CARE Lab - Exercises

Learning Objectives

- ❏ Train CARE networks using the example notebooks.
- ❏ Understand how to use CSBDeep and CARE.
- ❏ Use Tensorboard to monitor the training process (while ongoing).
- ❏ Learn how to adapt the examples to use your own data.
- ❏ Bonus: use a trained network from within Fiji

Preamble: Be sure you have followed the installation instructions in the separate document 'CARE Lab - setup'. If you did so, you should be good to go!

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 CARE network

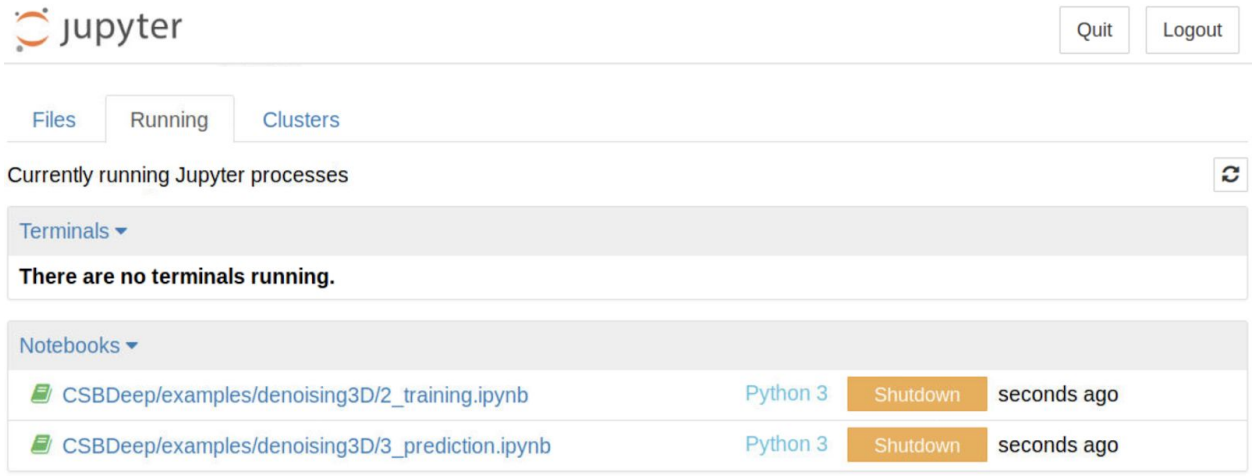
The GIT repo 'CSBDeep' you have previously cloned into '`<some_folder>/CSBDeep`' contains multiple example notebooks:

Every example will be downloading all required training data and is itself divided into three individual notebooks that need to be executed in sequence:

1_datagen.ipynb - network training usually happens on batches of smaller sized images then the ones recorded on a microscope. Hence, this notebook loads all your image data and chops it into many smaller pieces and stores it into the sub-folder '`data`' (you can see that folder on the screenshot below, but likely not yet in your own example folder).
Open this notebook, read all explanations, execute all cells, ask any questions that come up - then continue below...

2_training.ipynb - this notebook will train your CARE network. All outputs will be put into the folder '`models`'. While you execute this notebook, see what files will be generated inside the models-folder.
As you can see, the example notebooks are containing quite some additional explanations and some cells that have the purpose of showing you the data you are about to use and some sampled results. In this way you can be sure that the right things are happening.
Once you reach the cell that is actually starting the network training, you can go ahead and use Tensorboard (as described in the notebook) to monitor the training process. If you did not change the default settings you will have between 10 and 20 minutes to play with Tensorboard before the training of your first CARE network will be done. Enjoy (and please ask questions if you have any)!
At the very bottom you see, that we can even even export a so-called Tensorflow Saved Model'. Such a zip file contains all data and metadata to fully defined a Tensorflow model. We will later, in a bonus exercise, use such a saved model to apply a trained CARE network to raw data from within Fiji.

3_prediction.ipynb - the last of the three notebooks can be used to apply the network we have previously trained on any dataset you'd like to. Open the notebook and see what it does.
Note: if you happen to run into weird out-of-memory errors, you will need to shutdown notebooks that occupy GPU memory but are likely not used any longer. One good way to do so is by clicking on the 'Running'-tab in Jupyter, then on the orange 'Shutdown'-button next to the notebooks that are not longer needed:



Exercise #2: What is going on behind the scenes?

Seek to understand what happened behind the scenes while executing the three notebooks you just executed in Exercise 1.

Try to answer the following questions:

1. Where is the training and test data located?
2. How does data have to be stored so that CSBDeep will find and load it correctly?
3. Where are trained models stored? What models are stored? What is the difference between them?
4. Where in the notebooks is the name of the saved models determined?
5. How can you influence the number of training steps per epoche? What did you use (likely in the interest of time) and what value is suggested?
6. While this is not done is the example notebooks, how would you load an existing CARE network and continue to train it?
7. How would you do the same thing in such a way that the additionally trained model is stored separately (under a new name)?

Exercise #3: Train a CARE net without clean data.

Data by Reza Shahidi and Gaspar Jekely, Living Systems Institute, Exeter

In this exercise you will start with raw data and decide for yourself how to train a CARE network. The data contains the same sample, imaged at different levels of noise. Use CARE to improve the quality of noisy images. You will have to make decisions -- make them!

1. Download the data from https://tinyurl.com/yxlqqgm2 and unzip it into the **data** directory.
2. The file contains 2 *tif-stacks*, one for training and one for testing. Open **train.tif** or **test.tif** in Fiji or with python to look at the content. Each stack contains 7 images of the same tissue that were recorded with different scan time settings of a Scanning Electron Microscope (SEM):
    - Image 0 (1 in Fiji) is recorded with 0.2 μs scan time
    - Image 1 (2 in Fiji) is recorded with 0.5 μs scan time
    - Image 2 (3 in Fiji) is recorded with 1 μs scan time
    - Image 3 (4 in Fiji) is recorded with 1 μs scan time
    - Image 4 (5 in Fiji) is recorded with 2.1 μs scan time
    - Image 5 (6 in Fiji) is recorded with 5.0 μs scan time
    - Image 6 (7 in Fiji) is recorded with 5.0 μs scan time avg. of 4 images
3. Make a copy of the **1_datagen.ipynb**. Rename it to **1_datagenSEM.ipynb**.
4. How would you train a network to denoise images of 1 μs scan time? Which images do you think could be used as input and which as target?
5. Open the **training.tif** and save respective images from the stack into the **train/low** and **train/GT** folders. You can use Fiji, or the **imwrite** function from **tifffile:**

```
In [2]:
from tifffile import imwrite
data=imread('data/SEM/train/train.tif').astype(np.float32)
imwrite('out.tif',data[0])
```

6. Modify your **1_datagenSEM.ipynb** to work with your data and run it. Note that you are using 2D images instead of 3D stacks now.
7. Make a copy of **2_training.ipynb**, modify it accordingly and train a network on your data.

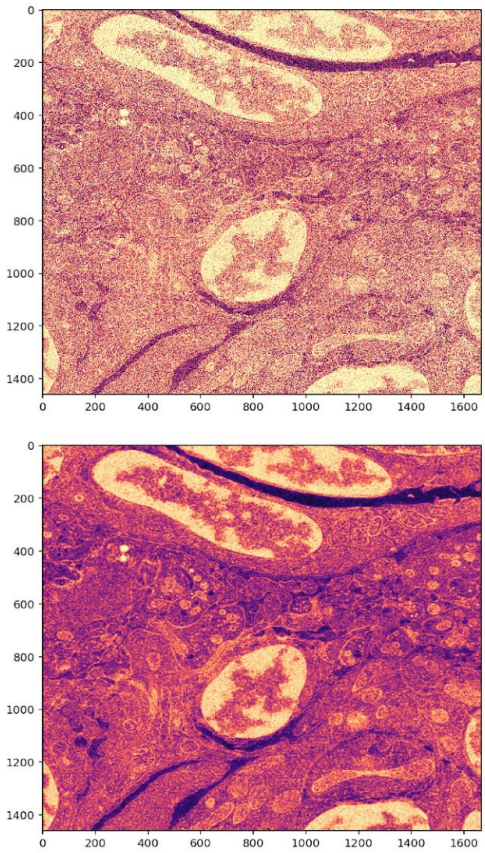
8. Make a copy of **3_prediction.ipynb** and modify it accordingly. Open **test.tif**, process it and look at the results for the different acquisition times.

In [14]:

```
# Load data, we are looking at the 1.0 usec data (slice 3)
Xtest=imread('data/SEM/test/test.tif').astype(np.float32)[3,:,:,np.newaxis]
restored = model.predict(Xtest, 'YXC')

#Show input, we are zooming in a bit.
plt.figure(figsize=(7,7))
plt.imshow(Xtest[1500:,:,0],cmap="magma")
plt.show()

#Show result, we are zooming in a bit.
plt.figure(figsize=(7,7))
plt.imshow(restored[1500:,:,0],cmap="magma")
plt.show()
```



9. Can you further improve your results by using the data differently or by tweaking the settings? How could you train a single network to process all scan times? Be creative! Surprise us!

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

You can use any `TF_SavedModel.zip` you created at the very bottom of any example notebook `2_training.ipynb`. Still, in case you want to use one we trained, feel free to use the model from below.

Fallback CAREd tribolium denoising model: *http://tinyurl.com/yysy5n2l*

Ok, with a trained and saved network at hand, please open Fiji and follow these 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. *Note:* you can find much more info here (but don't need it now)...
    - http://csbdeep.bioimagecomputing.com/
    - https://github.com/csbdeep/csbdeep_website/wiki/CSBDeep-in-Fiji
6. Restart Fiji
7. Open adequate data, for the provided tribolium network (or any other example network trained with CSBDeep), use adequate file from here: https://tinyurl.com/y2bhcylh
8. Now start "Plugins > CSBDeep > Run your network"
9. Click on "Browse" and select the trained network ZIP-file we downloaded from Google Colab.
10. Hit OK and wait… this will likely take quite a while because the network will not be run on your GPU, but CPU.
    Speeding this up is possible: https://tinyurl.com/y4bbjhc7 (see on bottom of that page)