Spikeometric: Linear Non-Linear Cascade Spiking Neural Networks with Pytorch Geometric

Jakob L. Sønstebø [1], **Herman Brunborg**[2]**, and Mikkel Elle Lepperød** [1,2]

1 Department of Numerical Analysis and Scientific Computing, Simula Research Laboratory, Oslo, Norway **2** Department of Physics, University of Oslo, Oslo, Norway

Summary

To understand the dynamics of the brain, computational neuroscientists often study smaller scale networks using simple cascade point-process models such as the Linear-Nonlinear-Poisson (LNP) model and the Generalized Linear Model (GLM) (Gerstner et al., 2014; Meyer et al., 2017; Paninski, 2004).

Stochastic models can give key insights into the behavior of a network on a systems level without explicitly modeling the subcellular mechanisms of each neuron. They lack some biological plausibility on the neuron level but have been shown to enjoy nice convexity properties, which can be fitted to observed spike data (Paninski, 2004). Traditionally, these are used as encoding models. For example, to study how multi-neuron systems process incoming stimuli (Pillow et al., 2008). Recently, these models have been used as generative models for inverse problems mapping activity to connectivity. As an example, (Das & Fiete, 2020) assessed bias and reconstruction errors in this setting.

This software expands on a simulator developed for testing novel reconstruction techniques using methods from the causal inference literature (Lepperød et al., 2020). It provides tunable generative models in a flexible framework based on PyTorch. The primary use case, so far, has been as a data-generator for inverse problems, but the framework can easily accommodate more complicated models for encoding applications.

Statement of need

Linear non-linear cascade models are much used in computational neuroscience and come in many flavors. Typical examples are SRM, LNP, GLM (Gerstner, 2008; Gerstner et al., 2014; Meyer et al., 2017). What unites these models is that they all model the spike response of a network through the same cascade-like sequence of steps. At each time step, a neuron receives input from its environment and converts it to a firing rate by a nonlinear function. This firing rate parametrizes a probability distribution from which spikes are drawn. This contrasts hard-threshold-based models, in which spikes are emitted whenever a variable representing the membrane potential exceeds a certain value. Although these models share many of the same underlying principles, no unifying framework currently permits easy implementation and direct comparison.

There exists a number of other simulation tools for simulating networks of neurons, notably NEST (Gewaltig & Diesmann, 2007), Neuron (Carnevale & Hines, 2006), Brian2 (Stimberg et al., 2019) and GENESIS (Bower et al., 2013). While all of these are primarily directed at solving systems where the dynamics are given by a system of differential equations, NEST includes a limited selection of pre-built stochastic point-process models similar to the ones found in spikeometric. However, this selection currently consists of only two models, which

are not included in the GPU compatible version NEST-GPU ([Golosio et al., 2021](); [Tiddia et al., 2022]()). There are also some `torch`-native frameworks like `BindsNET` ([Hazan et al., 2018]()) and `snnTorch` ([Eshraghian et al., 2023]()) but these focus on learning rather than simulation and are currently limited to LIF models. In general, `spikeometric` is more similar to the simulation tools in its focus, but has the automatic parameter tuning and `torch`-compatibility of the Spiking Neural Network libraries.

Implementation

The `spikeometric` package is a framework for simulating spiking neural networks using linear non-linear cascade models in Python. It is built on the PyTorch Geometric package and uses its powerful graph neural network modules and efficient graph representation. It is designed to be flexible and easy to use, while also being competitive on speed. Moreover, it's built in a way that accommodates usage or implementation of multiple different models sharing principles from the linear non-linear cascade family of models.

The `torch` backend makes simulating large networks on a GPU easy, with the extra benefit of having a familiar use pattern, reducing the friction of picking up a new tool. The package relies heavily on `PyTorch Geometric` ([Fey & Lenssen, 2019]()), with the networks being represented as `torch_geometric` `Data` objects and the models extending the `MessagePassing` base class. The `PyTorch Geometric` framework is a popular deep learning framework originally designed for Graph Neural Networks (GNNs), a class of neural networks for learning graph-related data ([Bronstein et al., 2021]()). It is the perfect setting for simulating neural networks with tunable parameters, allowing us to formulate the model's equations naturally in terms of vertices and edges, and giving us access to easy automatic tuning of parameters e.g. to match a certain firing rate, provided that the nonlinearity in the model is differentiable. The tuning functionality allows for fitting arbitrary parameters and can provide a starting point for implementing encoding models.

In addition to the models, the package includes dataset classes that can generate random connectivity matrices from a distribution or load pre-constructed connectivity matrices into `torch_geometric`'s `Data` objects to be passed straight to the model. These objects hold a sparse representation of our connectivity matrices and can be batched together to form isolated subgraphs of a big graph, letting us simulate many networks simultaneously.

Finally, to facilitate the common use pattern of adding an external stimulus to the simulation and recording the resulting activity, we have included various stimulation classes that can be easily added to the model and even tuned to provoke a certain response.

References

Bower, J. M., Cornelis, H., & Beeman, D. (2013). GENESIS, the GEneral NEural SImulation System. In D. Jaeger & R. Jung (Eds.), *Encyclopedia of computational neuroscience* (pp. 1–8). Springer New York. https://doi.org/10.1007/978-1-4614-7320-6_255-1

Bronstein, M. M., Bruna, J., Cohen, T., & Velickovic, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, *abs/2104.13478*. https://arxiv.org/abs/2104.13478

Carnevale, N. T., & Hines, M. L. (2006). *The NEURON book*. Cambridge University Press. https://doi.org/10.1017/CBO9780511541612

Das, A., & Fiete, I. R. (2020). Systematic errors in connectivity inferred from activity in strongly recurrent networks. *Nature Neuroscience*, *23*(10), 1286–1296. https://doi.org/10.1038/s41593-020-0699-2

Eshraghian, J. K., Ward, M., Neftci, E., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D. S., & Lu, W. D. (2023). *Training spiking neural networks using lessons from deep learning*. https://doi.org/10.48550/arXiv.2109.12894

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Gerstner, W. (2008). Spike-response model. *Scholarpedia*, *3*(12), 1343. https://doi.org/10.4249/scholarpedia.1343

Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press. https://doi.org/10.1017/CBO9781107447615

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia*, *2*(4), 1430. https://doi.org/10.4249/scholarpedia.1430

Golosio, B., Tiddia, G., De Luca, C., Pastorelli, E., Simula, F., & Paolucci, P. S. (2021). Fast simulations of highly-connected spiking cortical models using GPUs. *Frontiers in Computational Neuroscience*, *15*. https://doi.org/10.3389/fncom.2021.627620

Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., & Kozma, R. (2018). BindsNET: A machine learning-oriented spiking neural networks library in Python. *Frontiers in Neuroinformatics*, *12*, 89. https://doi.org/10.3389/fninf.2018.00089

Lepperød, M. E., Stöber, T., Hafting, T., Fyhn, M., & Kording, K. P. (2020). Inferring causal connectivity from pairwise recordings and optogenetics. *bioRxiv*. https://doi.org/10.1101/463760

Meyer, A. F., Williamson, R. S., Linden, J. F., & Sahani, M. (2017). Models of neuronal stimulus-response functions: Elaboration, estimation, and evaluation. *Frontiers in Systems Neuroscience*, *10*. https://doi.org/10.3389/fnsys.2016.00109

Paninski, L. (2004). Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, *15*(4), 243–262. https://doi.org/10.1088/0954-898x_15_4_002

Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E. J., & Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, *454*(7207), 995–999. https://doi.org/10.1038/nature07140

Stimberg, M., Brette, R., & Goodman, D. F. (2019). Brian 2, an intuitive and efficient neural simulator. *eLife*, *8*, e47314. https://doi.org/10.7554/eLife.47314

Tiddia, G., Golosio, B., Albers, J., Senk, J., Simula, F., Pronold, J., Fanti, V., Pastorelli, E., Paolucci, P. S., & Albada, S. J. van. (2022). Fast simulation of a multi-area spiking network model of macaque cortex on an MPI-GPU cluster. *Frontiers in Neuroinformatics*, *16*. https://doi.org/10.3389/fninf.2022.883333