

N2V Lab - Exercises

Learning Objectives

- Train N2V networks using the example notebooks.
 - Denoise images.
 - Understand how to use N2V.
 - Learn how to adapt the examples to use your own data.
 - Bonus: Denoise data that is degraded by structured noise.
 - Bonus: use a trained network from within Fiji
-

Preamble: Be sure you have followed the installation instructions in the separate document '**N2V Lab - Environment Setup**'. If you did so, you should be good to go! However, we do recommend that you also first do our '**CARE Lab - Exercises**', which do mention many of the problems you might run into.

Exercise #0: Start Jupyter

Open a terminal and navigate to the location where you cloned your N2V repository.

Please start the Jupyter server via:

\$ jupyter notebook

This should open a browser and show you all files contained in the directory.

Exercise #1: Train your first N2V network

The GIT repo you have previously cloned contains various example notebooks for different types of 2D and 3D data:



The screenshot shows a file browser interface with tabs for 'Files', 'Running', and 'Clusters'. Below the tabs, there is a prompt 'Select items to perform actions on them.' and buttons for 'Upload', 'New', and a refresh icon. The main area displays a directory tree for 'n2v / examples / 2D'. The directory contains several subfolders: '..', 'denoising2D_BSD68', 'denoising2D_RGB', 'denoising2D_SEM', 'structN2V_2D_convallaria', and 'structN2V_2D_synth_mem'. Each folder has a checkbox and a 'Last Modified' timestamp.

| Name | Last Modified | File size |
|--------------------------|----------------|-----------|
| .. | seconds ago | |
| denoising2D_BSD68 | 2 hours ago | |
| denoising2D_RGB | 2 hours ago | |
| denoising2D_SEM | seconds ago | |
| structN2V_2D_convallaria | 35 minutes ago | |
| structN2V_2D_synth_mem | an hour ago | |

Most example folders contain two notebooks: one for training ('01_training.ipynb') the network and one to apply it ('02_prediction.ipynb') to actually denoise data:



The screenshot shows a file browser interface with tabs for 'Files', 'Running', and 'Clusters'. Below the tabs, there is a prompt 'Select items to perform actions on them.' and buttons for 'Upload', 'New', and a refresh icon. The main area displays the contents of the 'denoising2D_RGB' folder. It contains two files: '01_training.ipynb' and '02_prediction.ipynb'. Each file has a checkbox, a 'Last Modified' timestamp, and a 'File size'.

| Name | Last Modified | File size |
|---------------------|---------------|-----------|
| .. | seconds ago | |
| 01_training.ipynb | 2 hours ago | 5.68 MB |
| 02_prediction.ipynb | 2 hours ago | 2.75 MB |

Note that in some of the examples both are combined to a single notebook.

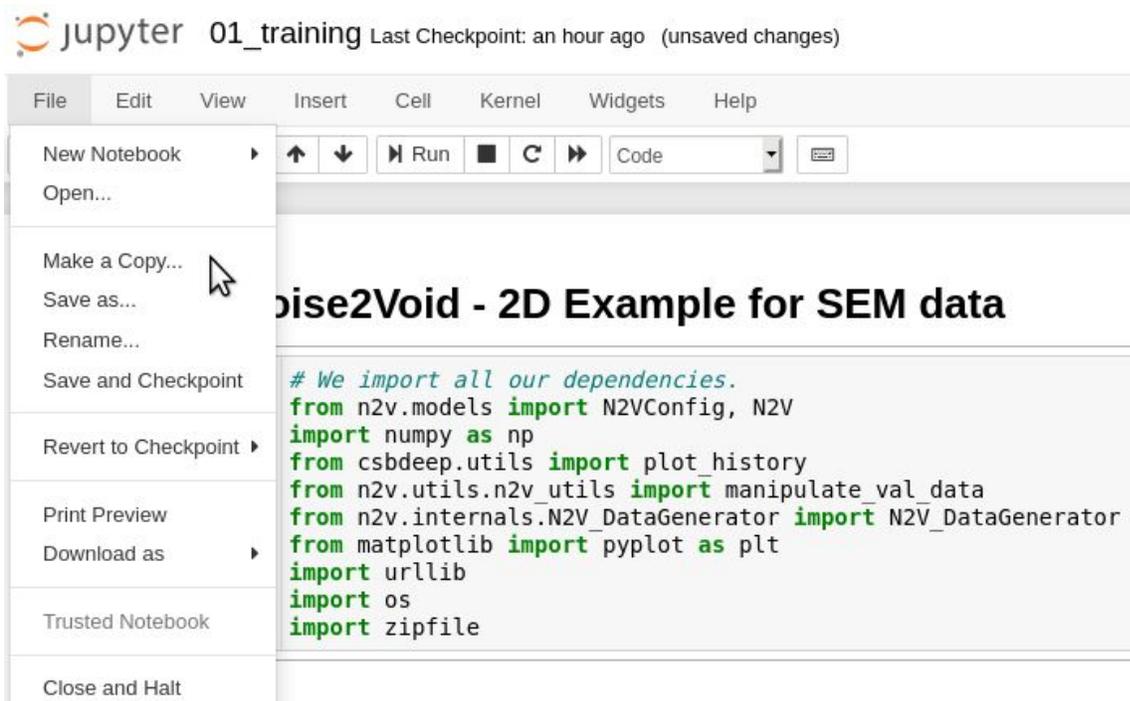
Choose any of the examples (except the 'structN2V' examples, which will be the subject in a separate exercise further down below) and execute each cell of the notebooks to train a network and then denoise some images.

Note: all notebooks will download the required example data for you.

Exercise #2: Train an N2V network on your own data and denoise it

The goal of this exercise is to teach you how to apply N2V to your own data. For Noise2Void, your data can simply be a collection of noisy images. You do not need any training pairs or clean ground truth images.

Depending on whether your example data is 2D or 3D, choose a corresponding example and make a copy of both notebooks.



The screenshot shows a Jupyter Notebook interface. At the top, the title bar reads "jupyter 01_training Last Checkpoint: an hour ago (unsaved changes)". Below the title bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The "File" menu is open, showing options: "New Notebook", "Open...", "Make a Copy...", "Save as...", "Rename...", "Save and Checkpoint", "Revert to Checkpoint", "Print Preview", "Download as", "Trusted Notebook", and "Close and Halt". A mouse cursor is hovering over "Make a Copy...". The notebook content area displays the title "Noise2Void - 2D Example for SEM data" and a code cell with the following Python code:

```
# We import all our dependencies.
from n2v.models import N2VConfig, N2V
import numpy as np
from csbdeep.utils import plot_history
from n2v.utils.n2v_utils import manipulate_val_data
from n2v.internals.N2V_DataGenerator import N2V_DataGenerator
from matplotlib import pyplot as plt
import urllib
import os
import zipfile
```

Try to adapt the notebook to use it with your own data.

If your data is in the same format as our example data, you mainly have to change the paths from which the data is read to point to your data:

```
In [ ]: 1 # We load all the '.tif' files from the 'data' directory.
        2 # If you want to load other types of files see the RGB example.
        3 # The function will return a list of images (numpy arrays).
        4 imgs = datagen.load_imgs_from_directory(directory = "PATH/TO/YOUR/DATA")
        5
        6 # Let's look at the shape of the images.
        7 print(imgs[0].shape)
        8 # The function automatically added two extra dimensions to the images:
        9 # One at the beginning, is used to hold a potential stack of images such as a movie.
       10 # One at the end, represents channels.
```

Noise2Void allows you to later denoise the same data that you used for training. Once you network is trained, adapt the prediction notebook to denoise your data.

Exercise #3 (bonus): Structured Noise2Void

While Noise2Void usually handles the shot noise and readout noise (common in low light data) quite well, some microscopes produce structured noise that cannot be removed by the standard Noise2Void.

Structured Noise2Void is a variation of Noise2Void that can mitigate this problem.

Execute the 'structN2V_2D_convallaria' example notebooks and have a look at the results. To achieve better results it can help to run the training for more epochs (e.g. 100).

Compare the results to what simple Noise2Void would give you. The easiest way to train a regular N2V network on the same data is to remove the 'structN2Vmask' parameter.

```
1 >ps_per_epoch is set to (number of training patches)/(batch size), like this each training patch
2 once per epoch.
3 VConfig(X, unet_kern_size=3,
4     train_steps_per_epoch=int(X.shape[0]/128), train_epochs=10, train_loss='mse', batch_norm=True,
5     train_batch_size=128, n2v_perc_pix=0.198, n2v_patch_shape=(64, 64),
6     n2v_manipulator='uniform_withCP', n2v_neighborhood_radius=5, structN2Vmask = [[0,0,1,1,1,1,1,1,1,0]])
7
8 >k at the parameters stored in the config-object.
9 )
```

For more details on the method checkout the paper:

<https://colemanbroad.github.io/colemanb-net/Removing%20Structured%20Noise%20With%20Self-supervised%20Blind-spot%20Networks.pdf>

Exercise #4 (bonus): Use a trained network from within Fiji

You can use any `TF_SavedModel.zip` you created at the very bottom of any training notebook.

With a trained and saved network at hand, please open Fiji and follow these sequence of steps:

1. Help > Update...
2. Manage update sites
3. Add update site
4. Give the new site a name, e.g. "CSBDeep", and point it to "<https://sites.imagej.net/CSBDeep/>"
5. Restart Fiji
6. Open the image that you want to denoise.
7. Click Import > bioimage.io.zip and choose the network ZIP-file you have exported from any of the Noise2Void notebooks.
8. Click 'Predict' and choose the image to be denoised and segmented. Also choose the Axes of prediction input (for 2D images this is likely to be 'YX').
9. Click 'OK' and wait.
10. More details on the the entire process can be found at <https://imagej.net/N2V>