

---

# gratin Documentation

*Release latest*

Hippolyte Verdier

Apr 24, 2023



# CONTENTS

- 1 Getting started 3**
  - 1.1 Training . . . . . 3
  - 1.2 Tests on simulations . . . . . 3
  - 1.3 Experimental trajectories . . . . . 4
- 2 Installation 5**
- 3 Contents 7**
  - 3.1 Simulation-based inference . . . . . 7
  - 3.2 License . . . . . 7
  - 3.3 Contributors . . . . . 7



This is the documentation of **Gratin** (*Graphs on Trajectories for Inference*).

It is a tool to characterize trajectories of random walks, i.e. motion driven by random fluctuations. This type of motion is observed at various scales and in a wide diversity of systems. While this package was developed for the purpose of analysing experimental data coming from photo-activated localization microscopy (PALM) experiments, nothing prevents it from being used on random walk recordings coming from other experimental setups and other domains !

To extract *summary statistics* describing trajectories, Gratin mixes two ingredients :

- an original neural network architecture using graph neural networks (GNN)
- an inference scheme : *Simulation-based inference*



## GETTING STARTED

### 1.1 Training

It only takes one function to train a model fitting your experimental data in terms of trajectory length, localization uncertainty, diffusivity range and time interval !

```
from gratin.standard import train_model

model, encoder = train_model(
    export_path = "/path/to/model", # indicate an empty folder where to store the model,
    ↪once trained
    num_workers = 4, # number of workers used to simulate trajectories during the,
    ↪training phase
    time_delta = 0.03, # time separating two successive position recordings in your,
    ↪trajectories (exposure time of the camera)
    log_diffusion_range = (
        -2.0,
        1.1,
    ), # log-diffusion is drawn following a truncated centered gaussian in this range
    length_range = (7, 35), # length is drawn in a log-uniform way in this interval
    noise_range = (
        0.015,
        0.05,
    ), # localization uncertainty, in micrometers (one value per trajectory)
    max_n_epochs = 100 # Maximum epochs on which to run the training.
)
```

### 1.2 Tests on simulations

Once the model is trained, you can check its performance on simulated data using the `plot_demo()` function. This will print the mean absolute error of the prediction of the anomalous diffusion exponent, and the F1 score of the random walk model classification task. This also plots embeddings of trajectories.

Note that you can specify traits of the trajectories on which you wish to test it, using the same parameters as the `train_model()` function. This is useful if you wish to test the model on data different from what it has been trained on. See *Simulation-based inference* for more details about the training procedure and the considered types of random walk.

```
from gratin.standard import load_model, plot_demo

model, encoder = load_model(export_path="/path/to/model")
plot_demo(
    model,
    encoder,
    length_range = (7, 55), # these values can differ from those used during training
    noise_range = (0.015, 0.05)
)
```

## 1.3 Experimental trajectories

Finally, to use a trained model to get embeddings of your own trajectories along with predictions of the anomalous diffusion exponent and of the random walk type, you can use the following function, where `trajectories` is a list of  $(\cdot, D)$  Numpy arrays representing D-dimensional trajectories (coordinates are assumed to be chronologically ordered).

```
from gratin.standard import get_predictions

df = get_predictions(model, encoder, trajectories)
# Returns a pandas DataFrame with prediction results
```

All this is illustrated in the example notebook [here](#).



## INSTALLATION

To install Gratin on your machine, run

```
pip install gratin
```

---

**Note:** Gratin relies on the `torch-geometric` package, the installation of which depends on your version of CUDA and Torch, as well as your OS. Note that it is **not mandatory to have a graphic card at all** to run Gratin.

You'll find [here](#) the one-line-command that will install it on your machine.

---



## CONTENTS

### 3.1 Simulation-based inference

Simulation-based inference is when blah blah...

### 3.2 License

The MIT License (MIT)

Copyright (c) 2022 Hippolyte Verdier

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 3.3 Contributors

This work was developed by Hippolyte Verdier ([e-mail](#)) during his PhD at the [Decision and Bayesian Computation lab](#) at the Institut Pasteur, funded by Sanofi.

Thanks to Gert-Jan Both for precious advice on code structure and coding practices.

- [genindex](#)
- [modindex](#)
- [search](#)