SpikeTime Spiking Neuronal Network Simulation in Julia Language

Tentative All of the SpikingNeuralNetwork code authors

Dr Russell Jarvis

Yeshwanth Bethi

Pablo de Abreu Urbizagastegui

***Abstract*—**

***Index terms*—A, B, C, D**

I. Introduction

While there is much focus on hardware advances that accellerate the simulation of large scale spiking neural networks, it is worthwhile to shift our attention to language advances that may also support accelerated large scale spiking neural network simulation. Some gains in biologically faithful neuronal network simulation can be achieved by applying recent computer language features. For example, the Julia language supports Sparse Compressed Arrays, Static Arrays, furthermore Julia provides very extensive support for CUDA GPU, as well as a plethora of reduced precision types. Julia also provides a high-level syntax that facilitates high code reuse while simplifying plotting and data analysis. These features lend themselves towards high-performance large-scale Spiking Neural Network simulation. Therefore, we are using Julia to develop an open-source software package that enables the simulation of networks with millions to billions of synapses on a computer with a minimum of 64GB of memory and an NVIDIA GPU.

Another major advantage implementing SNN simulations in the Julia language is reduced technical debt. The simulation code we are developing is both faster and less complicated to read compared with some other simulation frameworks. The simplicity of the code base encompasses a simple installation process. Ease of installation is an important part of neuronal simulators that is often overlooked when evaluating simulation merit, GPU simulation environments are notoriously difficult to install and this big technical burden of installation harms model portability and reproducibility. The Julia language facilitates the ease of installation to solve the "two language problem" of scientific computing. The simulator encompasses a singular language environment, which includes a reliable, versatile, and monolithic package manager. Furthermore the simulator installation includes no external language compilation tools or steps.

To demonstrate the veracity and performance of this new simulation approach, we compare the the Potjans and Diesmann model as implemented in the NEST and GENN simulators. In a pending analysis, we compare simulation execution speeds and spike train raster plots to NEST and GENN using the discussed models as benchmarks.

A review of the literature suggests that there is a desire to modernize pre-existing large scale network simulators, but such efforts fall short of re-writing existing simulator code in the Julia language. [1]

The discussed code repository takes its inspiration from a stale code base: StupidBear SpikingNeuralNetworks.jl

II. Theoretical Framework

Nothing new is presented in terms of theoretical framework.

We use the forward Euler implementations of synaptic current weight updates, and

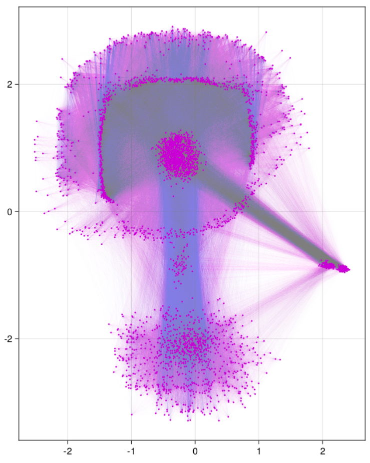
$$V_M \tag{1}$$

updates, as the forward Euler method is fast, and sufficiently robust for use on well known homogenuous Leaky Integrate and Fire neurons.

III. Methodological Framework

Weight update rules

IV. Result Analysis



SGtSNEpi visualization of the Potjans static connectome. Graph partitioning of the connectome adjacency matrix can be used to compile network models in a way that minimises spike traffic between GPU thread locks. Although the SGtSNEpi dimensionalityt reduction technique provides a nice over view of network structure at scale, it is not as fast or useful as other techniques that distribute the network based on effective connectivity measures. different technique we developed which is called Spike2Vec.

Validation of Network Simulation Results

V. Recommendations and Conclusions

1) *Contributions:*

2) *Bibliography:*

References

[1] O. Awile, P. Kumbhar, et al., “Modernizing the neuron simulator for sustainability, portability, and performance,” *Frontiers Neuroinformatics*, vol. 16, 2022.