Tutorial: Cellpose and Cellpose on ImageJ

I]- From your own computer

Cellpose is a Python 3,8 script, that we want to use remotely from the Python 2,7 implementation in ImageJ.
First thing to do is set up the proper environment to make Cellpose work in.

1) Set up the conda environment

A virtual environment is basically a directory that contains all the different files and script that we're going to need. The easiest way to set up a very nice Python environment is by downloading Anaconda. Follow what the installation set up says for a smooth set up. Congratulations, you can now use Python on your computer!

You can now see that you can interact with Anaconda in a few ways. For example you can open the Anaconda Navigator for a nice user interface (which slows down everything else, so I do not recommend using it, but it looks nice), or more importantly the Anaconda command prompt. You need to open one for the next steps.

Anaconda comes with «conda», which is the thing that handles the different environment that we may need. The next step is to set up a conda environment that contains Cellpose. In the Anaconda prompt that you just opened, type:

```
conda create --name cellpose python=3.8
```

This creates an environment called cellpose, that contains everything needed to run python scripts. There are newer versions of Python that you could use, but this is the recommended set up by the authors.

Next up is actually installing Cellpose in there. For that, type:

```
conda activate cellpose
```

This will tell conda that we want to work in the environment called cellpose. Then type:

```
python -m pip install cellpose[gui]
```

This tells conda to execute the python command «pip», which handles every new package installation or update, and to fetch the Cellpose program, along with its user interface (because it's nice for checking that everything works fine, but it takes a lot of space so you can just type cellpose instead of cellpose[gui] if you don't want it). This will take a bit of time, and you might get asked a few times to confirm that you want to install some packages. Just type «y» and hit Enter, we need those packages.

Well done, the environment is working on its own now! If you want to check, type a quick

```
python -m cellpose
```

This should open the cellpose user interface.

2) Preventing path issues and incompatibilities

When running Cellpose from ImageJ, the scripts actually opens the command prompt from the computer and tells it to run the Cellpose that we just installed. If you've been following, I said command prompt, not Anaconda prompt. The Anaconda prompt is for running commands from inside Anaconda, the command prompt is from running commands from the computer's system. However, we just said that our environment, that makes Cellpose work, is in Anaconda. Thus, we need to enable the command prompt to access our conda environments, because ImageJ can access the command prompt but not the Anaconda prompt. Does that make sense?

For that, open the command prompt with windows+R, then type cmd. You'll notice that this looks VERY similar to the Anaconda command prompt. Just type

```
conda init
```

And you're done! That was a lot of explanation for a tiny line.

This should be enough.

HOWEVER.

Sometimes, the stars align in a really bad way. If you have some troubles while running Cellpose, I would recommend following those next steps, but in theory they should be optional.

What might happen is that your computer still doesn't recognize conda, and is desperatly trying to do its best with something that it cannot find. To correct this, we will add conda to the PATH variable. *You will need administrator's rights over your computer for that part*. Open the Advanced System Settings of your computer and go to the «Environment variables» section (it should be at the bottom). Select Path, then Edit, and then New. Add every direction that contains conda, in order for your computer to know where to look when you call conda.

To be sure to have the right directions, you can type

```
where conda
```

In the command prompt, and it will give you where conda is (yes, pretty straightforward). Just copy-paste those directions in the Path variable.

Otherwise, you can just trust me and add those as new paths:

```
C:\Users\username\anaconda3\Scripts
C:\Users\username\anaconda3\Library\bin
C:\Users\username\anaconda3\condabin
```

And we are done for this part!

3) Make Cellpose work from ImageJ

Of course, you can now use Cellpose directly from the user interface if you installed it.

In my opinion, it is probably the best option if you have two images that need quick analysis.

However, the most probable situation is that you want to analyze a whole batch of images, and you might not want to sit in front of your computer manually changing the file every time. Or, even better, you figured it would be nice to have Cellpose work in the middle of a whole pipeline analysis in ImageJ.

Then the solution is to install the Cellpose wrapper for ImageJ. Basically, it will make the link between ImageJ and our Cellpose environment. I personally prefer this, because it avoids using the Cellpose user interface, meaning I can actually do something else with my computer in the meantime.

To do that, open ImageJ. Go to Help → Update → Manage update sites, and check «PTBIOP». Then, run all of the updates.

That's it! Now you can work Cellpose by going to Plugins → BIOP → Cellpose.

Last thing before having fun with this program: have a quick look at Plugins → BIOP → Cellpose → Cellpose setup…
The version of cellpose (at the bottom) should be the closest to the one you installed (version 2,0 at the moment). Also, the first line should match the direction you have for your conda environment. Depending on how you installed it, this may vary. It will either be the default version, or you will need to change the «.conda» section to «anaconda3» (this one is the most likely if you followed this tutorial).

And we're done! Have fun!

II]- From the HIVE

Good news: everything you need to use Cellpose on the HIVE, both with or without ImageJ, has already been installed!

Does this mean you don’t have to do anything and it will work perfectly? Sadly no, at least not for everything.

1) Running the Cellpose user interface

Open an Anaconda prompt and type:

```
conda activate cellpose
python -m cellpose
```

That’s it!

2) Running Cellpose on ImageJ

Go to Help → Update → Manage update sites, and check «PTBIOP». Then, run all of the updates.

Now you can work Cellpose by going to Plugins → BIOP → Cellpose.

Last thing before having fun with this program: have a quick look at Plugins → BIOP → Cellpose → Cellpose setup…
The version of cellpose (at the bottom) should be the closest to the one you installed (version 2,0 at the moment). The first line points to the directory where cellpose is in the HIVE. It should be C:\ProgramData\anaconda3\envs\cellpose.

When running Cellpose, the path for the models should be C:\Users\username\.cellpose\models\model_name.

III]- Cellpose and ImageJ : tips and tricks

1) Command lines

Here’s the python command line to run Cellpose from ImageJ:

```
from ij import IJ

IJ.run("Cellpose Advanced (custom model)", "diameter=30 cellproba_threshold=0.0 flow_threshold=0.4 anisotropy=1.0 diam_threshold=12.0 model_path=C:\Users\username\.cellpose\models\cytotorch_0 model=cyto nuclei_channel=0
```

cyto_channel=1 dimensionmode=2D stitch_threshold=0 omni=false cluster=false additional_flags='')

This will run Cellpose using the default parameters and the «cyto» model. It indicates that there is no channel containing nuclei staining, and one channel containing cytoplasm staining. Every argument must be present, even if empty (here : additional_flags=). Model path should be changed to match your own.

2) Cellpose input settings

Diameter: expected diameter of the cells to detect. The algorithm is flexible and will detect cells for which the diameter is not too far from this value. Setting this parameter to 0 means that Cellpose will take the diameter value from the model currently being used (recommended for trained models).

Flow_threshold: default 0.4, can be set between 0 and 1. If Cellpose outputs too many weird-shaped ROIs, lower the threshold. However, if Cellpose seems to be missing a lot of cells, raise the threshold (I use 0.6).

Cellproba_threshold: should be set to 0 in most cases. If Cellpose seems to be missing a lot of cells, you can lower it (I used -1.0 for one model).

Additional_flags: depending on what you want to do with the cellpose output and how it should be computed, there are a variety of arguments that can be passed on here. Go to this page for additional information on how to write those arguments and see this post for the complete list of what you can pass on here.

For more information on settings, see the documentation.

3) Training a model

As a side note: whenever possible, you should process your images before passing them on into Cellpose, to remove some noise and get a better quality overall.

The pre-trained models in Cellpose usually work quite well. However, it may happen that your dataset is a bit too noisy, or that you might just want more precision.

Training a model requires a bit of preparation. Create a folder that contains a fair number of pre-processed images that you want to use for training. Open the cellpose GUI and load in the first image. Choose the model that you think already matches your data the better. You could also train a model from scratch.

Follow the «Training instructions» in the Models menu: I don’t have anything to add to them. You will need time (if you keep the default settings, which I recommend), even on the HIVE. Training on a dozen images takes half a day.

The trained model will be in C:/Users/username/.cellpose/models, and in the folder you created. Its name will also appear in the gui_models text file.

4) Using someone else's model

Download the file containing the model and put it in C://Users/username/.cellpose/models. Add its name in the gui_models text file. It will be available to use both from the GUI and from ImageJ.

5) About the output

5.1: in the Cellpose GUI

The cellpose GUI takes an image (lots of formats are accepted, including tiff) and outputs a file that contains both the images and its «processing», meaning the ROIs it found. Via the «File» menu, those ROIs can be saved under several formats, depending on what's most usefull for your dataset and what you want to do with the images.

In the GUI, you can actually modify those ROIs if necessary. Use right click + draw to add a ROI and Ctrl+click to delete one.

If you want to import those ROIs in ImageJ, you want to export them from cellpose using «Save Outlines as text for ImageJ». The default version of ImageJ doesn't know what to do with those files, so we need to install a Plugin that will be able to handle them properly. This plugin is called «imagej_roi_converter», it can easily be found here. Download it as a python script and put it in your Fiji distribution (in my case: fiji-win64/Fiji.app/plugins). When you restart ImageJ, you will find it in the Plugins menu.
This method is far from being the most efficient way to analyze images, but for a small number of images it might actually be quicker to use this instead of taking the time to write a full script that allows you to modify and retrieve the ROIs in ImageJ.

5.2: through the Cellpose wrapper for ImageJ

When running through ImageJ, cellpose will detect cell shape and output a mask containing the found ROIs.
The way it does that is by not directly giving the ROIs, but instead drawing a mask where all the found areas are represented by a unique pixel intensity value, in grayscale.

If all the areas are nicely separated, a quick way to get the ROIs is to threshold the mask from 1 to the max intensity, then to run Analyze Particles on it (with the «Add to Roi Manager» box checked).

If some of the areas are adjacent, they won't be properly separated by Analyze Particles. In that case, this code will do the job:

```
from ij import IJ, WindowManager, Prefs
from ij.measure import ResultsTable
from ij.plugin.frame import RoiManager

rm = RoiManager.getRoiManager()
mask = IJ.getImage()
IJ.run(mask, "Select All", "")
roi = mask.getRoi()
rm.addRoi(roi)
rm.runCommand(mask,"Measure")
table = ResultsTable.getResultsTable()
max_int = table.getValue("Max", 0)
total_roi = range(1, int(max_int)+1)
rm.setSelectedIndexes([0])
rm.runCommand(mask, "Delete")
# Isolating T-cells
for value in total_roi:
	temp = mask.duplicate()
	IJ.setThreshold(temp, value, value)
	IJ.run(temp, "Convert to Mask", "")
	IJ.run(temp, "Create Selection", "")
	roi = temp.getRoi()
	rm.addRoi(roi)
	temp.close()
# Clearing useless windows for the next image
IJ.selectWindow("Results")
IJ.run("Close")
```

If you want to use that script as a plugin, save it as a python script and put it in your Fiji distribution (in my case : fiji-win64/Fiji.app/plugins), then restart ImageJ. As a site note, if you want the script name to be shown in the Plugin menu, it must contain at least one «_» and no blank spaces.