# **Instruction to Deep-SMOLM**

Contents

[**Instruction to Deep-SMOLM** 1](#_Toc109752481)

[**Preparing the environment and software** 3](#_Toc109752482)

[**Matlab** 3](#_Toc109752483)

[Install matlab here 3](#_Toc109752484)

[**Visual studio code (VS code)** 3](#_Toc109752485)

[Download visual studio code here: 3](#_Toc109752486)

[Learning debugging in visual studio code here: 3](#_Toc109752487)

[**Anaconda** 3](#_Toc109752488)

[Download the Anaconda here: 3](#_Toc109752489)

[Build environments for Deep-SMOLM: 3](#_Toc109752490)

[**Install packages** 4](#_Toc109752491)

[Launch visual studio from anaconda 4](#_Toc109752492)

[Change the environment to the new created environment through terminal in VS code 4](#_Toc109752493)

[Install packages through conda using terminal 4](#_Toc109752494)

[**Generating the training data** 5](#_Toc109752495)

[**Option 1**: generating training data from matlab scripts 5](#_Toc109752496)

[**Option 2**: use training data in OSF 5](#_Toc109752497)

[**Training Deep-SMOLM** 6](#_Toc109752498)

[**Set comet ml account to monitor the training process** 6](#_Toc109752499)

[**Modify the config\_orientations.json** 6](#_Toc109752500)

[Parameters you might need to change 6](#_Toc109752501)

[Parameters you can keep it be the same 8](#_Toc109752502)

[**Running the training code** 8](#_Toc109752503)

[**Estimation using Deep-SMOLM** 8](#_Toc109752504)

[**For simulated data with ground truth** 8](#_Toc109752505)

[Check if the data loader method match to your data 8](#_Toc109752506)

[Modify the config\_orientations.json file 8](#_Toc109752507)

[Run the estimation code 9](#_Toc109752508)

[**For Experimental data** 9](#_Toc109752509)

[Check if the data loader method match to your data 9](#_Toc109752510)

[Modify the config\_orientations.json file 9](#_Toc109752511)

[Run the estimation code 10](#_Toc109752512)

# **Preparing the environment and software**

## **Matlab**

Matlab is used to generate the simulate data

### Install matlab here

[Installation and Licensing Documentation (mathworks.com)](https://www.mathworks.com/help/install/)

## **Visual studio code (VS code)**

I recommend use visual studio code for running Deep-SMOLM due to its user-friendly debugging interface. But any other platform will also work.

### Download visual studio code here:

[Download Visual Studio Code - Mac, Linux, Windows](https://code.visualstudio.com/Download)

### Learning debugging in visual studio code here:

<https://code.visualstudio.com/Docs/editor/debugging>

## **Anaconda**

Using anaconda for installing packages

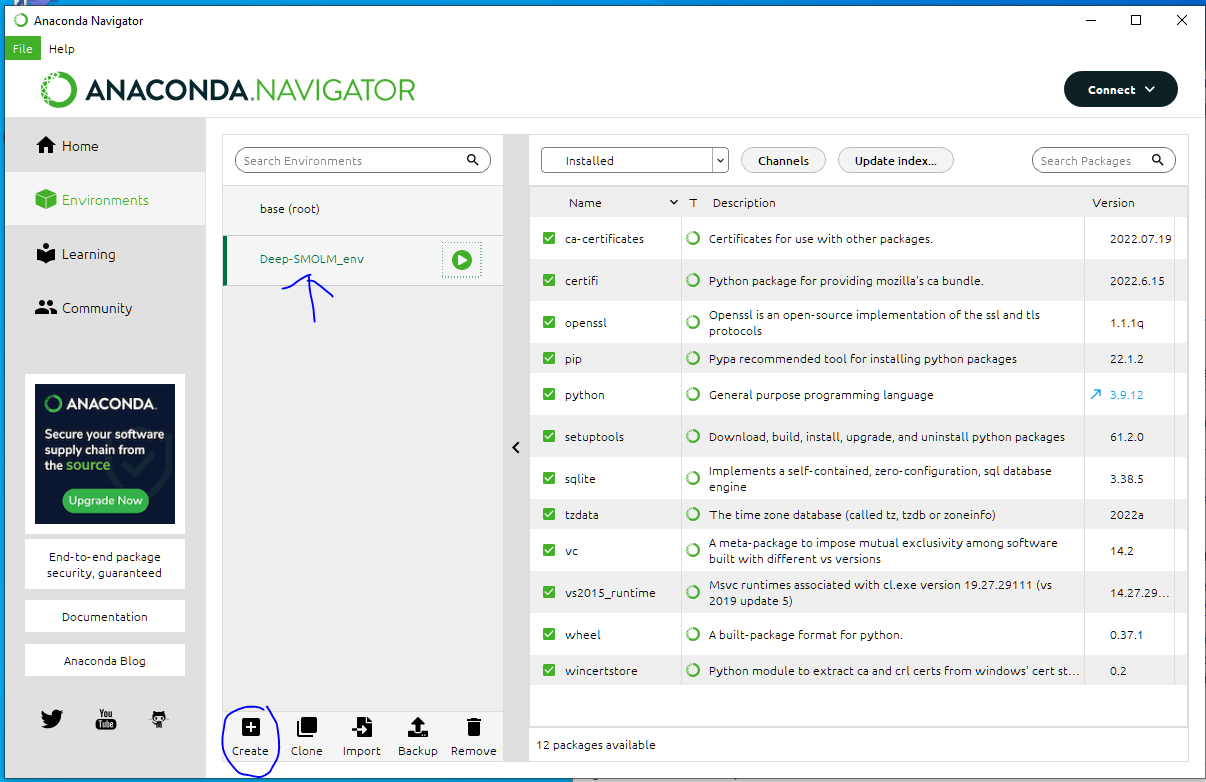
### Download the Anaconda here:

[Anaconda | The World's Most Popular Data Science Platform](https://www.anaconda.com/)

### Build environments for Deep-SMOLM:

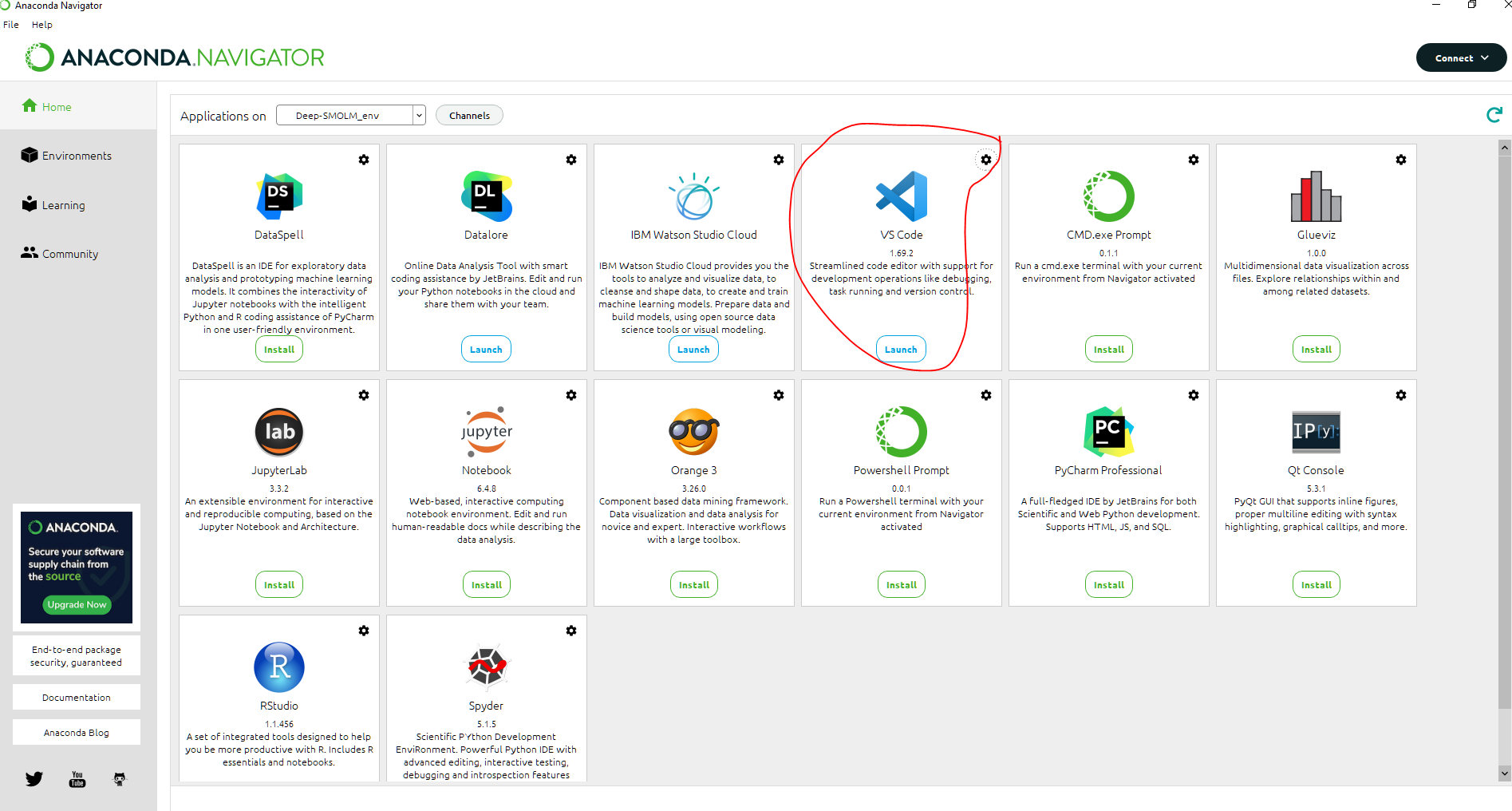
For different algorithms, they are might designed based on different environments. It would be good to have a specific environment for Deep-SMOLM and build related packages inside this environment.

In here, I created a new environment named Deep-SMOLM\_env



## **Install packages**

### Launch visual studio from anaconda



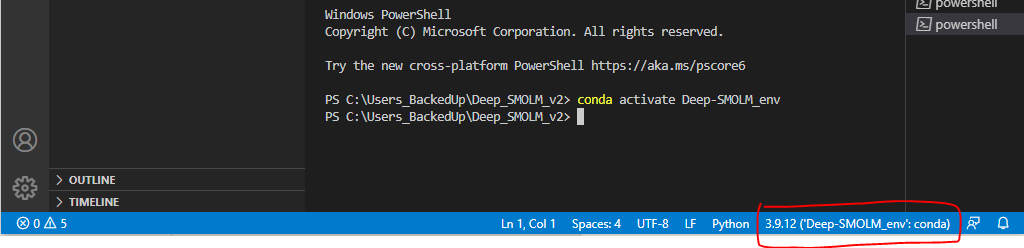
### Change the environment to the new created environment through terminal in VS code



Type in: conda activate [new environment name]



If success, you will see that the visual studio is running under the designed environment



### Install packages through conda using terminal

Make sure the previous changing environment is successful. Otherwise, the packages won’t install in the desired environment.

Type in the following command in the terminal of visual studio one by one. This will take some time.

conda install -c pytorch pytorch

conda install -c anaconda numpy

conda install -c comet\_ml comet\_ml

conda install -c anaconda scipy

conda install -c conda-forge tifffile

conda install -c conda-forge matplotlib

conda install -c anaconda pillow

conda install -c conda-forge opencv

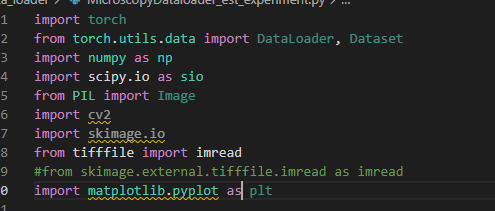
conda install -c anaconda scikit-image

conda install -c pytorch torchvision

conda install -c conda-forge tqdm

Once you successfully installed all the packages; the wave underlines in the code will disappear

Examine all the .py files in Deep-SMOLM, make sure there is no wave underlines. If there is, install the related package



# **Generating the training data**

There are two options for getting the training data

## **Option 1**: generating training data from matlab scripts

* All the simulated data is generated using Matlab scripts under Deep\_SMOLM\_v2\forward\_model\_pixOL
* Generate training data using generate\_images\_opt\_large\_dataset\_more\_info.m for pixOL microscope without aberration;
* Generate training data using generate\_images\_opt\_large\_dataset\_little\_3D\_cubic\_model.m for calibdated pixOL microscope; This code generate training images containing emitters that is imaged with focal plane randomly sampled from [-150,150] nm

## **Option 2**: use training data in OSF

Instead of generating your own training data, you can also directly use the training data stored in the OSF

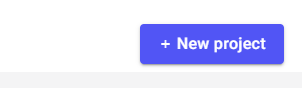
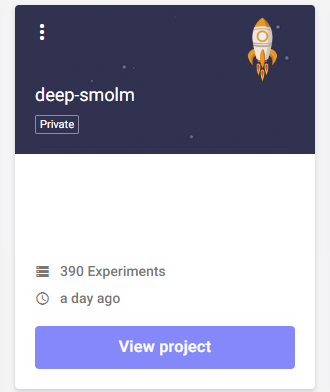
https://osf.io/x6p8r/?view\_onlyb263a8693c5e4418a0b962df31ca0101

# **Training Deep-SMOLM**

## **Set comet ml account to monitor the training process**

I use comet.ml to monitor the training process.

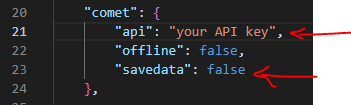
* Create your comet.ml account here: [www.comet.com](http://www.comet.com)
* Create new project

* Read and API of the new project and change it to the configuration

Read your API key number under the account profile. (<https://www.comet.com/docs/rest-api/getting-started/>)

At the Deep-SMOLM file: config\_orientations\_v2.json, enter your API key and change the “savedata” option to be true if you want to turn on the monitoring



## **Modify the config\_orientations.json**

config\_orientations\_v2.json is a file that contains all the information about your training, testing, validation parameters.

### Parameters you might need to change

* Name of the training; other names (name1, name2) won’t be used in the running.

"name": "training\_perfect\_pixOL",

* Information of your comet.ml account; the training/testing loss will be save to this commet

    "comet": {

        "api": "your API key",

        "offline": false,

        "savedata": false

    },

* Change the “save\_dir”: directions you want save the trained results

    "trainer": {

        "epochs": 100,

        "warmup": 0,

        "save\_dir": "../data/save/",

        "save\_period": 1,

        "verbosity": 2,

        "asym": false,

        "sym": false,

        "percent": 0.9,

        "subset\_percent": 1.0,

        "monitor": "min test\_loss",

        "early\_stop": 10

    },

* Change the “batch\_size” based on your GPU’s memory

    "data\_loader": {

        "type": "MicroscopyDataLoader",

        "args":{

            "batch\_size": 32,

            "shuffle": false,

            "validation\_split": 0,

            "num\_workers": 16

        }

    },

* Change your training data’s name, location, and size

“noise\_image\_name”: the passion shot noise is added during the training, so please use noiseless image

“GT\_image\_name”: name of the ground truth image

“file\_folder”: location where you save your training data

“number\_images”: number of the training data

Again: “noise\_image\_name1”, “noise\_image\_name1”,”file\_folder1” won’t be read in the running code, they are just for record to easily switching between different options

    "training\_dataset": {

        "noise\_image\_name1": "image\_with\_poission\_bkgdRmvd\_up",

        "noise\_image\_name2":"image\_with\_poission",

        "noise\_image\_name":"image\_noiseless",

        "background\_name":"",

        "GT\_image\_name": "image\_GT\_up",

        "GT\_list\_name": "",

        "file\_folder": "../data/opt\_PSF\_data\_1000vs2/training\_20220711\_pixOL\_SNR1000vs2\_gamma\_linear\_photon\_poisson\_distribution\_gradient\_model/",

        "file\_folder1":"../data/opt\_PSF\_data\_1000vs2/training\_20220526\_pixOL\_SNR1000\_2\_gamma\_linear\_distribution\_photon\_poisson/",

        "number\_images":30000

    },

### Parameters you can keep it be the same

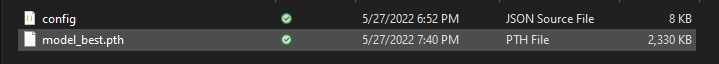
Please understand other parameters by reading through the code

## **Running the training code**

* Once you modify the config\_orientations.json, you can directly run the code: Deep\_SMOLM\_main.py either through running or debug running.

I highly recommend running through the debug mode, so that if something is run, you can easily debug the code

* You can monitoring your training curves at [www.comet.ml](http://www.comet.ml)
* Once the running is finished, you will find your results at the folder your assigned for saving the data



# **Estimation using Deep-SMOLM**

## **For simulated data with ground truth**

Use Deep\_SMOLM\_est.py

### Check if the data loader method match to your data

* Modify the MicroscopyDataloader\_est.py data reading code to fit into your data if needed.

Current reading method: each estimation paired image is saved into one .mat file; the code read the data batch by batch

### Modify the config\_orientations.json file

    "est\_dataset": {

        "noise\_image\_name":"image\_with\_poission",

        "GT\_image\_name": "image\_GT\_up",

        "background\_data":"img\_bkg\_",

        "GT\_list\_name": "GT\_list",

        "file\_folder4": "../data/opt\_PSF\_data\_1000vs2/phantom\_20220616\_dense\_SMs\_1000vs2/",

        "file\_folder2": "../data/opt\_PSF\_data\_1000vs2/phantom\_20220711\_dense\_SMs\_1000vs2\_omega\_2/",

        "number\_images":4000,

        "batch\_size": 16,

        "save\_name":"phantom\_orientation\_estimation\_dense\_SMs.mat"

    },

### Run the estimation code

* Use Deep\_SMOLM\_est.py
* Give the direction of Deep-SMLM model

args.add\_argument('-r', '--resume', default="your direction/model\_best.pth", type=str,help='path to latest checkpoint (default: None)')

## **For Experimental data**

Use Deep\_SMOLM\_est\_experiment.py

### Check if the data loader method match to your data

* Modify the MicroscopyDataloader\_est\_experiment.py data reading code to fit into your data if needed.

Current reading method: a stack of images is saved in the .tiff format; the code read the offset image, and read the single molecule image batch by batch

### Modify the config\_orientations.json file

"est\_dataset\_experiment": {

        "noise\_image\_name": "\_centerY\_y410\_x\_200\_FoV420\_220\_",

        "GT\_image\_name": "",

        "file\_folder": "../data/experiment\_data/20220530\_A\_beta\_amyloid8\_data9\_16/",

        "batch\_size":16,

        "number\_images\_per\_dataset":1000,

        "number\_FoV": 14,

        "number\_dataSet": 5,

        "starting\_dataSet":12,

        "upsampling":6,

        "offset\_name":"offset\_centerY\_y410\_x\_200\_FoV420\_220\_",

        "background\_name":"\_bkg\_centerY\_y410\_x\_200\_FoV420\_220\_",

        "tophoton": 0.29,

        "y2x\_channel\_ratio":1.145,

        "save\_name":"A\_beta\_data"

    }

### Run the estimation code

* Use Deep\_SMOLM\_est\_experiment.py
* Give the direction of Deep-SMLM model

args.add\_argument('-r', '--resume', default="your direction/model\_best.pth", type=str,help='path to latest checkpoint (default: None)')