CellDetective: User Manual

Rémy Torro

August 9, 2023

1 Scope

2 Installation

3 Quickstart

4 My first experiment

4.1 Data structure

The must now be manually dropped into each position folder of each well.

4.2 Recommended preprocessing

4.2.1 Image alignment

Carry the image alignment (or registration) using Fiji and our provided macros. Run through the position/well structure.

4.2.2 Image normalization

Adhesion study: exploit the multiposition information to reconstruct a background from each experimental condition. Use provided macros (Fiji or python, but propose a macro that runs through the position/well structure).

5 Segmentation

5.1 Import a segmentation model

5.1.1 Cellpose

The training of a Cellpose model can be carried using a command line interface or a GUI. By convention the user should provide a channel showing the cell membrane and an optional nucleus channel, to be used as anchor for segmentation. Obviously, as a Deep Learning model, it is also possible to pass an combination of channel as the input to the network (we will just not use the nucleus anchor technique). The Cellpose CLI provides an *–all_channels* key that relaxes the input condition on the model.

CellDetective allows the user to import a custom-trained Cellpose model directly into the interface. We account for the variety of possible Cellpose models by allowing the user to clearly define the input parameters. The pretrained model name (using the CLI, python API or the Cellpose GUI) is parsed to set the residual, style and concatenation settings. The CLI naming convention automatically accounts for this, we recommend not to rename the models if you intended to deviate from the *res=on*, *style=on*, *concatenate=off* setting.

If you use the *–all_channels* tag during training, it is important that when reloading the pretrained model, Cellpose reads the relevant channels. According to the source code, this mode is triggered when *model_type=None* in the CellposeModel instance and *channels=None* in the eval function. This condition will trigger the automatic detection of the number of channels from the shape of the image to predict. Currently, setting *channels=None* breaks Cellpose. We opened a request to fix this bug. Otherwise, the eval method of CellposeModel triggers the eval method of the UnetModel. Replace channels=channels[i] if (len(channels)==len(x) and (isinstance(channels[i], list) or isinstance(channels[i], np.ndarray)) and len(channels[i])==2) else channels with channels.

Update: in order to truly activate the *–all_channels* mode, one must also pass *–pretrained_model None* to not take *cyto* as a starting point. The number of input channels should be automatically detected (I tested it on ADCC MCF7 nuclei data with 3 channels: BF, PI, H). The following line: *»» training network with 3 channel input ««* should be displayed. The model can then be reloaded as follows:

```
model = CellposeModel(gpu=True,
 model_type=None, pretrained_model="/home/limozin/Desktop/MCF7_nuclei_cellpose/dat
 diam_mean=30.0,
 nchan=3
)
```