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:

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:

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

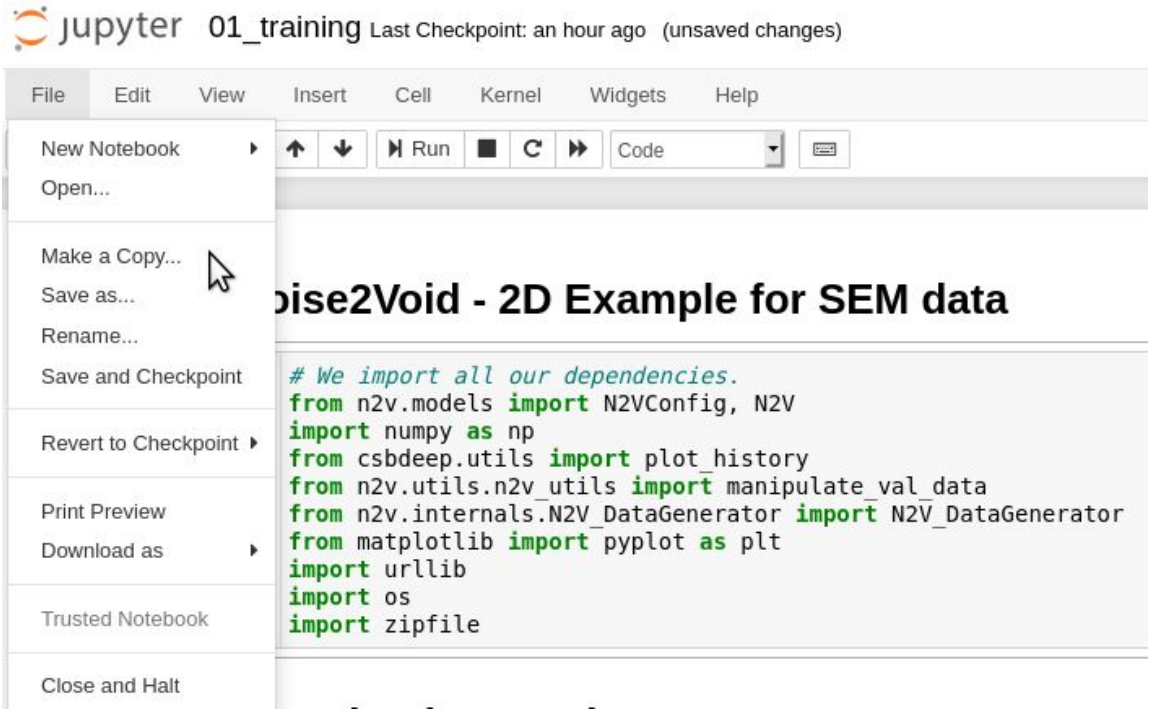
Choose any of the examples (except the 'structN2V' examples, which will be the subject in a separate exercise further down below) and execute each cell of the notebooks to train a network and then denoise some images.

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.



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 [ ]:
# We load all the '.tif' files from the 'data' directory.
# If you want to load other types of files see the RGB example.
# The function will return a list of images (numpy arrays).
imgs = datagen.load_imgs_from_directory(directory = "PATH/TO/YOUR/DATA")

# Let's look at the shape of the images.
print(imgs[0].shape)
# The function automatically added two extra dimensions to the images:
# One at the beginning, is used to hold a potential stack of images such as a movie.
# 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.

```
eps_per_epoch is set to (number of training patches)/(batch size), like this each training patch
 once per epoch.
VConfig(X, unet_kern_size=3,
        train_steps_per_epoch=int(X.shape[0]/128), train_epochs=10, train_loss='mse', batch_norm=True,
        train_batch_size=128, n2v_perc_pix=0.198, n2v_patch_shape=(64, 64),
        n2v_manipulator='uniform_withCP', n2v_neighborhood_radius=5, structN2Vmask = [[0,0,1,1,1,1,1,1,1,0]])

ok at the parameters stored in the config-object.
))
```

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