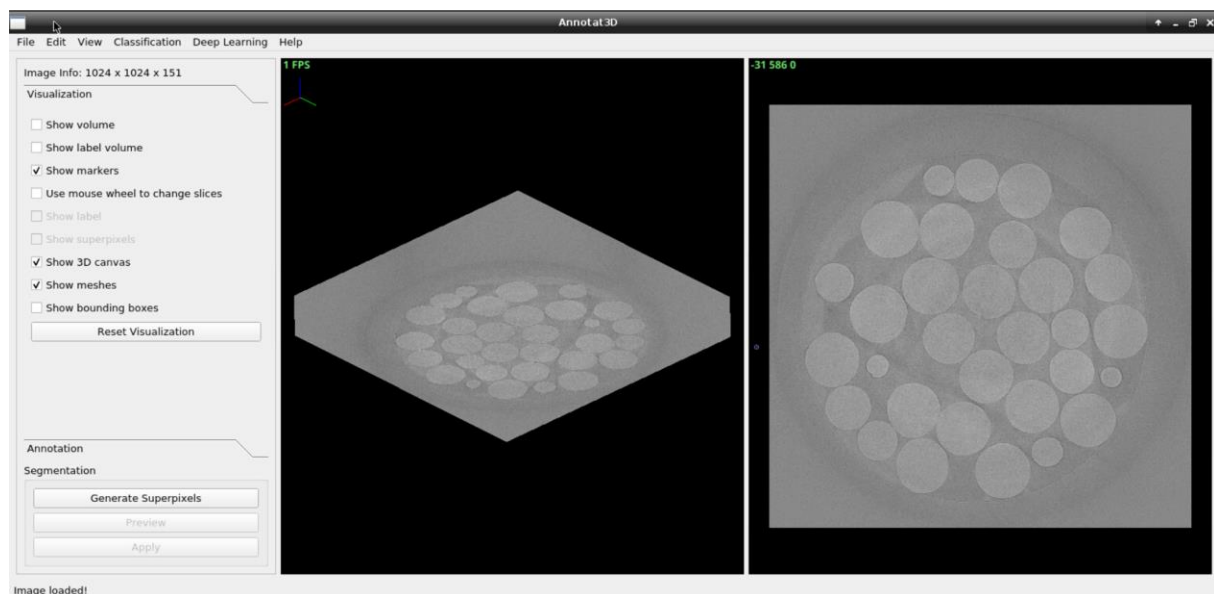


After selecting the option to load the image, a window to select the file will open. At the bottom of this window you can select the type of image you want to open; you can choose **tiff** or **raw**.

- If you select a **tiff image**, it will load directly into the application.
- If the selected image is **raw** it will open a third window, like the image below, where you should place the dimensions of the image and the type of the image.

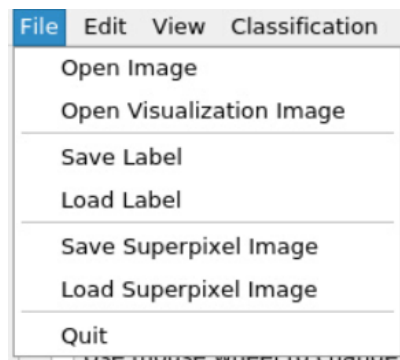


The result should be presented as the data below.



II. Menus

II. I File

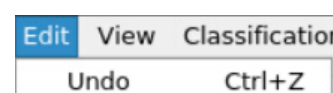


Open visualization image: Can be used if you need to open a secondary image that can be used for visualization only. For instance, one might be interested in segmenting an image in its raw form but performing visualization/annotation by also considering a treated/filtered version of the same image, in order to facilitate the process of understanding what is present in the original image. *Obs: all computations are done in the original image that is loaded, so the visualization image is only considered for display.*

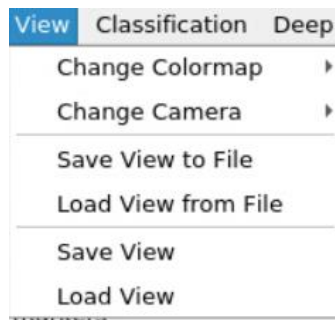
If you have already segmented data (see section VI. II), use the other commands to *load* or *save* your label/Superpixel image.

II. II Edit

At edit menu you can undo the previous action.

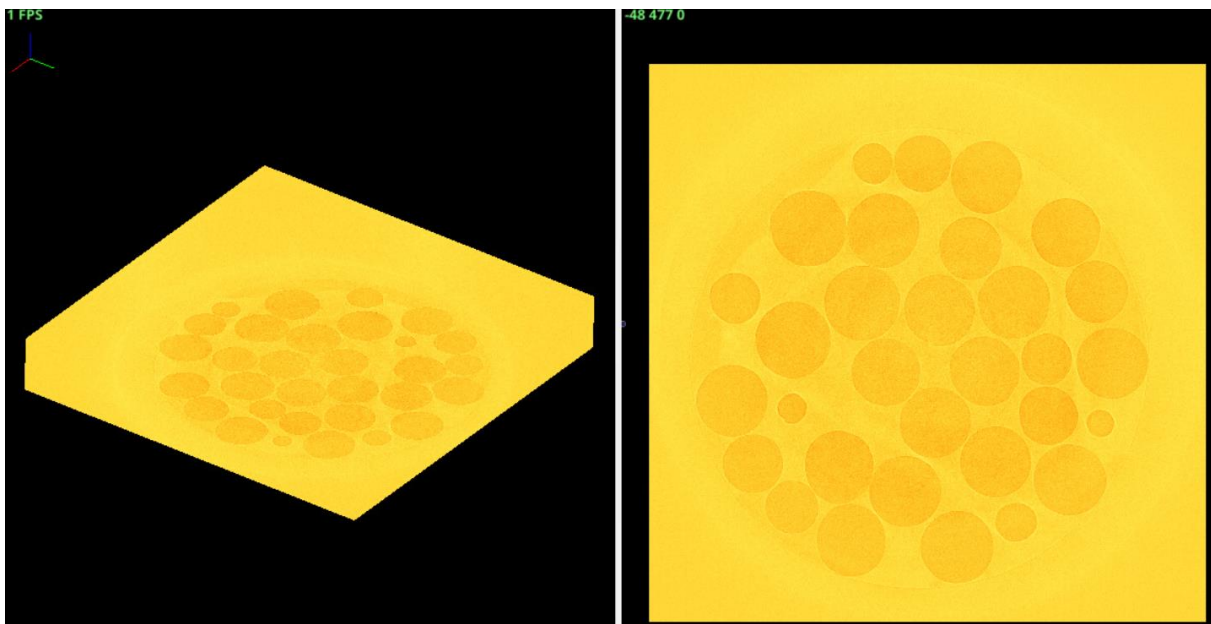


II. III View



Change colormap: Allows visualization of 3D and 2D images in different colormaps:

Grays, Viridis, Fire, Diverging, HSL, HUSL, SingleHue, CubeHelix, Winter, Hot, Ice, Autumn, Blues, Cool, Greens, Reds, Springs, Summer, LightBlues, Orange, Coolwarm, PuGr, GrBu, GrBy_d, RdBu, RdYeBuCy.



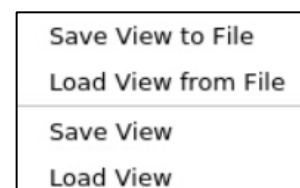
Change camera: The camera options are:

- 3D Arcball
- 3D Turntable
- 3D Fly

Save view: Sets the current position as the new home position.

Load view: Resets the camera to the home position.

Save view to file/Load view from file: If you export the home position, you can import it in a new Annotat3D project.



III. Visualization Options

Image Info: 1024 x 1024 x 151

Visualization

☐ Show volume

☐ Show label volume

☒ Show markers

☐ Use mouse wheel to change slices

☐ Show label

☐ Show superpixels

☒ Show 3D canvas

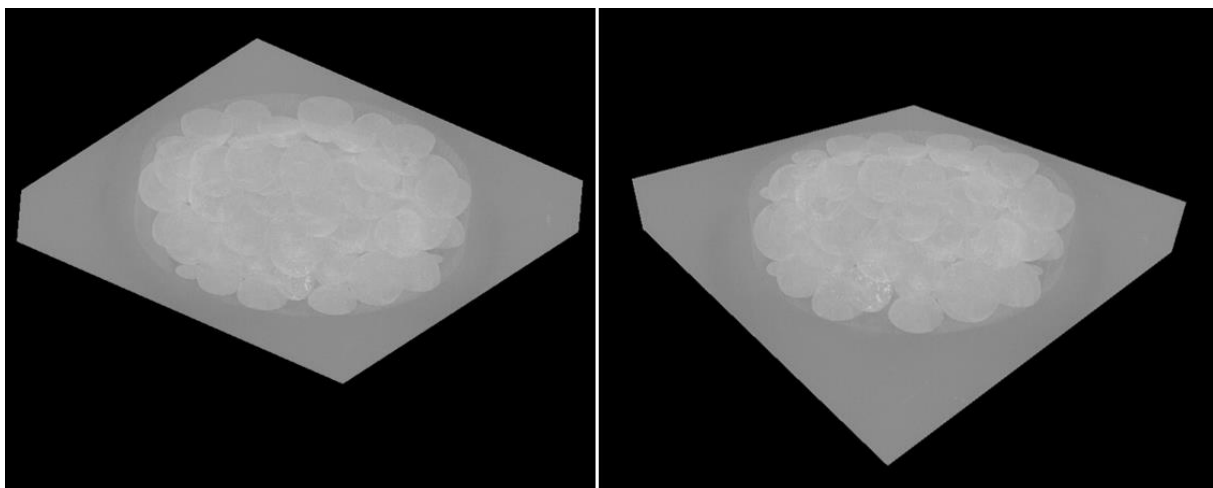
☒ Show meshes

☐ Show bounding boxes

Reset Visualization

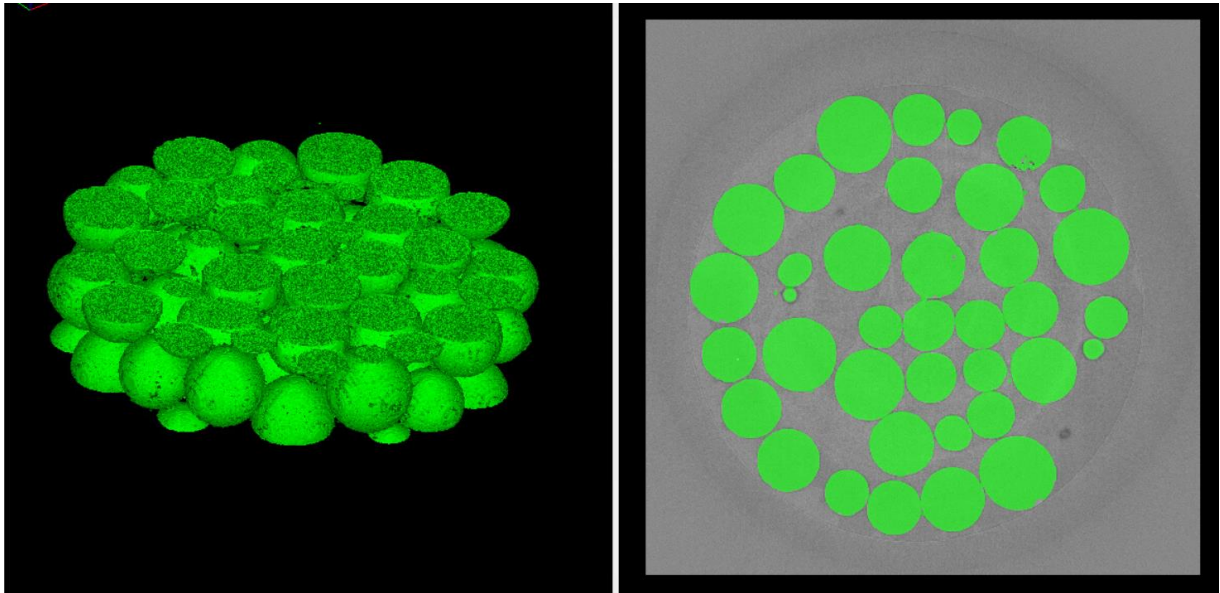
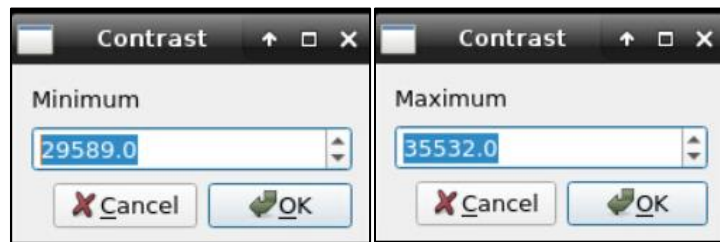
Show volume: Generates a 3D volume of your data. Unclick *show slices* (eye buttons at slices XY, XZ, YZ) to see the volume. You can set the threshold to show a specific portion of the volume. The 3D volume will be generated in an orthographic view. For Perspective view click Shift + drag the mouse upwards while holding the right button.

Note: this method may take a long time to render large images (e.g., greater than 1024x1024x1024).

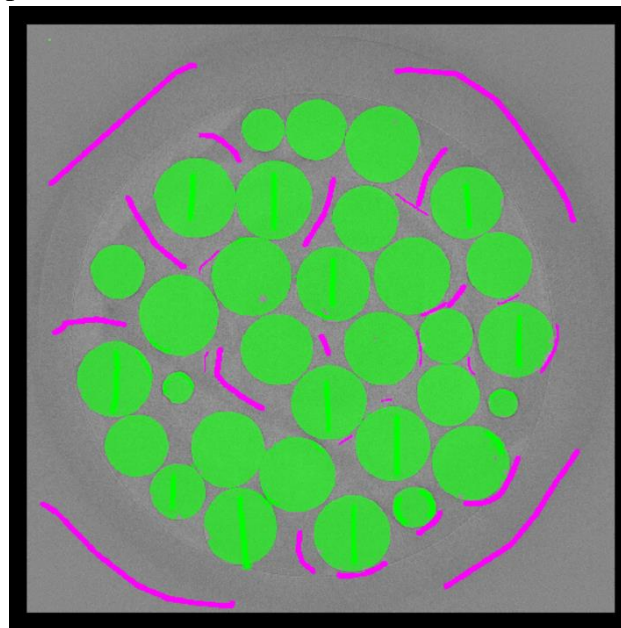


Orthographic (left) and Perspective view (right)

Show label volume: Generates a 3D volume of your label. Unclick *show slices* (eye buttons at slices XY, XZ, YZ) to see the volume.

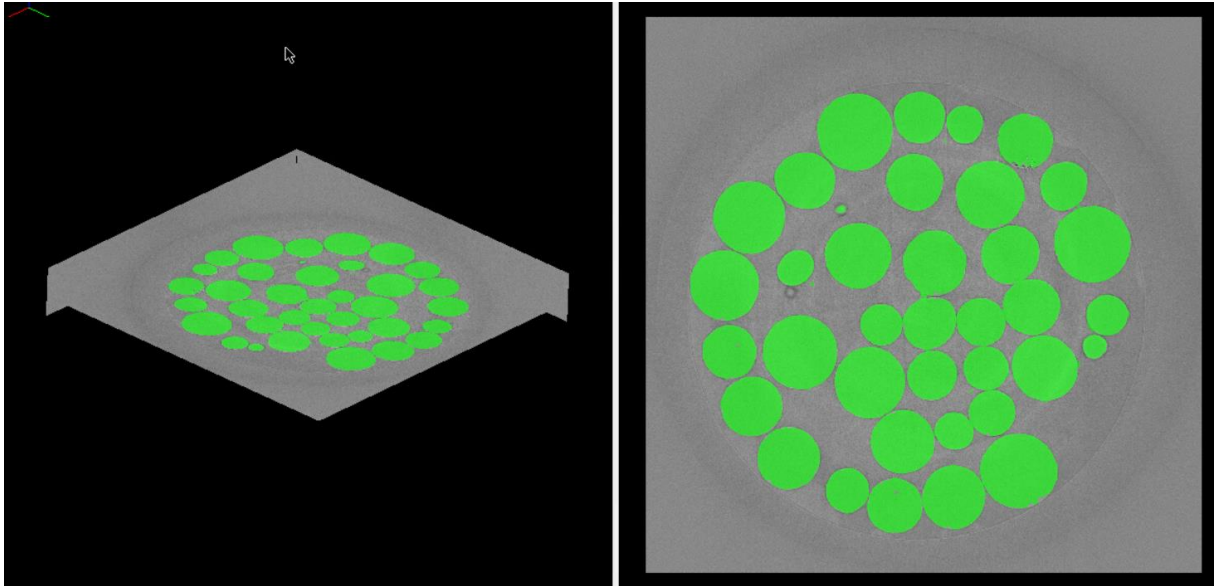


Show markers: This option makes the markers visible.

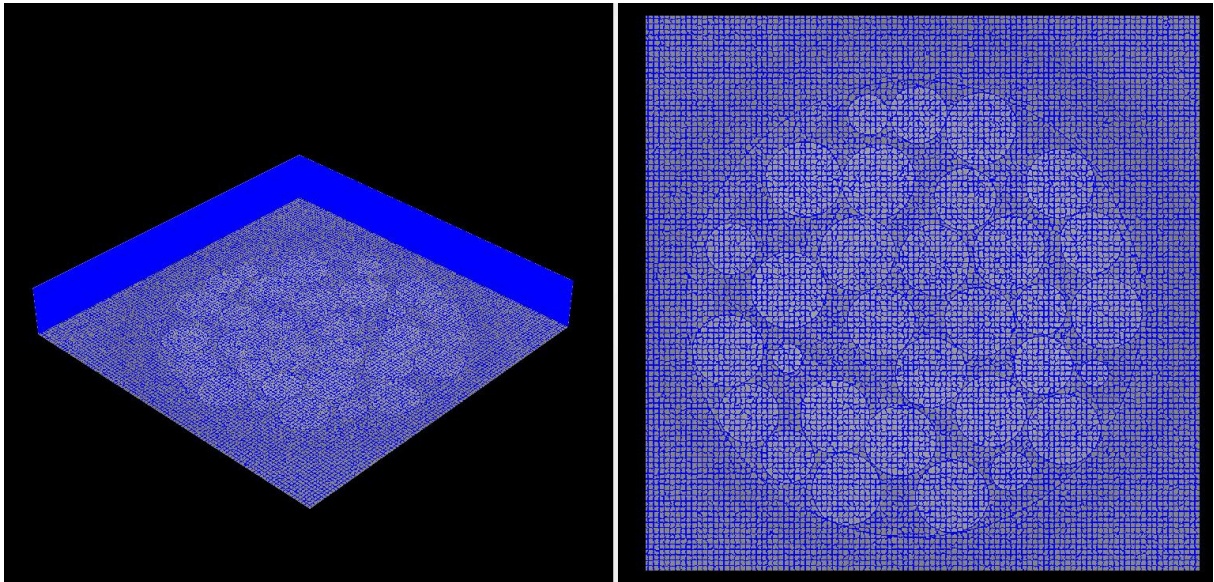


Use mouse wheel to change slices: When enabled, scroll the mouse wheel at 3D view to change slices. When disabled, the mouse wheel at 3D view will apply zoom.

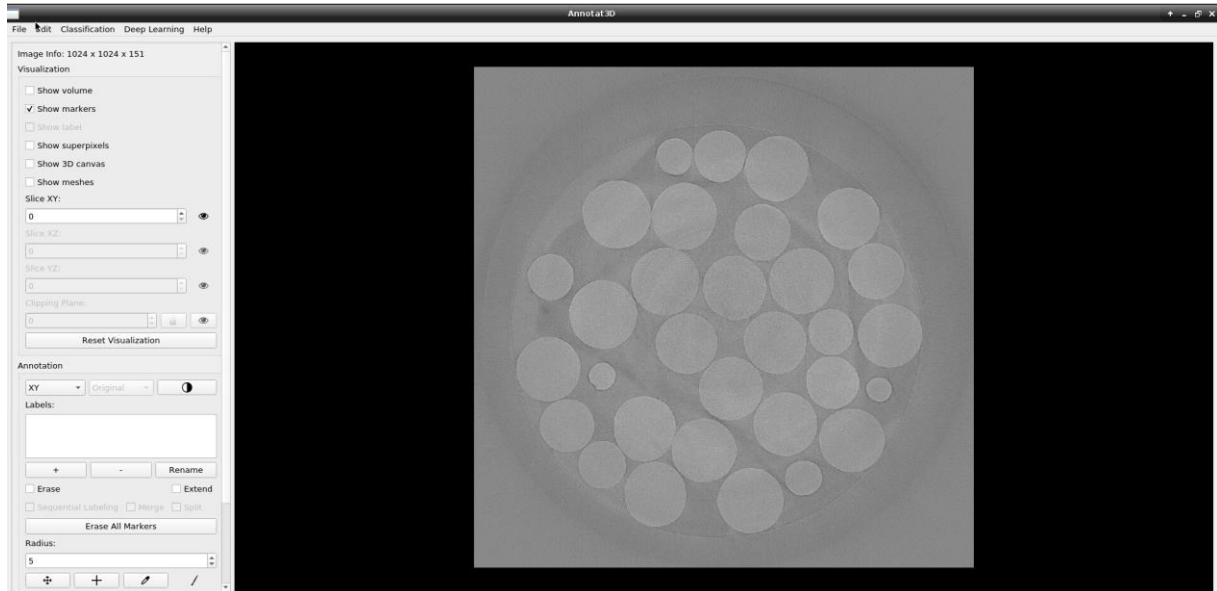
Show label: Available after the segmentation. This option makes the label visible (toggles the overlay of your classification).



Show superpixels: Available after the generation of superpixels.

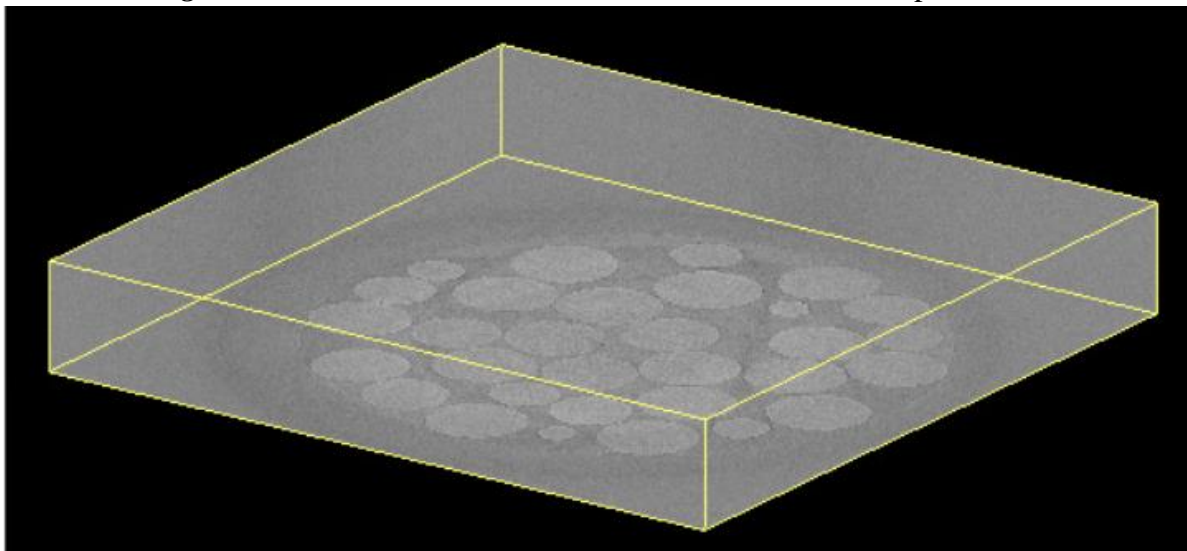


Show 3D canvas: If not enabled, only the 2D view will be available.

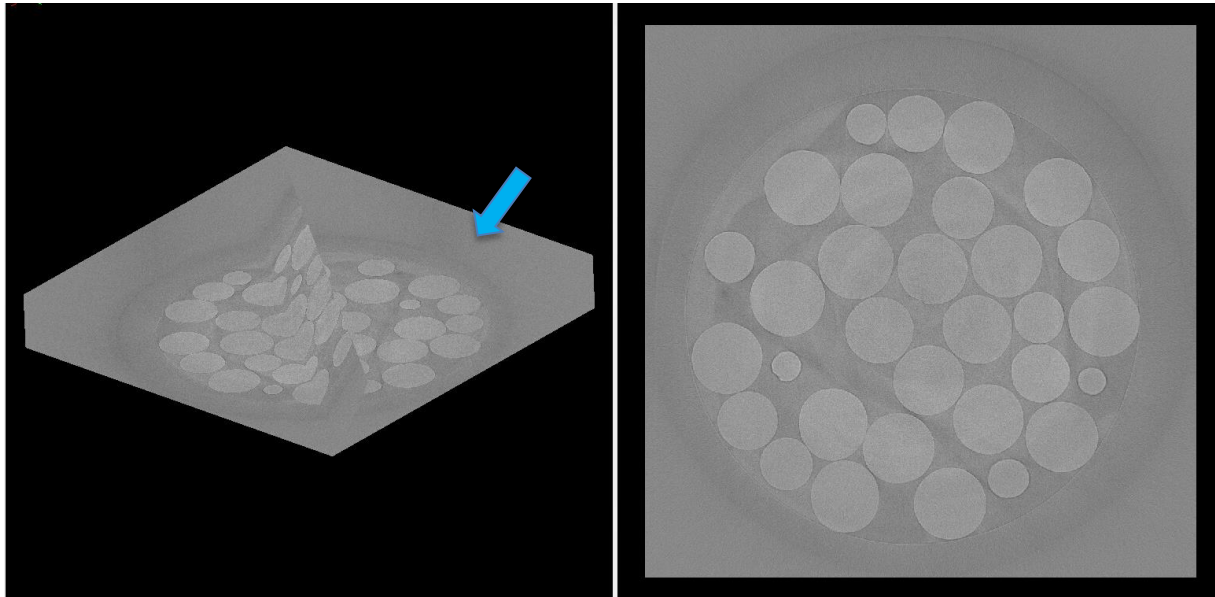


Show meshes: This option will render only the selected label as a 3D meshes. First you need to select the label. For this, click Ctrl + Material.

Show bounding boxes: It enables a box that shows the limits of the sample.



Clipping Plane: Click on the eye to view the plane and move the left image (3D view) of the screen to bring the axis to the desired region. When you find the correct position, click on the lock to secure the axis position.

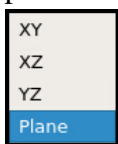


Tip: If you have more than two labels (e. g. several different cells), use clipping plane to see an individual material better. For this, select the material (ctrl+shift) and center the material (Center button **+**, Annotation submenu). Then, remove the planes (click on the eyes) and configure the clipping plane that best fits to your material.

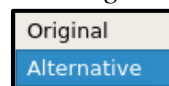
IV. Annotation Options




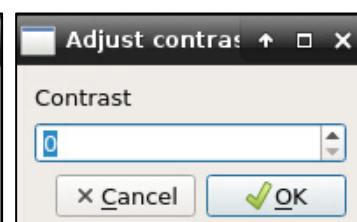
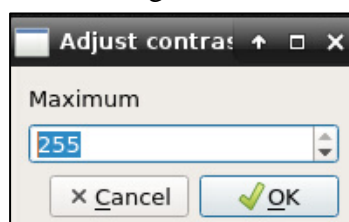
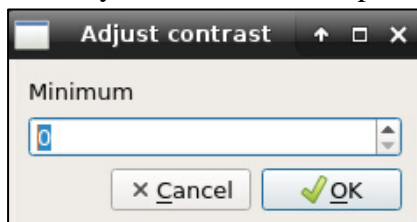
Plane view: Allows you to change the annotation plane. Plane stands for clipping plane.



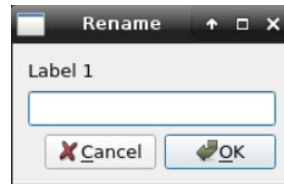
Selected image: Selects the image that will be used for visualization. *It will only be activated if you opened a visualization image. Obs: The segmentation methods do NOT consider the visualization image in their algorithms.*



Contrast: Adjust the contrast by clicking on the circular button . If you want to undo your action, you need to set the parameters to original value.



- , + , **Rename**: Use it to remove a label, add a label or modify the name of the selected label.

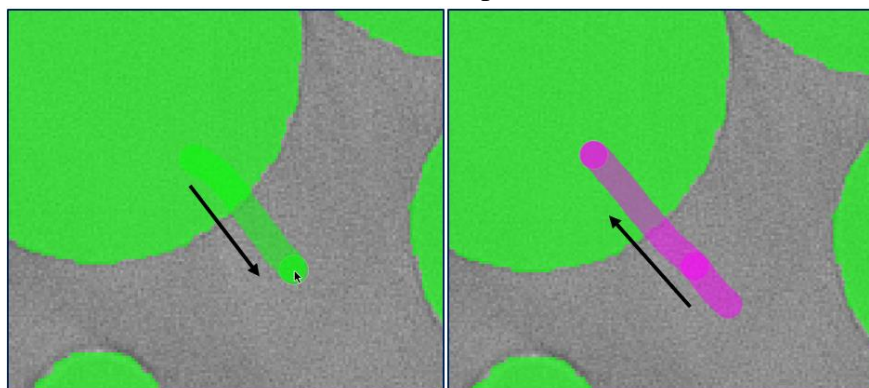


Erase Mode: Use it to delete markers with left click selection. You can also click Ctrl+z to undo last marker. If you need to remove an entire marker at time, click on it with the right mouse button.

You can also click: Edit > Undo

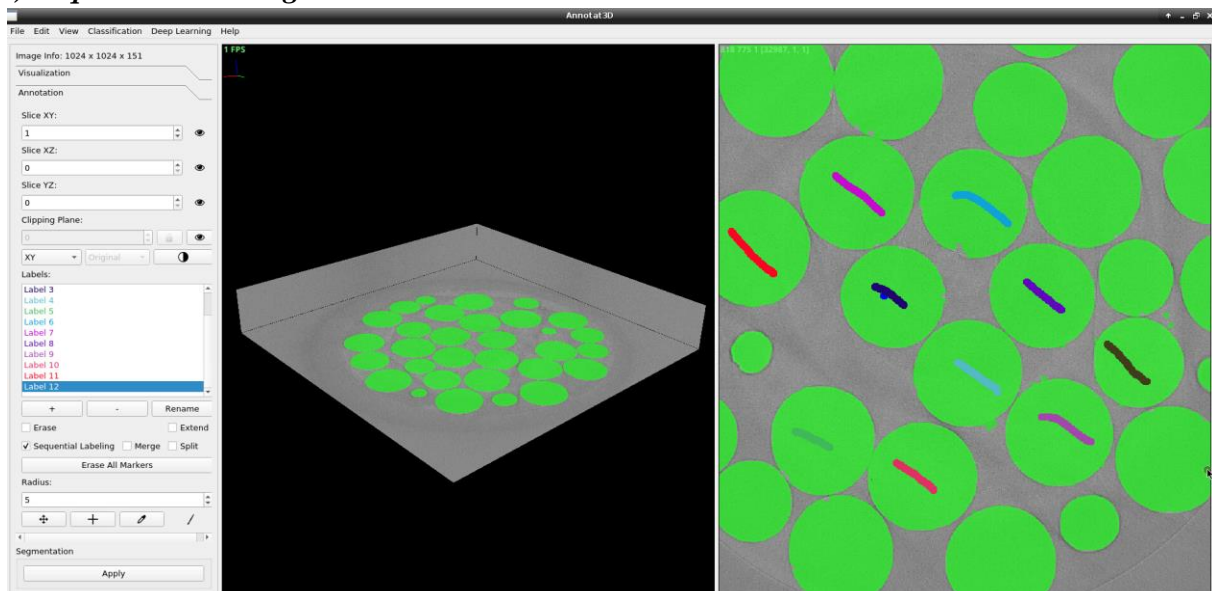


Extend Label: Instead of selecting the label from the menu, this option allows the label to be extended from its initial click. That is, if one clicks on the image where it is classified as "A" and drags the mouse, the whole marker will correspond to "A".




[For Correction of previous classifier and annotation]


i) **Sequential Labeling**: Each click creates a different label.

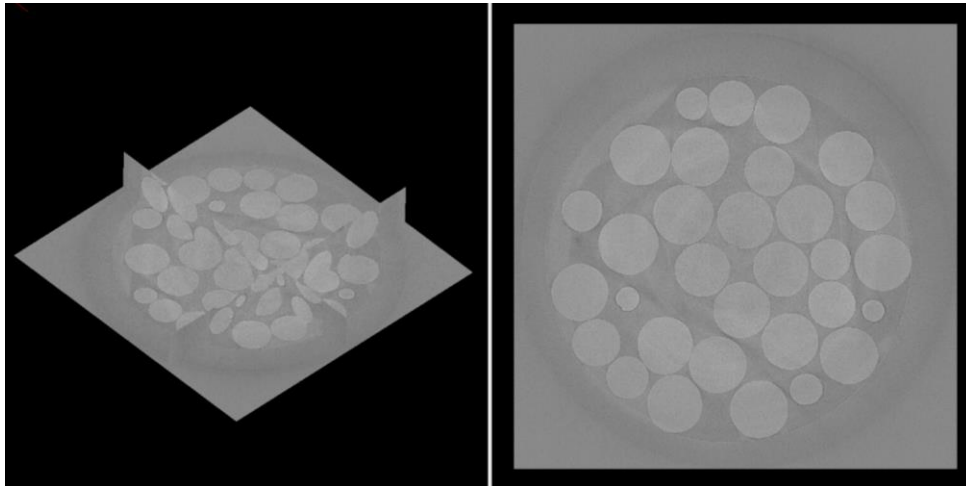


ii) **Merge Labels**: Puts together two labels based on the first label.

Radius: Determines the size of the brush.

Position button : Allows you to modify the position on 3D (rotation) and 2D (translation) canvas via drag and drop.

Center button : Centers the visualization ortho-slices on the clicked location. Click on the position that you want to center in 2D view and it will center the planes on the 3D view. If you want to put your axis in the material center, select Ctrl+Shift+material.

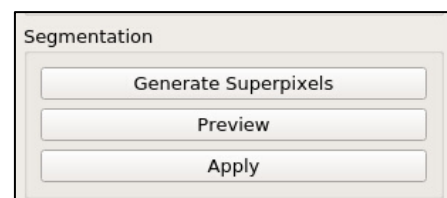


Material button : Selects the label clicked by the user on the annotation canvas.

Painting button : Activates scribble drawing mode.

V. Segmentation Module and Classification Menu

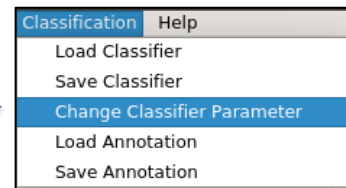
Before defining the labels and adding the respective markers, you need to select **Generate Superpixels** and then **Apply**. After the first classification, there is no need to Generate Superpixels again, except if you change the classifier parameters. If you want to save your superpixel classification to use it later (after you exit the software), click on: **File > Save Superpixel Image**, otherwise you'll have to generate it again when you enter the software.



Note: The superpixels should be adjusted to fit their boundaries corresponding to the edge of the objects in the image in order to give a good classification.

Preview: Should be used for larger images (e.g., greater than 1024x1024x1024), especially when a great number of filters have been selected for computation. Preview computes feature extraction and classification for the current slide and a few others before/after it.

To change the superpixel classification, go to:
Classification > Change Classifier Parameter



You will see the following table.

 A screenshot of a dialog box titled 'Change Classifier Parameters'. The dialog is divided into several sections:

- Feature Extraction:** Contains two columns of checkboxes. The first column has 'FFT Gauss' (checked), 'None (Original Img)' (checked), 'Sobel' (unchecked), 'Minimum' (unchecked), 'Average' (unchecked), 'Median' (unchecked). The second column has 'FFT Gabor' (unchecked), 'FFT DoG' (checked), 'Membrane Projections' (checked), 'Maximum' (unchecked), 'Variance' (unchecked), 'LBP' (unchecked). Below these is a text field for 'Multi-scale Filtering Windows' with the value '1, 2, 4, 8'.
- Feature Selection:** Contains a checked 'Enable?' checkbox and an 'Importance threshold' text field with the value '0.01'.
- Superpixels:** Contains a 'Waterpixels' dropdown menu, a 'Waterpixels Seed Spacing' text field with '7', and a 'Waterpixels Compactness' text field with '10000.0'. Below these is a 'Superpixel Feature Pooling' section with three checkboxes: 'Mean' (checked), 'Minimum' (unchecked), and 'Maximum' (unchecked).
- Classification:** Contains a 'Classifier' dropdown menu with 'RandomForest' selected, and a 'Random Forest N. Trees' text field with '200'.

 At the bottom of the dialog are 'Cancel' and 'Ok' buttons.

V. I Feature Extraction

Enable filter options for image processing. The filters are used to enrich the image information for machine learning-based image segmentation. They tend to capture different properties of the materials in order to facilitate their separation during segmentation. For instance, Membrane Projections filter improves classification of membrane-like structures.. Some of these filters are directly related to Multi-scale Filtering Windows (MFW), where every number of adjacencies is important to the final calculation. Here is the definition of these filters:

- **FFT Gauss (Fast Fourier Transform Gauss):** The Fast Fourier Transform (FFT) is an algorithm to compute the Fourier coefficients of a finite sequence. In this case it performs n individual convolutions with Gaussian kernels with the normal n variations of MFW. That means: \uparrow radius diameter \downarrow blurred the image becomes until the pixels are homogeneous.
- **None (Original Img).**
- **FFT Gabor:** It determines if there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. In the spatial domain, it is a Gaussian kernel function modulated by a sinusoidal plane wave.
- **FFT DoG (Fast Fourier Transform Difference of Gaussians):** Calculates two Gaussian blur images from the original image and subtracts one from the other.
- **Sobel:** Calculates an approximation of the gradient of the image intensity at each pixel.
- **Membrane Projections:** Enhances membrane-like structures of the image through directional filtering.
- **Minimum, Maximum, Variance, Average, Median:** The pixels within a radius of MFW pixels from the target pixel are subjected to the selected operation and the target pixel is set to that value.
- **LBP - Local Binary Pattern:** It is a texture operator that tries to capture how are the neighborhoods allocated. It labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

Multi-scale Filtering Windows: The associated values indicate the radius of square windows considered around each pixel to compute the above filter extraction methods, or some value proportional to that. For instance, a sequence of "2, 4, 6" will be used by filter "Minimum" to compute the minimum pixel intensity in filtering windows with sizes of 5x5, 9x9, and 13x13 pixels, respectively (note that the *radius* of the windows are given).

Feature selection: This parameter allows an evaluation / score of the feature importance to the classification. The procedure consists of a comparison of how much accuracy is lost when the feature is removed, based on the results with the annotated data, so, it is recomputed with each new annotation. Therefore, the features contributing less than the submitted threshold value, e.g. 0.01, will be disregarded in the rating.

Note: The value is presented as percentage. If an error popup appears, the threshold set is too high and must be lowered.

V. II Superpixels

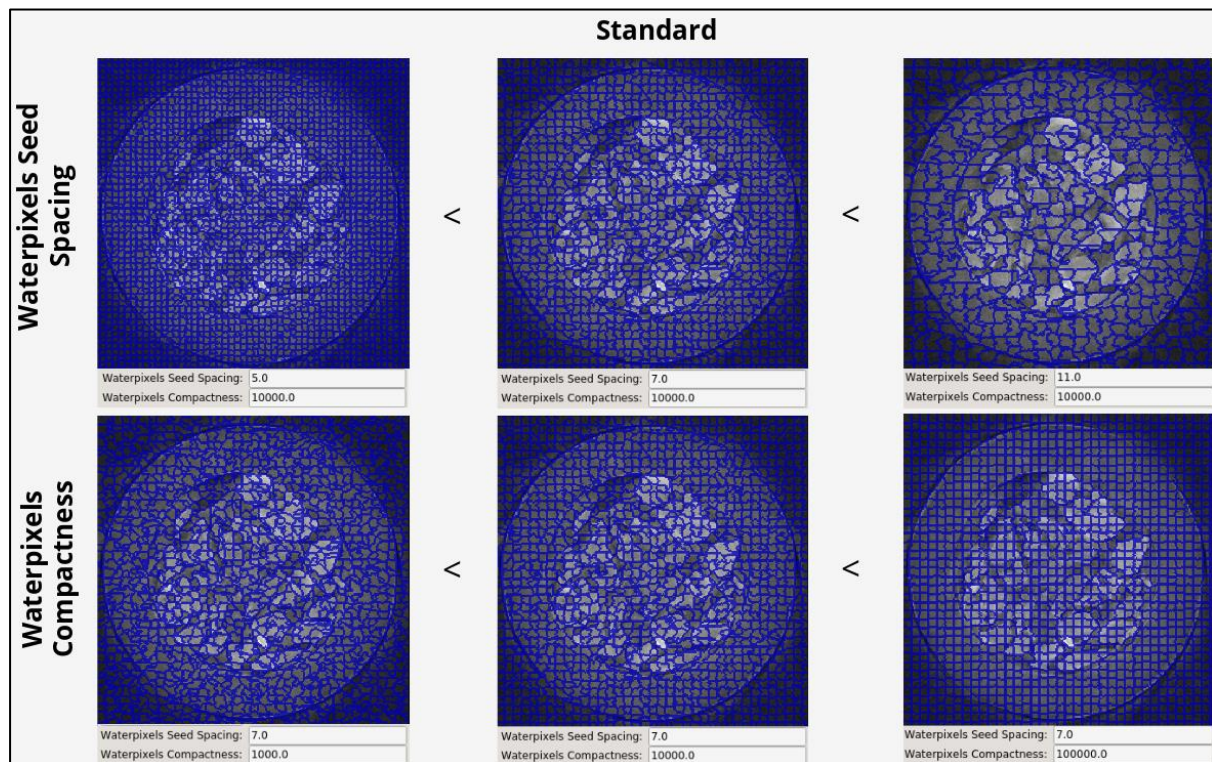
Waterpixels (3D) or Slic:

- **Waterpixels/Slic Seed Spacing:** It determines the space between the seeds. The higher the number, less superpixels are generated.
- **Waterpixels/Slic Compactness:** Increasing the number will make the superpixels more regular and will make them lose edge information.

Note: SLIC (values between 0 e 1); Waterpixels (Powers of 10, e.g., 100, 500, 1000, 10000, 20000).

Waterpixels Waterpixels Seed Spacing: 7 Waterpixels Compactness: 10000.0	SLIC SLIC Seed Spacing: 7 SLIC Compactness: 0.6
--	---

Waterpixels 3D Waterpixels Seed Spacing: 7 Waterpixels Compactness: 10000.0



Supapixel Feature Pooling: Defines how the characteristic vector will be computed. Characteristics about a local region (i.e., a super pixel) are aggregated by means of a function that reduces image information from one resource vector per pixel to a superpixel image vector.

Classification:

Classifier: There are two available: *RandomForest* and *SVM*.

- *The Random Forest Classifier* is an ensemble method that computes several random decision trees that partitions your training data (i.e., the selected superpixels) into the different classes/labels. It offers faster training times and robust classification.
- *The Linear SVM classifier* attempts to find hyperplanes that separate the training samples in the feature space. It may be faster than RandomForest, but requires the selection of parameter C to improve classification results (C is given in powers of 10 such as 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, etc.). Higher values of C (e.g, greater than 1.0) makes the error margin around

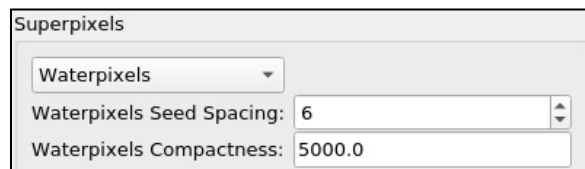
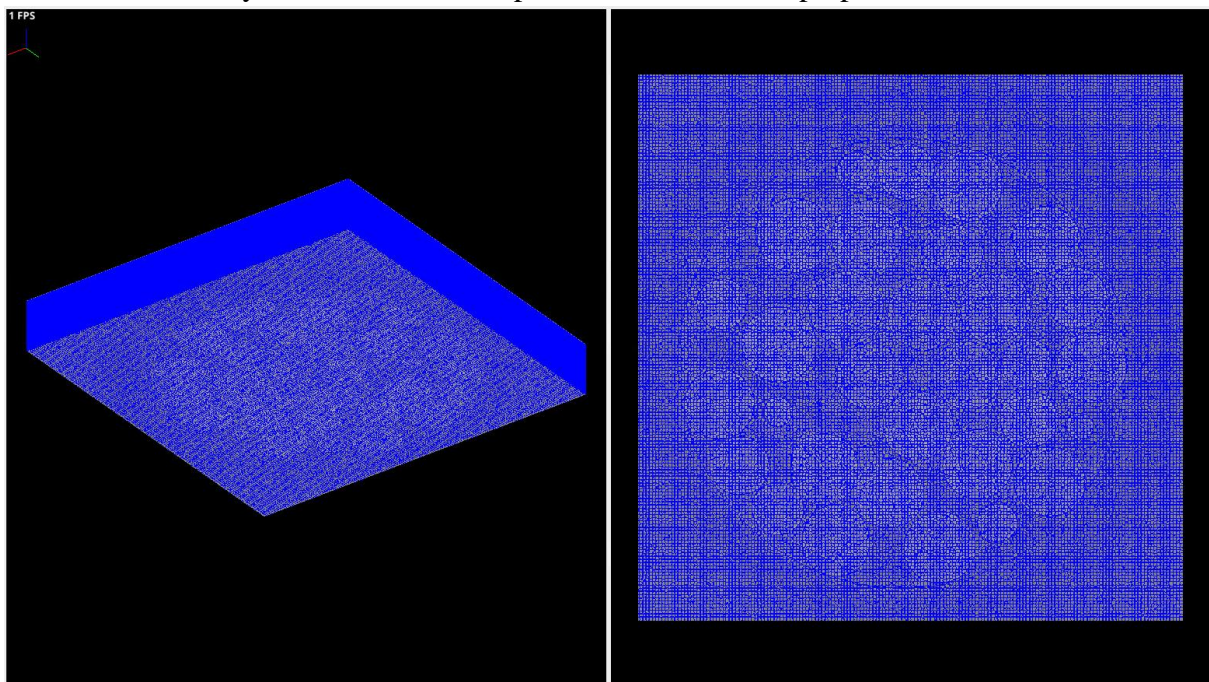
hyperplane stricter and therefore classification during training should have less errors. However, this may make the classifier harder to generalize. The user should try different values based on a fixed set of markers to determine the best value for C.

VI. Segmentation

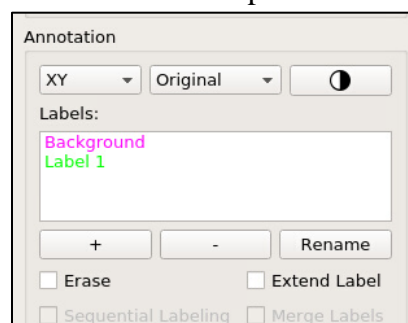
VI. I First Segmentation

In order to segment your image, you should follow these steps:

1. Select *Generate Superpixels*. Change **the classifier parameters until it gives a good adherence to the borders of the objects**. One should ideally attempt to strike a balance between boundary adherence and compactness/number of superpixels.

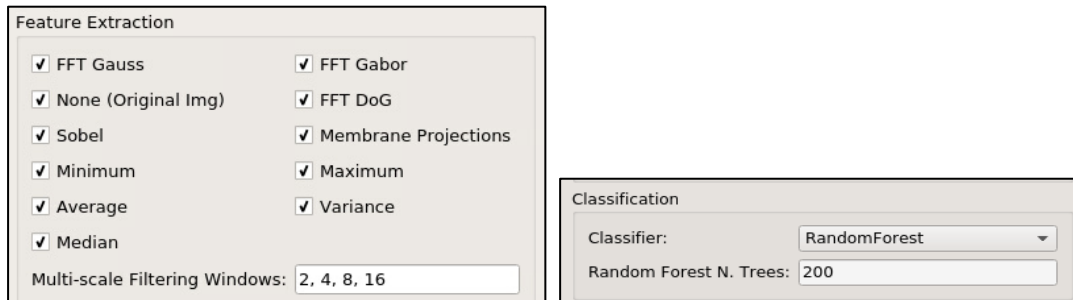


2. Add and rename the labels and add the respective markers

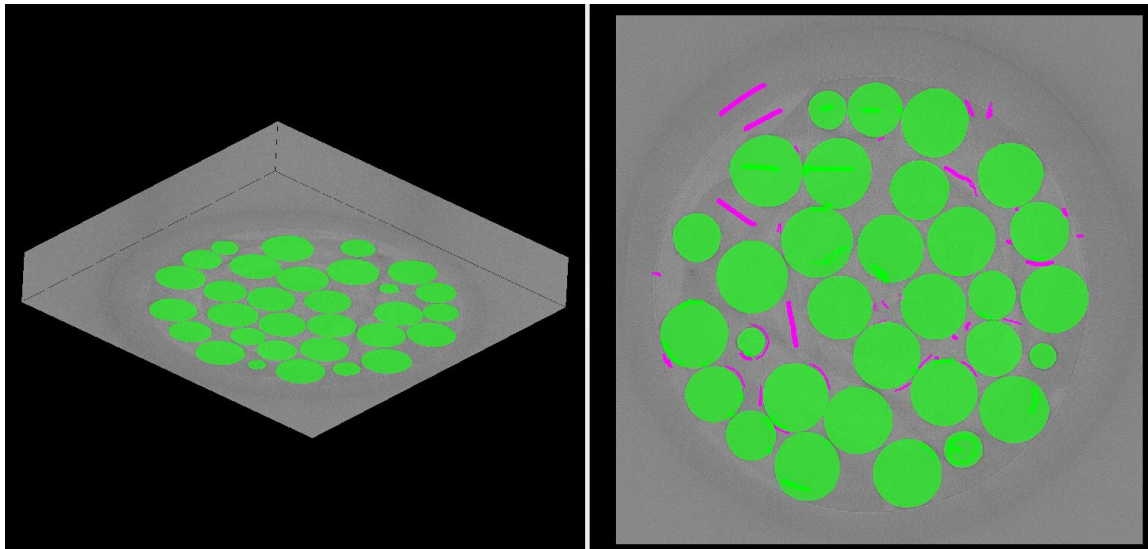


3. Select **Apply**.

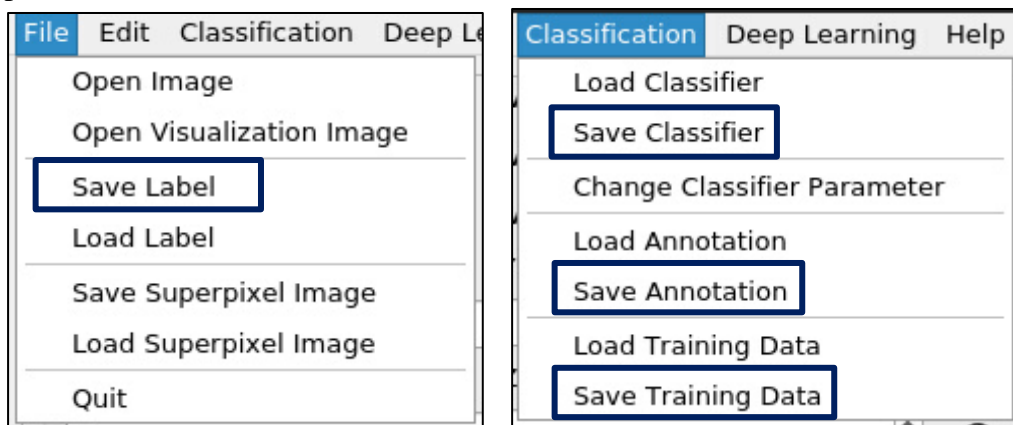
4. Alter the filters in Feature Extraction and Multi-scale Filtering Windows in order to see if the segmentation quality improves.



5. Keep classifying until you get a good segmentation.



6. Click on **File > Save label** to save the result of your segmentation. To save your current classifier, annotation and training data, go to **Menu>Classification>Save Classifier/Save Annotation/Save training data**. At the same place you can load your previous classifier and annotation.



In order to save, here are some definitions:

Label

- Label is the output of your segmentation.
- You can edit later the label if you load:
 - ***Just the label:*** In this case, the annotation starts from zero. So the quality of annotation might not be great.
 - ***Label + Annotation:*** With this, you can modify you label in a better way. When you train, the annotation protects what you already segmented.
- ***Note:*** With these options, you are just correcting the generated label, there isn't a classifier involved on the process.

Annotation

- It is the marked on pixels.
- It is only useful when it is related to the original image, where the annotation was made.
- It can be used for:
 - Changes on the classifier – Because it is just the markers, there is no feature applied, therefore, you can set the new features (new classification).

Classifier

- It predicts the class of given data points.
- You can save it for later use on other images, but the parameters won't be changed.
- Apply the classifier in other images only if you are sure that it is the final version.

Training data

- It is actual data set used to train the classifier.
- It has the features extracted for the markers, therefore, it is the filter result applied to image.
- If you load the training data and apply more training (new annotation), you can update your classifier.
- This can be used for any image.

Therefore, if you load the Training Data or the Annotation, there is no need to load the Classifier.

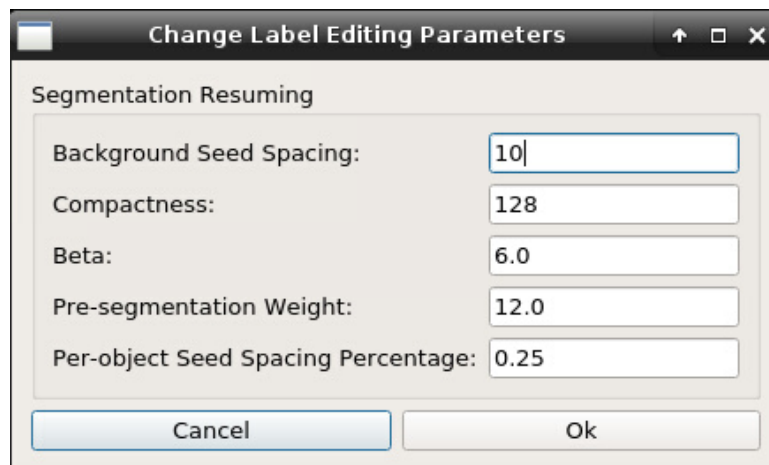
VI. II Loading and editing a previous label

Load Label and Annotation

Click on File>Open Label. Select Yes for Load Label Editing module.



Then, go to **Classification>Change Classifier Parameters.**



Change Label Editing Parameters:

Background Seed Spacing: Sets the background superpixels size. The higher the value, the larger the superpixels. which would avoid for example background selection as part of a cell that I marked with sequential labeling, extend labels or merge labels.

Compactness: Increasing the number will make the superpixels more regular and will make them lose edge information.

Beta: Defines the extension of generated label in the marked region. If you increase the parameter, the segmentation will cover a larger area than the annotation. If it decreases, the generated label will have an area closer to the one annotated by the user.

Pre-Segmentation Weight: Determines how much the current segmentation matches the original. Increasing it will result in fewer segmentation deviations from the previous one.

Pre-object Seed Spacing Percentage: Indicates the grid of segmentation to spread the seeds that the program uses to classify them as labels. E.g. If it is at 0.25, it means that the grid is dividing that region by 4.

Obs: After modifying your previous label, don't forget to save it and the annotation.

Annotation options:

After loading a label and annotation, Sequential labeling and Merge will appear as an option (See section IV).

<input type="checkbox"/> Erase	<input type="checkbox"/> Extend
<input type="checkbox"/> Sequential Labeling	<input type="checkbox"/> Merge <input type="checkbox"/> Split

VII. Keybord Shortcuts

Command	Execution
L	Show labels
S	Show superpixels
K	Show markers
Click Material + Ctrl	View a specific material
Click Material + Ctrl + Shift	Bring the axes to the geometric center of the material
Ctrl+z	Undo previous action (you must click apply to count the withdrawal on the label)
Shift	Keep it selected so you can drag the left image with your mouse (translation).

VIII. Deep Learning Menu

VIII. I What is Deep Learning

A supervised classifier includes a set of machine learning strategies that aim to learn, given examples of data separated in groups, a function F that better separates new data, according to the given examples. In Deep Learning, these groups are learned with the help of neural networks, which aim to learn a function that maps the original data to such groups. In the image segmentation domain, we try to map each voxel of the image to a label that identifies it. In other words, deep learning on Annotat3D is a type of machine learning that trains a classifier to perform an image classification in order to obtain a label image.

In the figure bellow, you can see an example of input data (CT image) and the label generated after the deep learning.

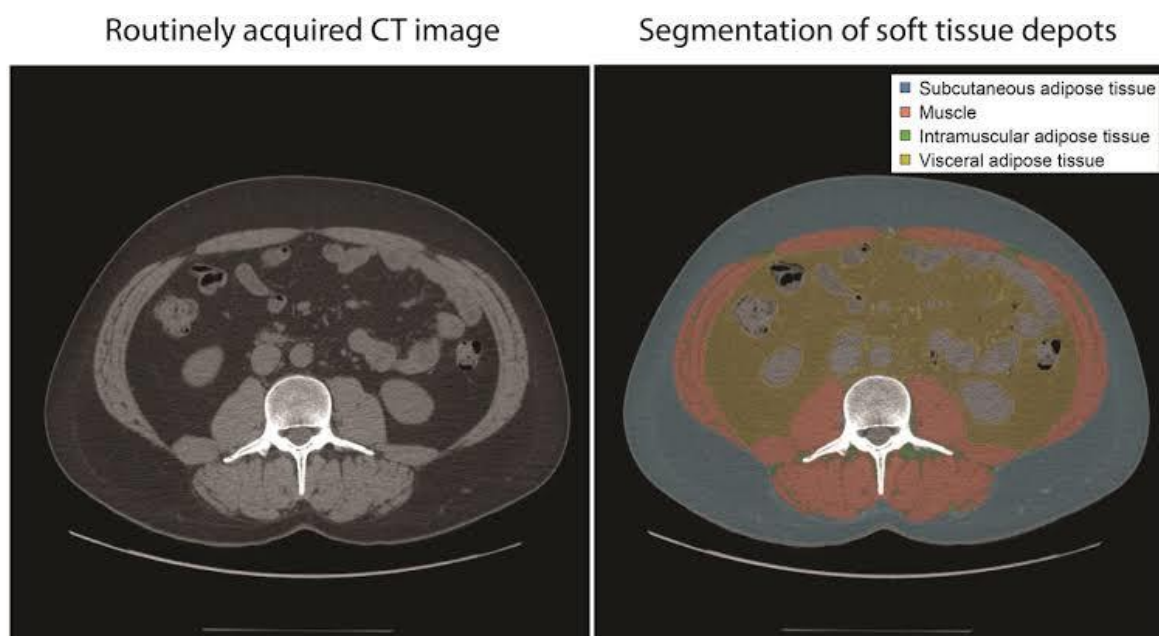
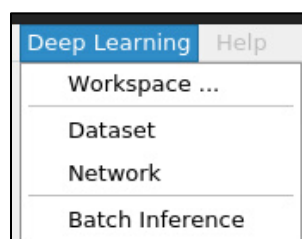


Image example taken from paper "Reliability and validity of the new VikingSlice software for computed tomography body composition analysis" (2019).

VIII. II What you need before using Deep Learning

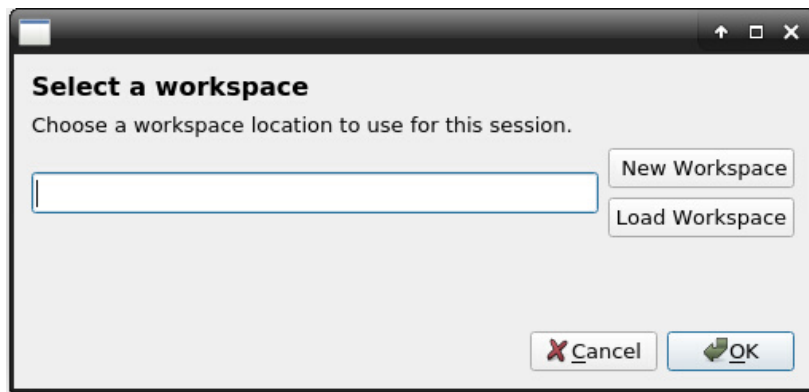
First, you'll need a set with original (grayscale) and segmented (label) images with the same dimensions. Those images don't need to have the same name, only to be in the same order in the list of images when creating a dataset. But we advise renaming all of them with initial name in order to facilitate the visualization, e. g. tomography1_100x100x100_8bit_gray.tif and tomography1_100x100x100_8bit_label.tif. This organized data will be used to train with predefined parameters, allowing the computer to learn by itself by recognizing patterns in various processing layers.

This is the Deep Learning Menu:



VIII. III Workspace Submenu

In the Workspace Submenu you can set the Workspace location. Everything will be saved in this folder, so it's better to create this folder inside the storage folder (i.e., DDN).

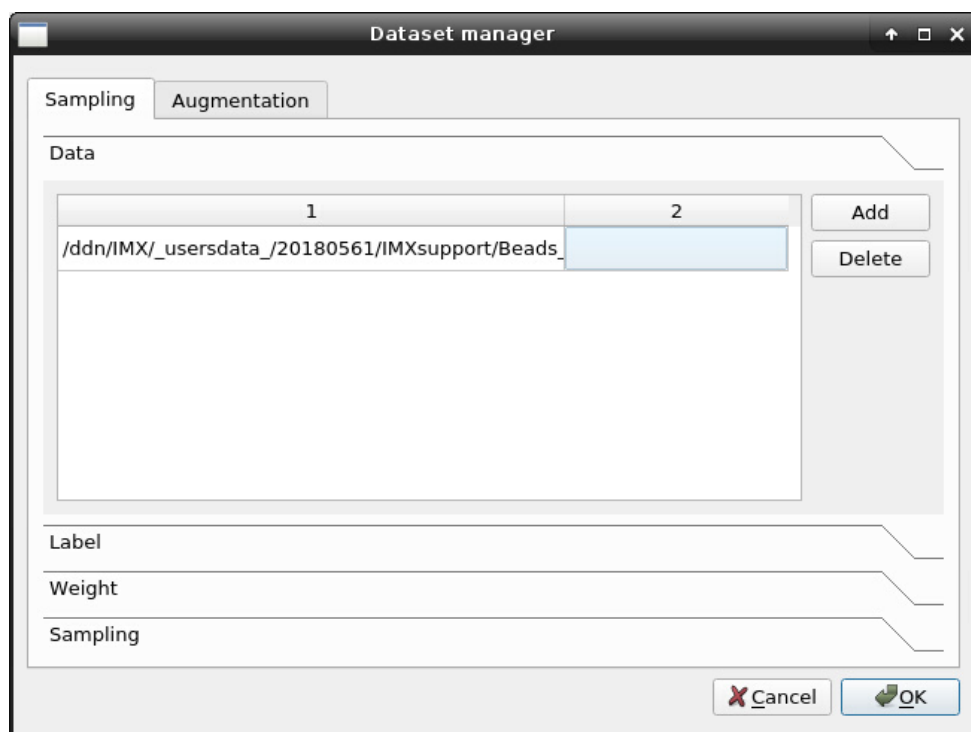


VIII. IV Dataset Submenu

VIII. IV. I Sampling

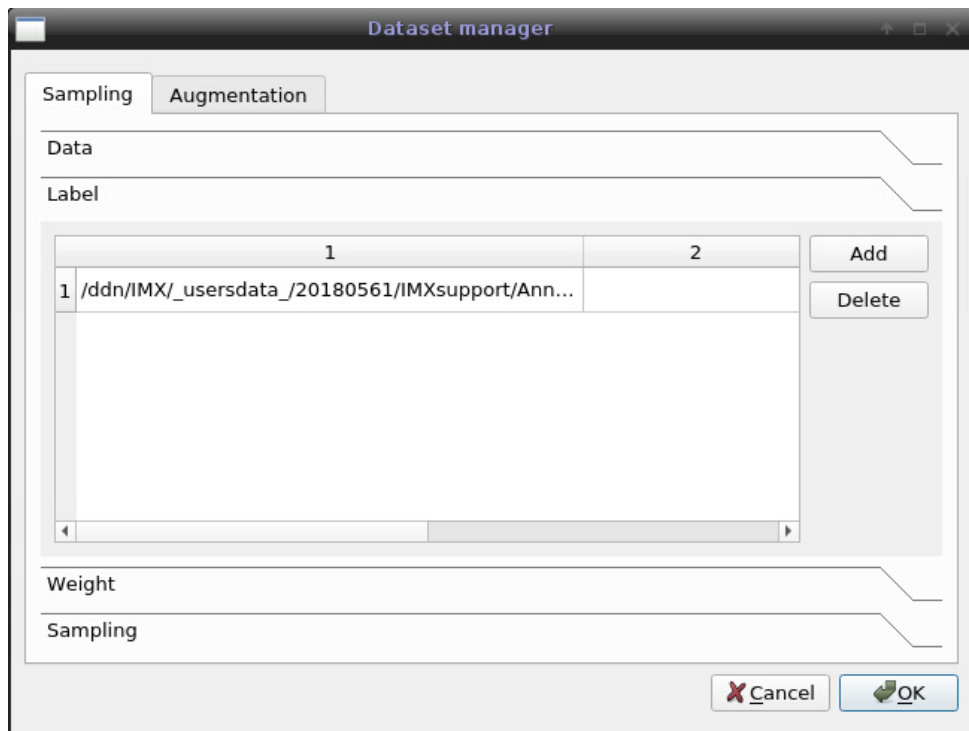
Data: You can add files in tiff or raw extensions. This is the grayscale image.

Note: In case of raw images, it is important to point that, in order to infer the image properties, the file must contain necessary information (size and bitdepth), for example: `this_is_my_file_200x200x100_16bit.raw`



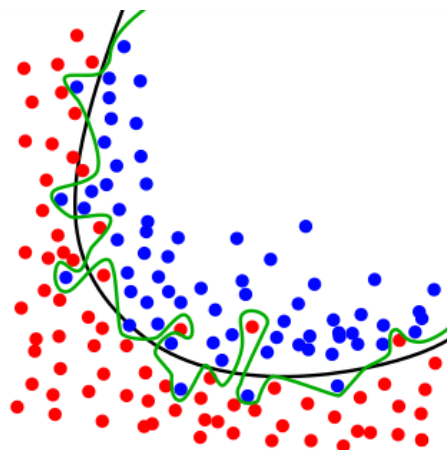
Label: Keep in mind that the label must have enough visual field in order to understand the shape and texture of your object.

Tip: Depending on the complexity of your sample, it might be interesting to create a classifier to separate background and sample and another one to identify differences inside your sample.



Attention: Overfitting is a problem that can be originated from a limited dataset, in which there is a small amount of data, in which there is a small amount of data with good annotation. Because there isn't a lot of data for training, the network will be biased by the previous information. In another words, it becomes overly accurate for the input data (low error), but it is not capable of achieving a proper result if you apply it to another image. The major issue is that it learns a lot about the outliers in your data, being unable to segment another image with high quality.

Example: Segmentation of a curve, where the black dots are the input image (different from the original). In blue, the result of overfitting (considers every outlier), in blue, the result of good segmentation. If you apply the network into your original data and it gets a good result, but it is a disaster on another images, it might be a sign of overfitting.

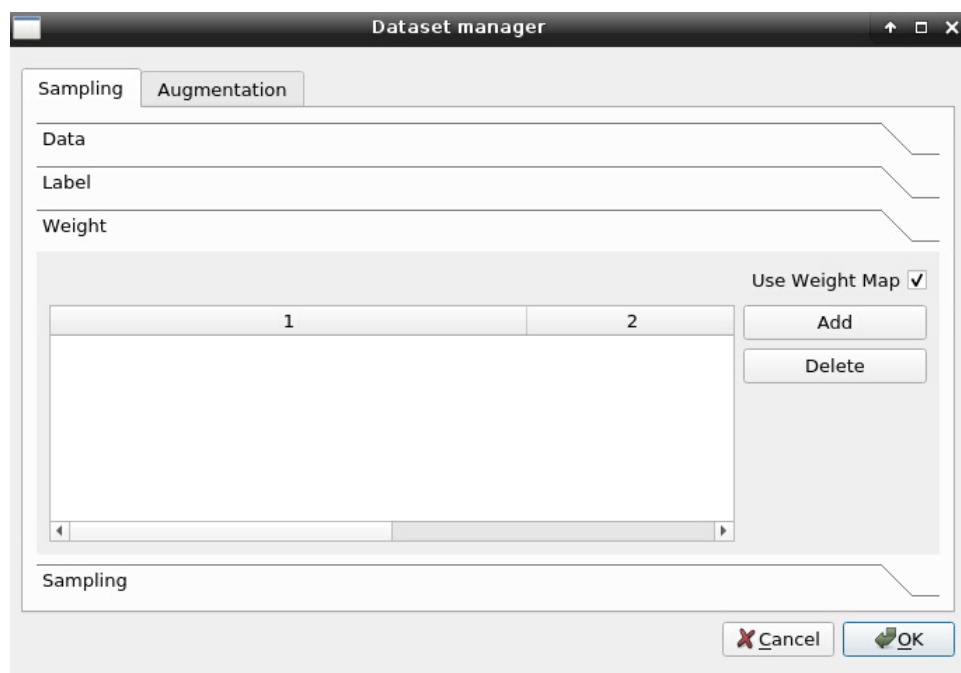


This is a diagram showing overfitting of a classifier. While the black line fits the data well, the green line is overfit¹.

¹Image extracted from: <<https://commons.wikimedia.org/wiki/File:Overfitting.svg>> Access: 30/10/19.

Weight map: It is used when you want to make the classifier understand that it must focus on specific areas. The weight map is an image whose voxel intensities make the network training penalize the classification error proportionally to the value of the voxel. Hence, higher values will force the network to improve classification for certain voxels. When certain materials/objects are small, for instance, it might be interesting to provide a weight image with higher values for those objects. There is no limit to the values in the weight map, although assigning weights in the range of $[0, 255]$ may force the network to completely ignore regions (i.e., voxel values of 0) or penalize errors for some voxels up to 255 times more. If you do not provide a weight image, the network training will assume that the image has uniform weight equal to 1.0.

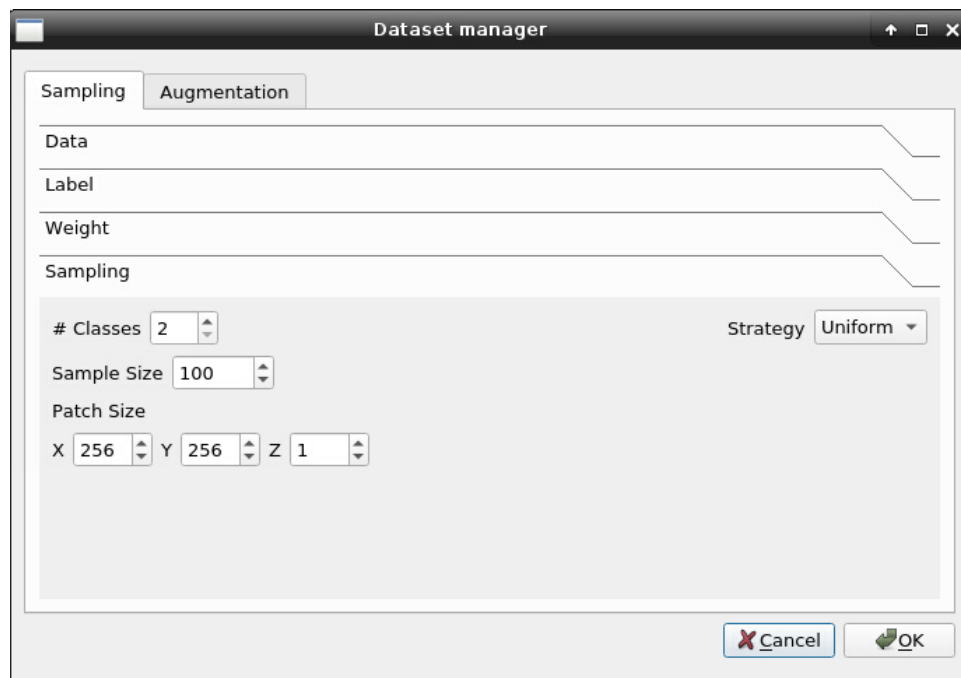
Note: The weight values must be positive.



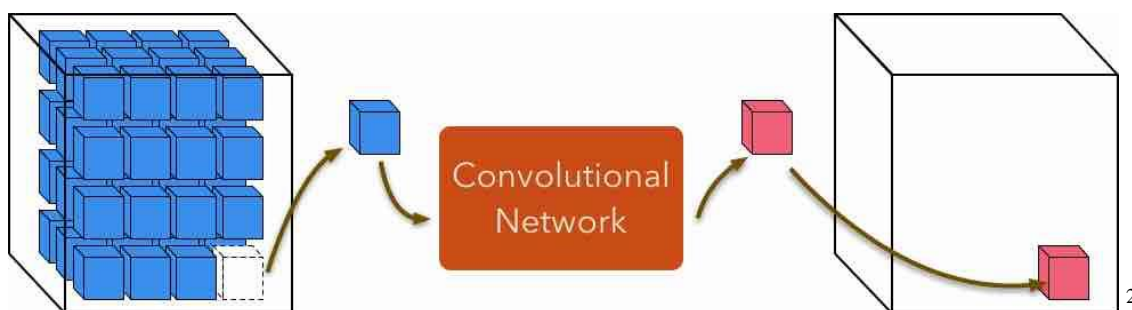
Example 1: There is an image with 5% of the volume of the sample corresponding to a label A. The rest of the image is distributed in labels B, 40%, C, 35% and D, 20%. Because of the low probability of a certain are to be correspondent to label A with uniform weight, there is a higher chance for the training to mis classify it as B, C or D. That's why in this case it would be useful to use a mask where the A regions have a higher weight. This situation is known as Class unbalance (when classes have too low percentages).

Example 2: If you have two very similar patterns for two different labels, it might be useful to attribute different weights for each object.

Sampling: You must define:

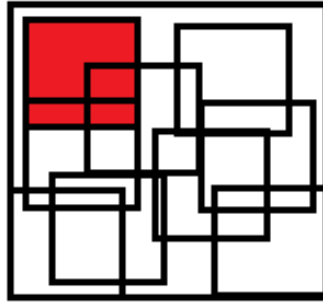


- **Class** – The number of label classes.
- **Strategy** – Uniform. It means equal probability of extraction/selection of patches for training. It randomly extracts patches over an uniform distribution, but that **doesn't mean it's a perfect grid**.
- **Sample size** – The network always works with a small crop of the image henceforth denoted patch (pink cube), so this parameter sets the number of patches on the input image (blue cubes). The more you cover the image (higher sample size), the slower the training, but possibly results are more accurate. Patches may overlap.

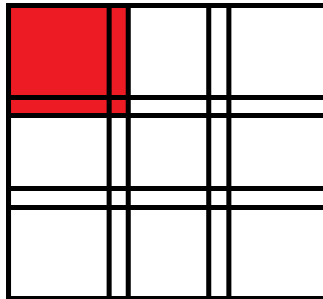


Example of an uneven grid: This is just an example to clarify what a sample and patch size are. In this image, the sample size is 9 with random and uniform distribution. In this situation there is an overlap and each subimage (in red) is a Patch. Every square has the same probability of being chosen for the training.

² Image extracted from: <https://niftynet.readthedocs.io/en/dev/window_sizes.html> Access: 30/10/19.



Example of perfect grid: Sample size: 9; patch is the red square. This isn't how the patch distribution occur.



- **Patch size** – The training uses all of the patches, but at a random order. It defines how the network shall go through the input data. The size should be bigger than the smallest object that you want to observe. In another words, you are setting the field of view for the training.
 - *Note:* The input at V-net (network) has a patch size of 64x64x64. And the input at U-net 2D is 224x224.

After this, click on Augmentation.

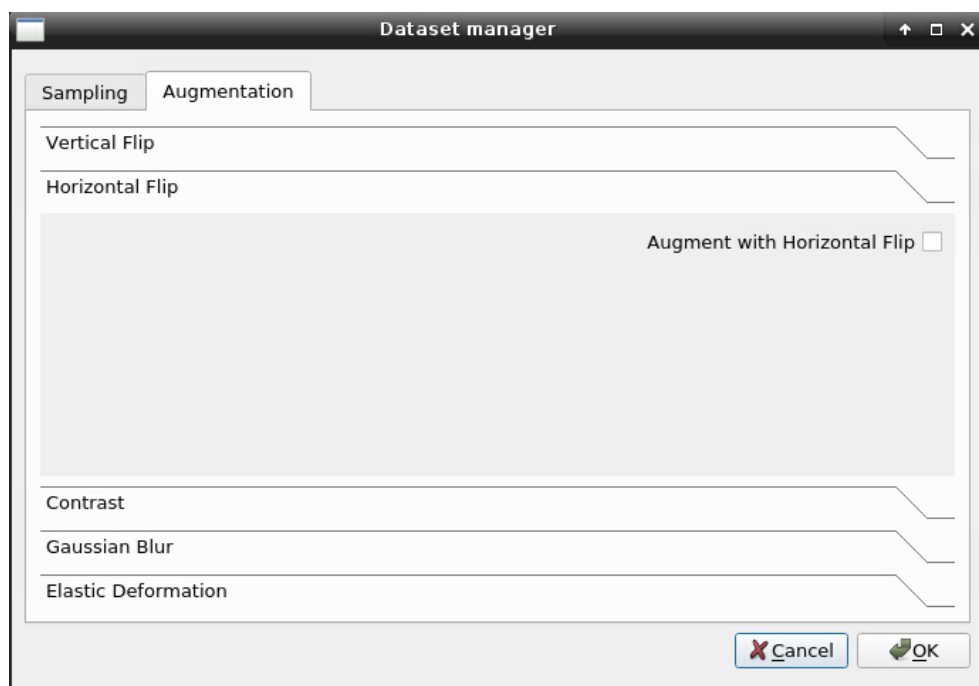
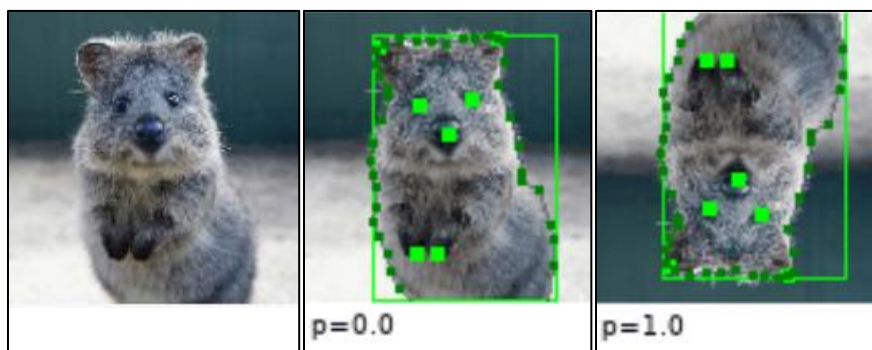
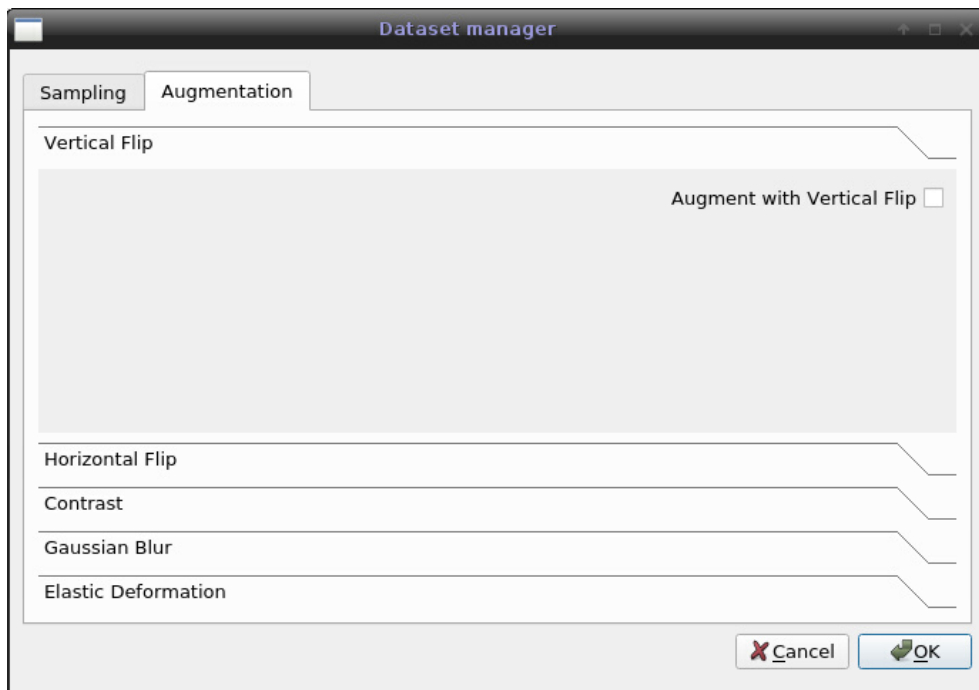
VIII. IV. II Augmentation

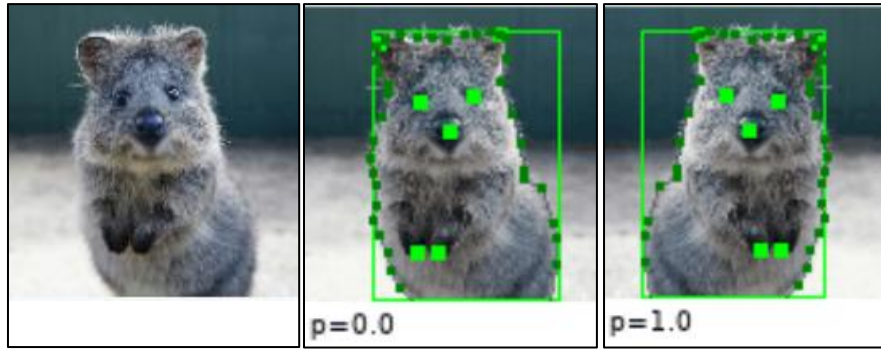
Data augmentation strategies are able to generate artificial data by applying small deformations on the original data. The main purpose is to generate slightly different data, increasing the number of labeled data when scarce, and creating more challenging problems for the classifier, leading to a more generalized network. The following parameters are directly related to shape and are applied into the original image.

Example images below were extracted from <<https://github.com/aleju/imgaug>>.

Vertical and Horizontal flip: Enabling it allows the network to understand that different patterns of rotation exists.

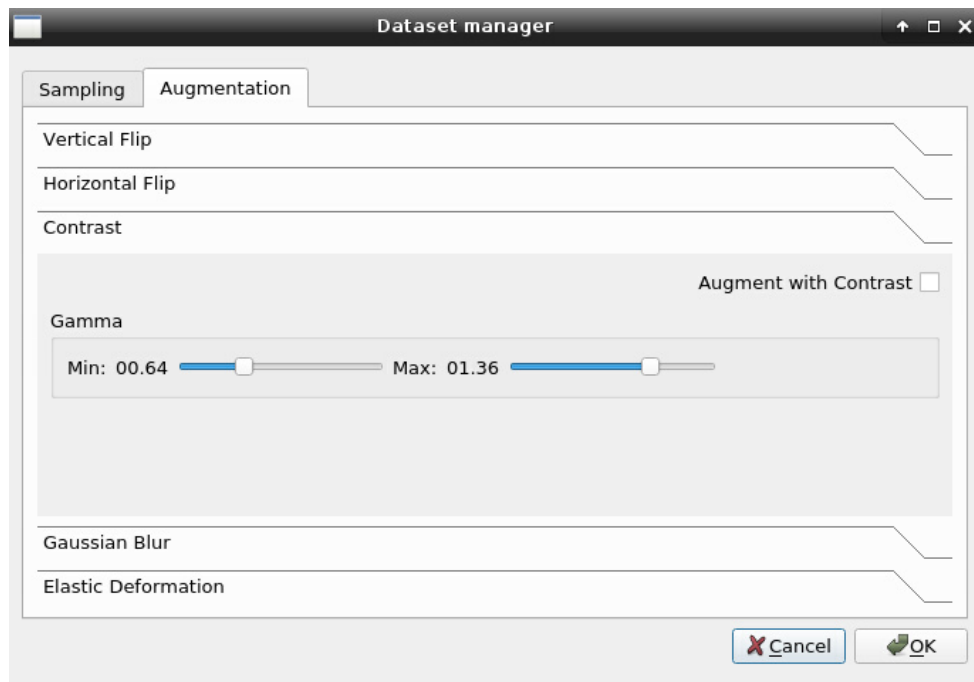
Example: You want to train in order to segment dog's profiles. You insert a dataset with dog's profiles to the left, but you know for certain that they could also be turned to the right, but not upside down. In this case you would habilitate the horizontal flip, but not the vertical one.



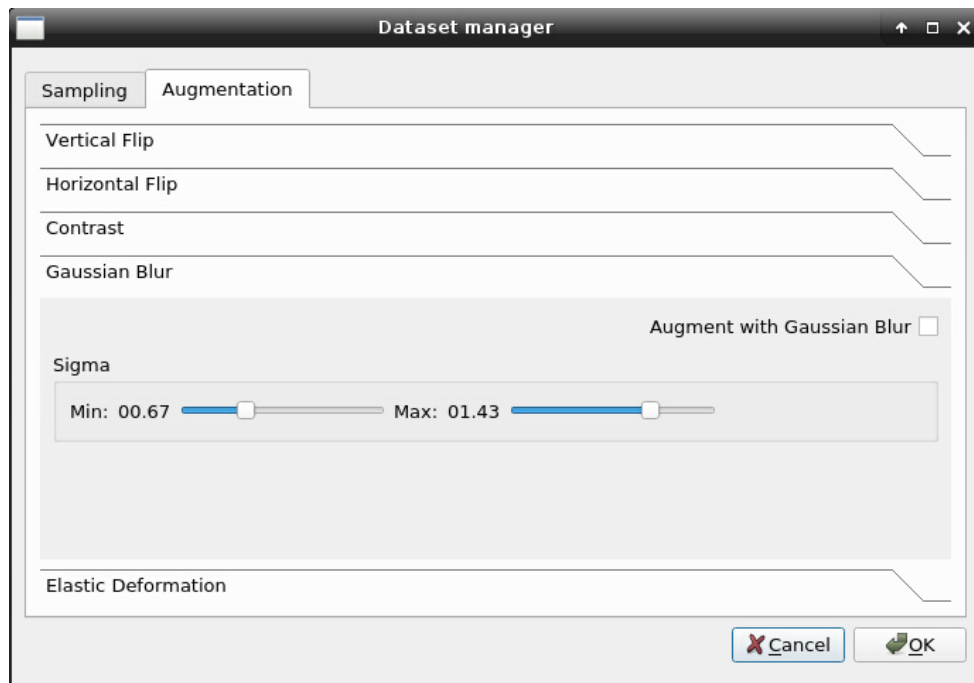


Contrast: If you need to segment images with different contrast patterns in relation to the dataset, is better to enable this option. It will make the network to learn not only one pattern of contrast, but a whole new range of possible contrast situations.

Example: You put a dataset with a girl with a specific color, but in your segmented images this girl appears in different colors and contrast. You want the network to correctly identify the girl in the images regardless of the contrast changes.

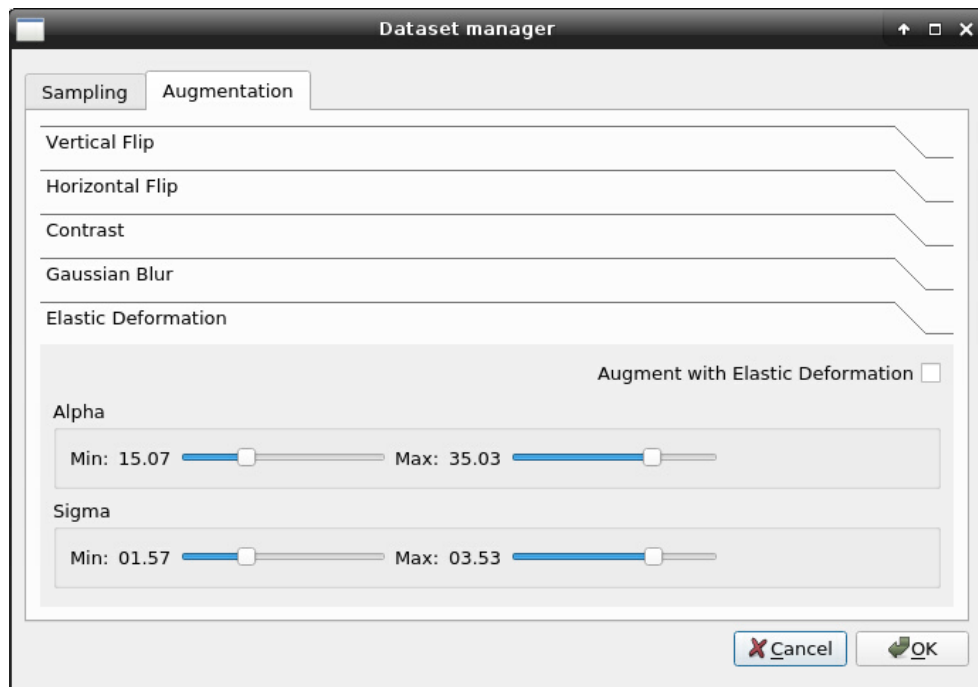


Gaussian Blur: The higher the σ value, the higher the smoothing. In this parameter, you basic is destroying the texture in order to make a better classification of shape.

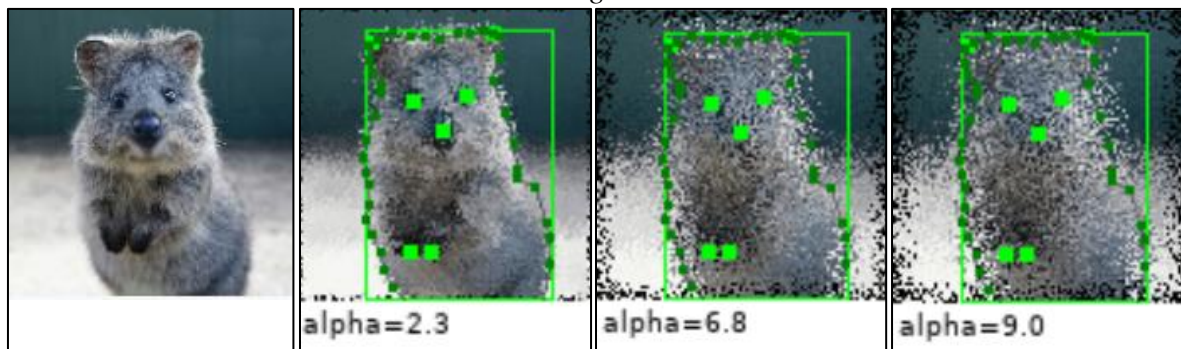


Elastic transformation: It determines if in your training it will accept variations in shape (distortion) and texture.

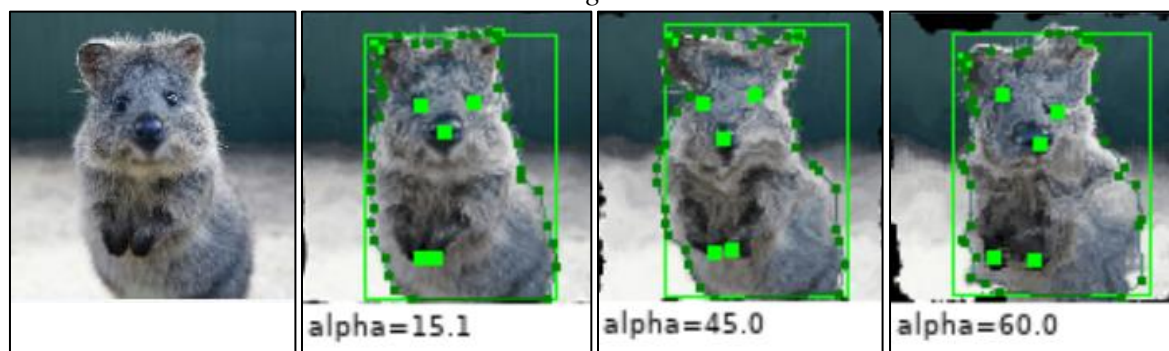
- Alpha stands for the deformation intensity.
- Sigma stands for the quantity of random noise over deformation to change the pixel intensity value (randomness over deformation).



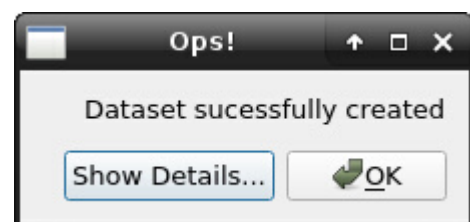
With Sigma 0.2



With Sigma 0.5



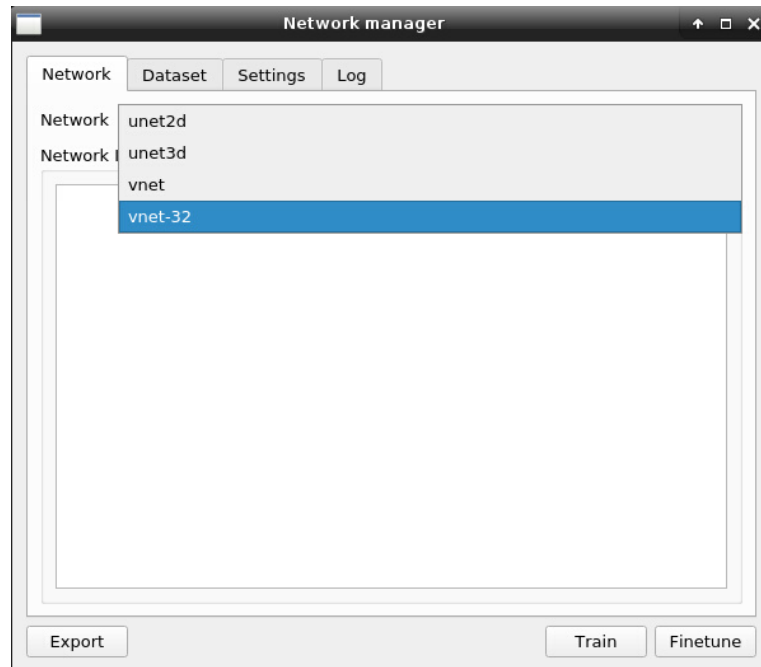
Once everything is set, click “OK” and the following popup shall appear. Click on ok and then close the dataset window.



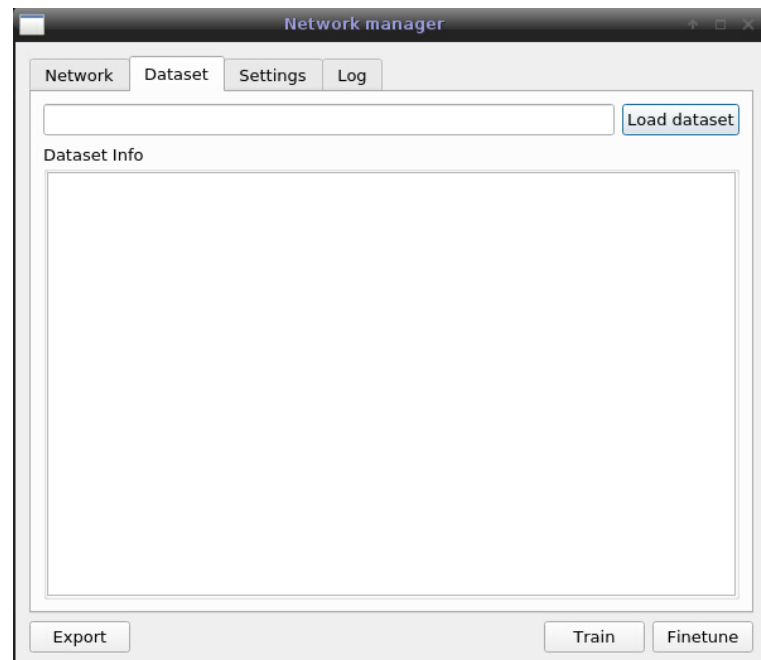
VIII. V Network Submenu

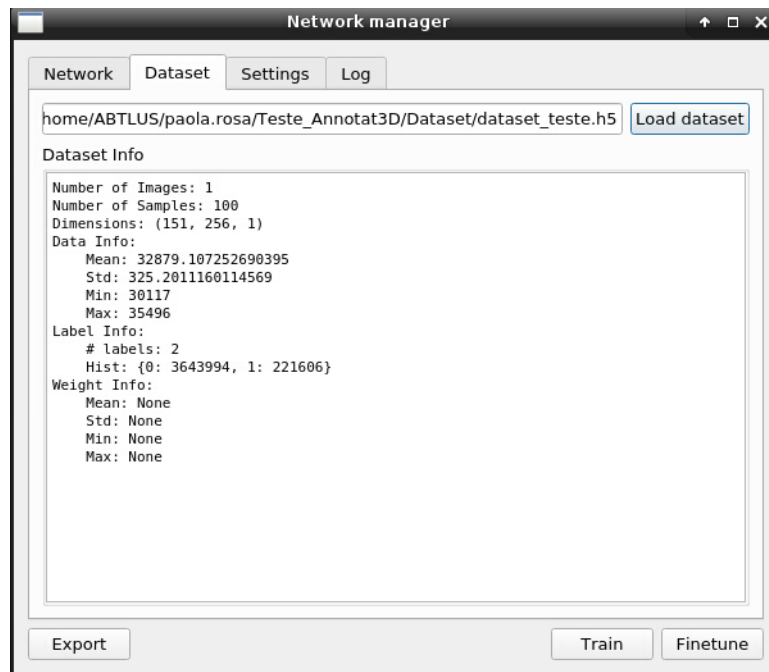
Network: It sets the type of neural architecture of the network that will be used for training. The options are:

- ***U-net – 2D and 3D:*** It is the best option for 2D training and works faster for 3D training.
- ***V-net – 3D:*** It usually gives better results, but it demands more computationally.

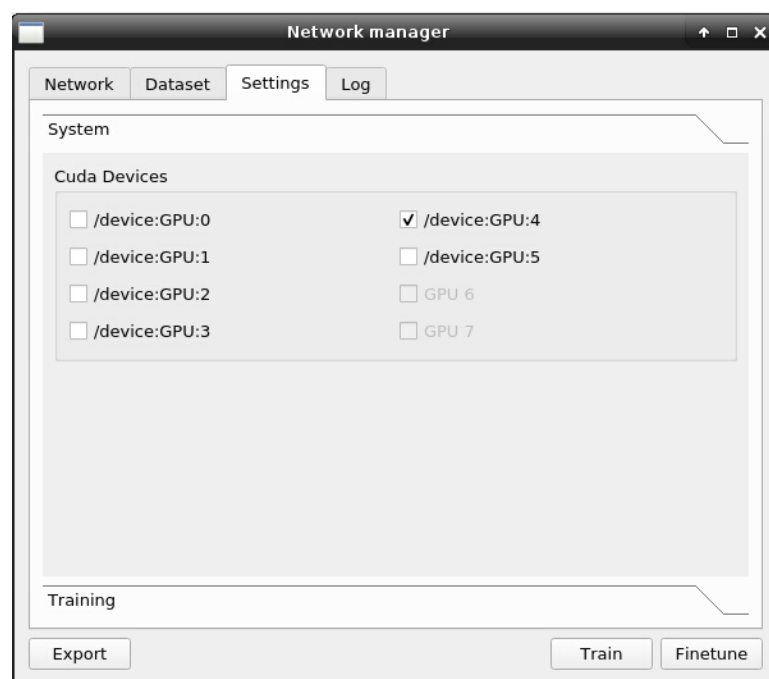


Dataset: Load the dataset created on the previous step (see section VIII. II).





Settings: It determines the GPUs that will be used.



In order to use the correct GPUs available, check it out using the Terminal. Type:
watch -n 0.1 nvidia-smi

You will see the following information:

```
Every 0.1s: nvidia-smi
```

Tue Oct 22 15:24:28 2019

NVIDIA-SMI 418.39 Driver Version: 418.39 CUDA Version: 10.1									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M			
0	Tesla K80	On	00000000:0E:00.0	Off			0		
N/A	34C	P0	59W / 149W	344MiB / 11441MiB	0%		Default		
1	Tesla K80	On	00000000:0F:00.0	Off			0		
N/A	60C	P0	148W / 149W	10942MiB / 11441MiB	100%		Default		
2	Tesla K80	On	00000000:87:00.0	Off			0		
N/A	61C	P0	142W / 149W	10976MiB / 11441MiB	100%		Default		
3	Tesla K80	On	00000000:88:00.0	Off			0		
N/A	39C	P0	76W / 149W	416MiB / 11441MiB	0%		Default		
4	Tesla K80	On	00000000:8D:00.0	Off			0		
N/A	31C	P0	60W / 149W	291MiB / 11441MiB	0%		Default		
5	Tesla K80	On	00000000:8E:00.0	Off			0		
N/A	59C	P0	152W / 149W	6715MiB / 11441MiB	100%		Default		

Processes:					GPU Memory Usage
GPU	PID	Type	Process name		
0	3688	G	/usr/lib/xorg/Xorg		23MiB
0	4272	C	python3		56MiB
0	23065	C+G	python3		132MiB
0	25863	C	python3		56MiB

Check for the GPUs with low memory usage (red rectangle).

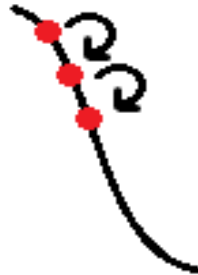
Training: In this part, you will set parameters that will directly affect the GPU memory usage and the quality and velocity of training.

The screenshot shows a window titled "Network manager" with tabs for "Network", "Dataset", "Settings", and "Log". The "Settings" tab is active, showing a "Training" section. In this section, there are three input fields: "Batch Size" set to 4, "Iterations" set to 1000, and "Learning Rate" set to 0.0000100000. A note next to the Batch Size field says "(This affects the GPU memory usage)". Below these fields is a "Loss" dropdown menu with three options: "Cross Entropy", "Dice" (which is currently selected and highlighted in blue), and "Cross Entropy + Dice". At the bottom of the window, there are three buttons: "Export", "Train", and "Finetune".

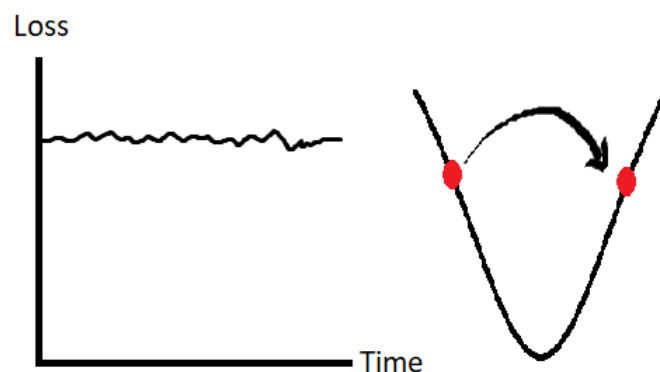
You must imagine that basically the training is trying to determine a function that can predict labels based on a dataset. Ideally, this function must obtain the lowest possible error, also known as the optimum condition (global minimum). Optimization is carried out by a variation of the gradient descent method in order to find a minimum point of the error function.



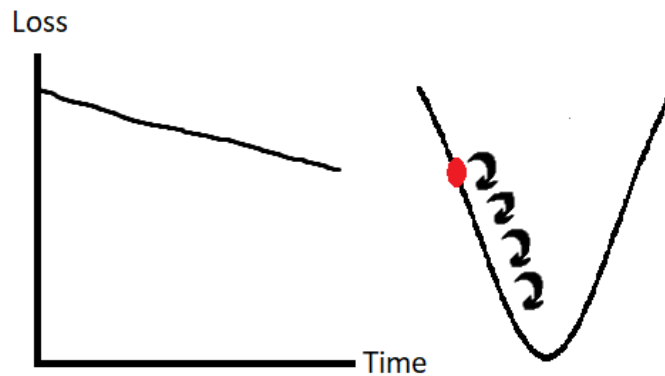
- **Batch size:** defines how many image patches will be considered at the same time during network training. It is related to GPU performance.
 - The higher the Batch Size, the higher the smoothness of the error function since more image patches are considered during optimization, thereby reducing the likelihood of optimization falling into a local minimum.
 - When you have a lot of incorrect annotation, it might be better to increase the batch size.
- **Iterations:** This parameter determines how many times the network will test. Each new interaction means an update in you training, giving new results (a new matrix is updated by combining values because of the update in the error function).
 - *Tip:* Do a first training with few iterations (1000) just to see if the other parameters seem to make sense. Then, increase the iterations in order to give better results.
- **Learning rate:** It is a gradient descent parameter considered because the shape of the error function and its global minima are not known. Hence, it is necessary to determine a suitable learning rate to find the best approximation for it (i.e., trying to find a satisfactory local minimum that may or may not coincide with a global minimum). The \uparrow higher the learning rate, \uparrow the bigger the step taken at each point, \uparrow therefore, it is a faster training.



- If you end your training with a relatively constant error, e.g. it's taking a very long time to decrease the value, you probably should increase the learning rate (increase the step between points).



- On the other hand, when your loss is constant or varies a lot, it is probably better to decrease the learning rate.



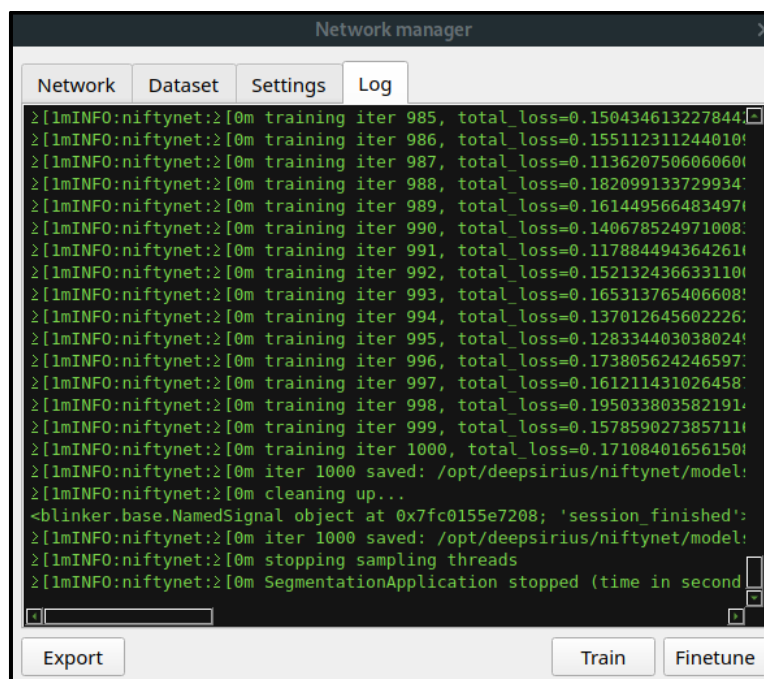
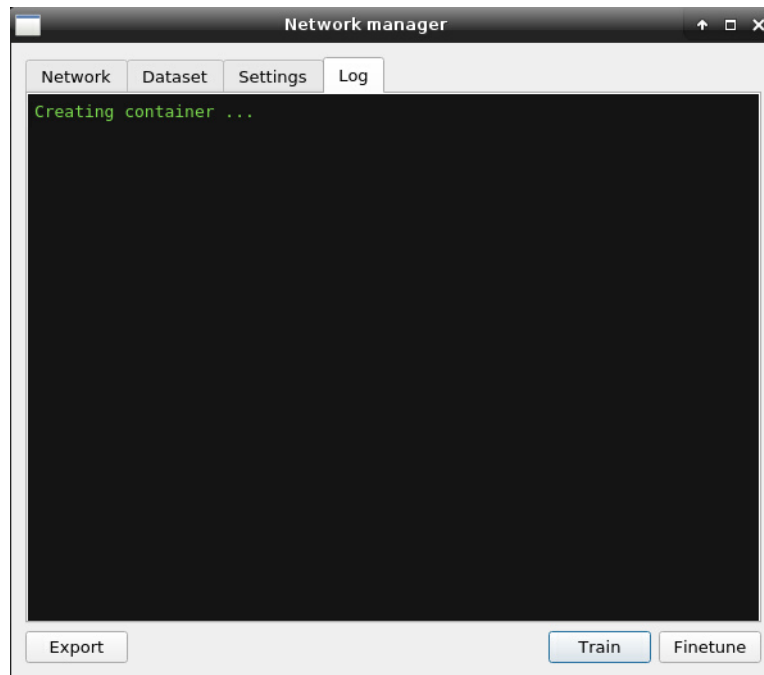
- The problem about very small steps might be that it will be trapped into a local minimum that is not the lowest point, because it stops on the first that it finds.



• **Loss:** The loss function is responsible for comparing the output provided by the network during training with your ground truth. It essentially measures the error that is obtained during optimization to model the classification function. Different loss functions may be more suitable to different data:

- Cross entropy – It compares the dataset and the other images on a voxel to voxel basis. It gives more importance to texture.
- Dice – It overlaps the dataset and the other images. It is a summarized comparison. It gives more importance to shape.
- Cross entropy + Dice – Takes both in consideration.

Once you are done with the settings, start the training by clicking in Train (you are starting a training from zero).

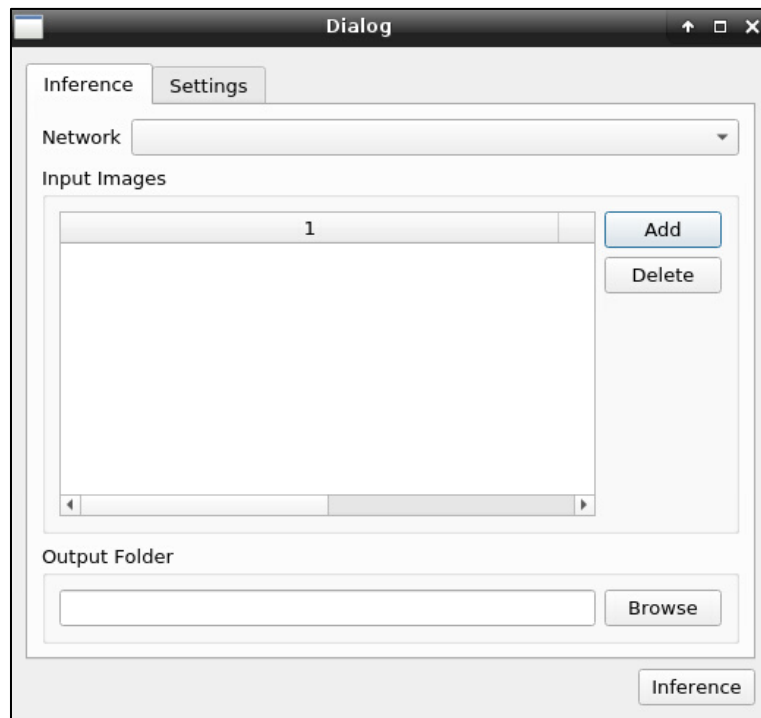


Export if you are satisfied with the error result, it works as a checkpoint because you can retrain it. If you aren't satisfied, change the parameters and then click on Finetune. If you trained one network and then clicked on Train, it will start all over again.

VIII. VI Batch Inference Submenu

Inference: With your network set, you may now present a new image for classification.

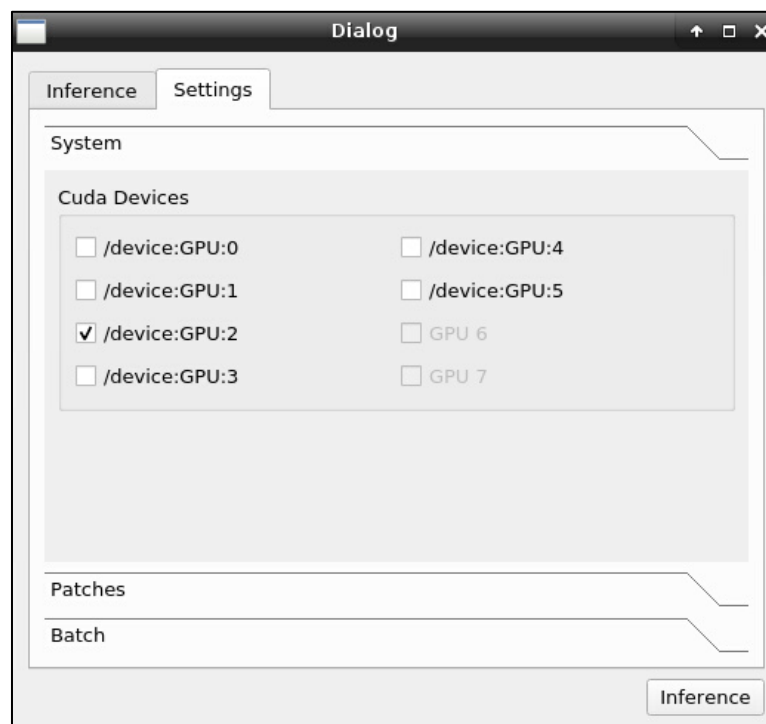
First, select the network created (see section VIII. III). Then, add the Input images. And select an Output folder.



Click on Settings to continue.

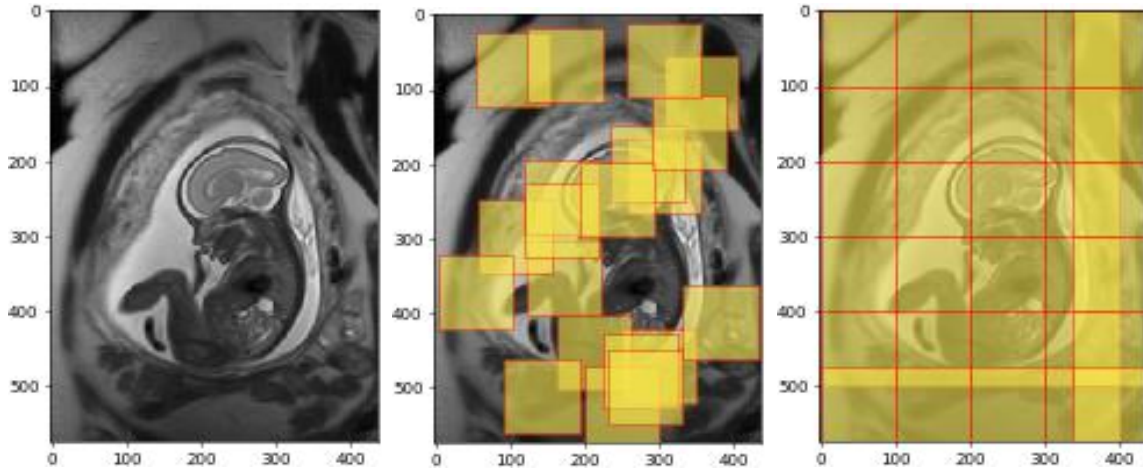
Settings:

- **System:** It determines the GPUs that will be used.



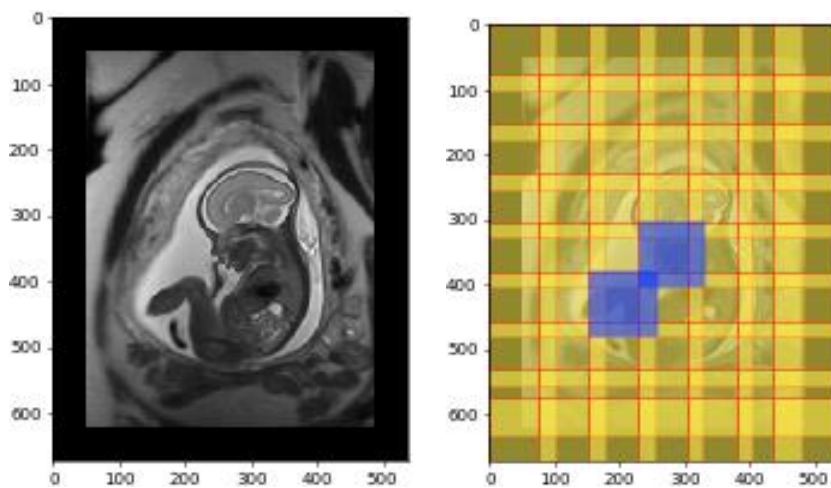
Borders: Volume Padding and Patch Border

Here we have an example of an input image³ (left). As described at **Sample Size** topic, the image has a uniform distribution of patches (middle and right). The following parameters are used to set the input image in order to prevent the presence of artifacts at the final image.



Volume Padding – Adds a 0-valued frame around the input image to ensure that boundary effects to classification being done patchwise be mitigated.

Patch Border – Similar to Volume Padding, this parameter controls the amount of overlap between patches sampled over the target image for inference. Lower values increase classification speed, at the cost of edge artifacts. In another words, determines how much edge will be thrown away when making the inference. We throw it away because of an edge effect on each patch inside the image.



Left: Volume Padding example. **Right:** Patch Border example.

³ Image extracted from: <https://niftynet.readthedocs.io/en/dev/window_sizes.html> Access: 30/10/19.



Batch:

