Spark: Modular Spiking Neural Networks

An Open Source GPU-based Framework For SNNs

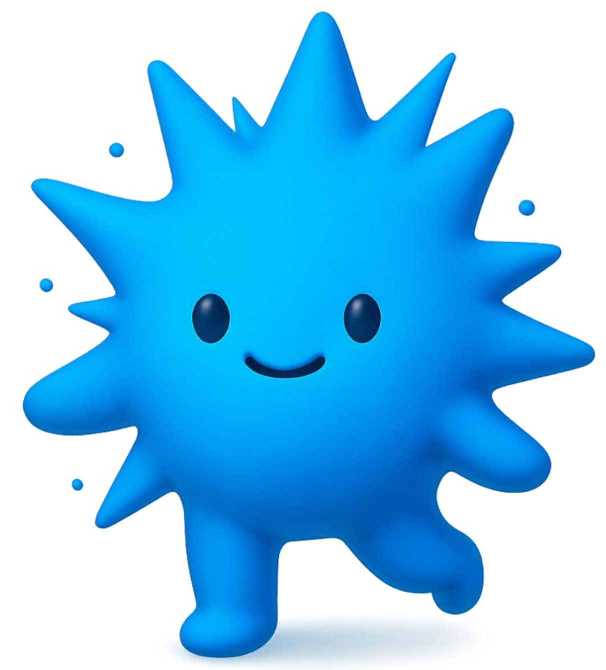
Mario Franco [1*], Carlos Gershenson [1]

[1] School of Systems Science and Industrial Engineering, Binghamton University, USA

*mfrancomndez@binghamton.edu

Paper

Github



High Performance

Powered by JAX

Models may be abstracted and further optimized by a JIT compiler

Modular & Extensible

Modular by design

Models are abstracted as collections of simple and reusable modules

Seamless Workflow

Transform input data and output streams with the built-in IO interfaces

Go seamlessly from one data type to another

Graph Editor

Design complex SNN architectures by dragging, dropping, and connecting pre-built neural components

Abstract

We present a new framework for spiking neural networks --- ***Spark*** --- built upon the idea of modular design, from simple components to entire models. The aim of this framework is to provide an efficient and streamlined pipeline for spiking neural networks.

We hope that a framework compatible with traditional ML pipelines may accelerate research in the area, specifically for continuous and unbatched learning, akin to the one animals exhibit.

Moreover, *Spark* is bundled with several utilities to streamline and simplify the entire pipeline of design, construction and execution SNNs; I/O interfaces, controllers and graphical interfaces, etc.

Design

Compile your great research ideas into a self-contained module

Assamble

Construct a neural model through code or the graph editor

Execute

JIT compile your model and run it as any other ANN framework

Notes

Currently, Spark is the early version of an open source research framework, not a complete end-to-end solution for SNN, developed by a single PhD student out of frustration with the hardships of reproducibility around SNNs.

If you are interested try it and if you like the idea of what it can become do not be afraid to contribute: emails and pull requests are welcome!. What makes any open source project great is the community behind it!.

BINGHAMTON UNIVERSITY | THOMAS J. WATSON COLLEGE OF ENGINEERING AND APPLIED SCIENCE

Cartpole™: A simple study case

We showcase Spark by addressing and solving the classic "hello world" control problem: the Cartpole.

Our approach to the Cartpole problem consists of a reasonable architecture bias, a three-factor modulated plasticity rule and letting the network do what it wants to do (self-organize) with some sporadic feedback.

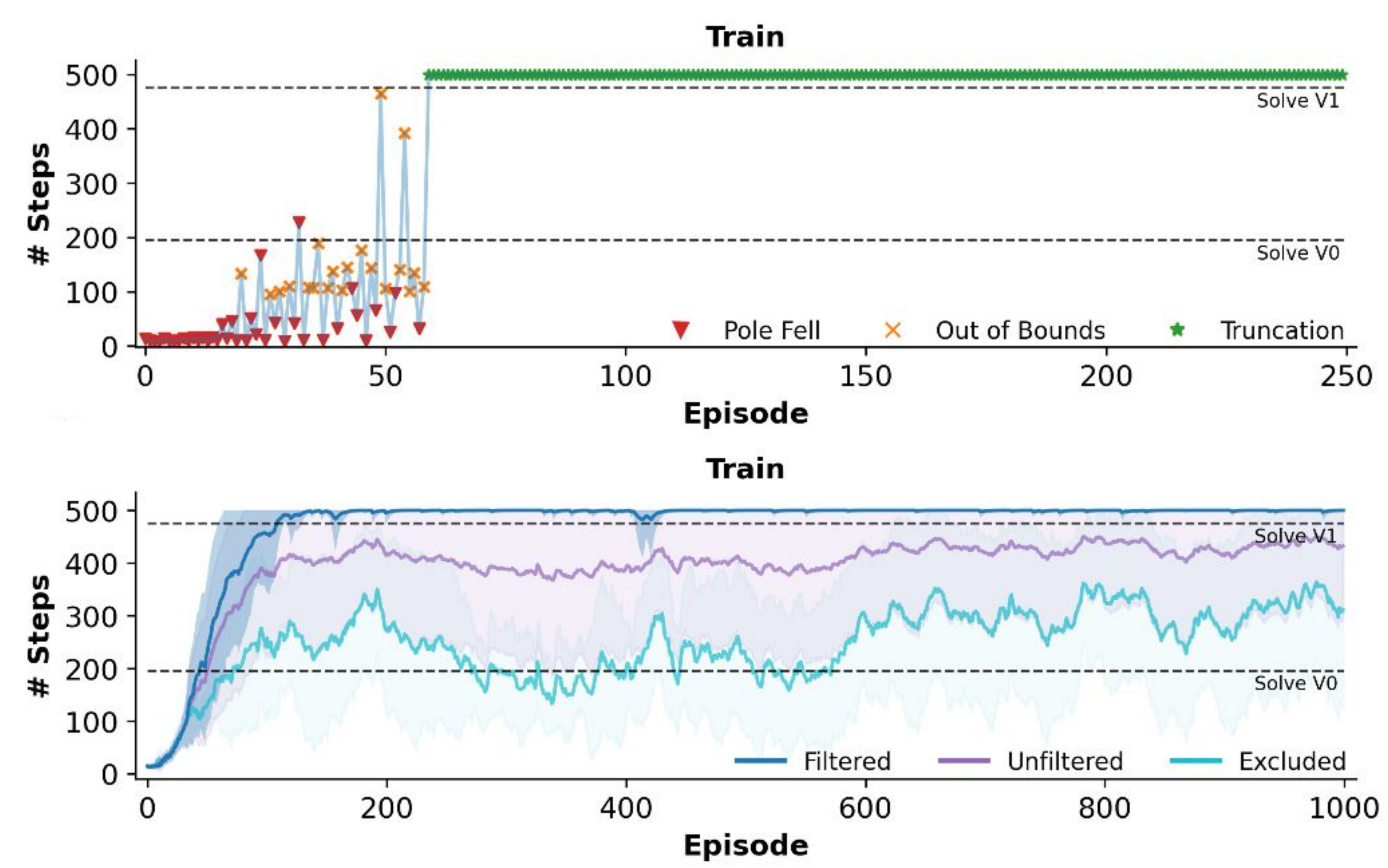


Figure 2. (up) Training trajectories of a single agent, with episode end annotations. (bottom) Average of the exponential moving average (τ = 0.8) of performance of 25 agents. Agents under the filtered category (16/25) were capable of solving the Cartpole problem within ~40-80 episodes. Excluded shows the performance of the agents that failed to stabilize quickly (9/25).

Swift execution is critical to success!

When considering an iterative learning paradigm, performance becomes critical; how much one can compute directly links to how much one can learn.

In order to showcase the capabilities of ***Spark***, we use Brian2 as a reference; Brian2 has comparable or better performance than other similar frameworks. Although Brian2 was designed for accurate simulations, Brian2's C++ compiled models are extremely performant.

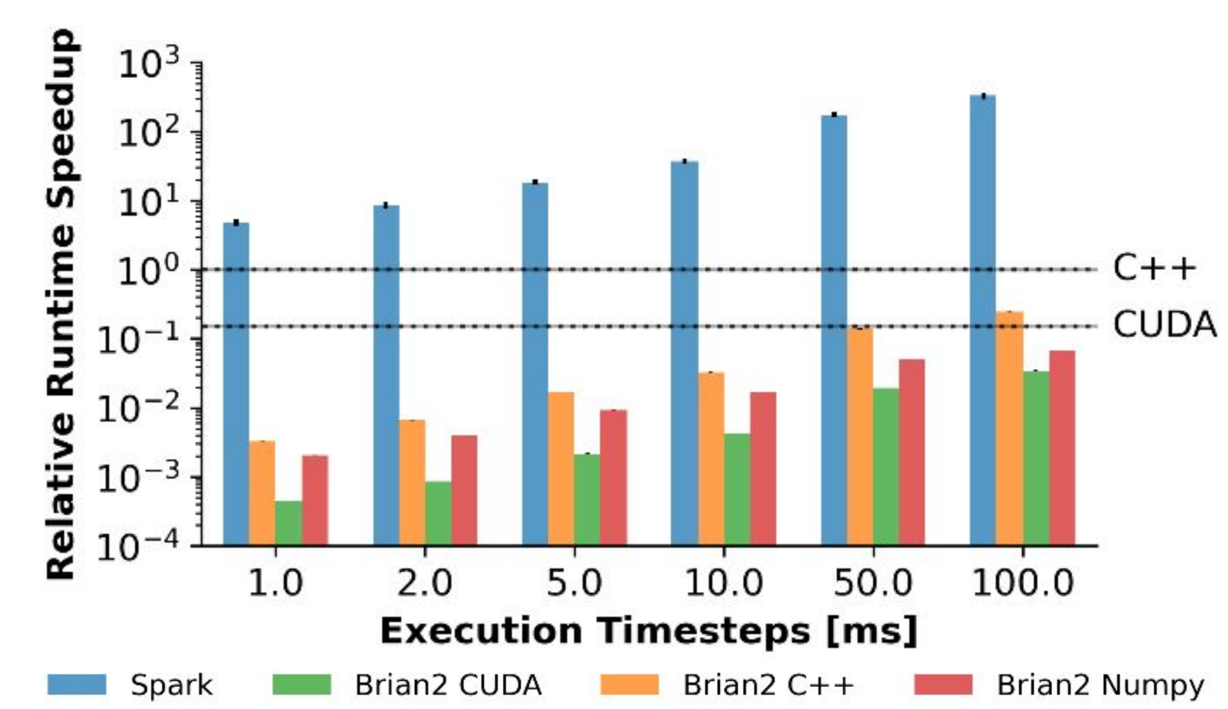


Figure 1. Performance benchmark for a network with 7 neuron pools of 1024 neurons with biased connectivity among pools. Speeds are averaged from 25 different networks and 5 runs per network. Speeds are normalized with respect to an end-to-end Brian2 execution. Execution timesteps indicate how many timesteps each model computes before a new input signal is presented to the network and a new output signal is registered from the network. Dashed lines denote the best-case scenario, i.e., when the network is not interrupted, for Brian2 with the C++ and CUDA backends.

References

1) Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.
2) Ebner, C., Clopath, C., Jedlicka, P., and Cuntz, H. (2019). Unifying long-term plasticity rules for excitatory synapses by modeling dendrites of cortical pyramidal neurons. Cell Reports, 29(13):4295–4307.e6.
3) Fuchsberger, T. and Paulsen, O. (2022). Modulation of hippocampal plasticity in learning and memory. Current Opinion in Neurobiology, 75:102558.
4) Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. (2024). Flax: A neural network library and ecosystem for JAX.
5) Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.
6) Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Processing Magazine, 36(6):51–63.