CARLsim 4: An Open Source Library for Large Scale, Biologically Detailed Spiking Neural Network Simulation using Heterogeneous Clusters

Ting-Shuo Chou*,[2], Hirak J Kashyap*,[1], Jinwei Xing[2], Stanislav Listopad[1], Emily L Rounds[2], Michael Beyeler[1,3,4], Nikil Dutt[1,2], Jeffrey L Krichmar[1,2]

[1]Department of Computer Science
[2]Department of Cognitive Sciences
University of California, Irvine
Irvine, CA, USA 92697

[3]Institute for Neuroengineering
[4]eScience Institute
University of Washington
Seattle, WA, USA 98195

∗ Authors contributed equally to this work
Email: jkrichma@uci.edu

Abstract—**Large-scale spiking neural network (SNN) simulations are challenging to implement, due to the memory and computation required to iteratively process the large set of neural state dynamics and updates. To meet these challenges, we have developed CARLsim 4, a user-friendly SNN library written in C++ that can simulate large biologically detailed neural networks. Improving on the efficiency and scalability of earlier releases, the present release allows for the simulation using multiple GPUs and multiple CPU cores concurrently in a heterogeneous computing cluster. Benchmarking results demonstrate simulation of 8.6 million neurons and 0.48 billion synapses using 4 GPUs and up to 60x speedup for multi-GPU implementations over a single-threaded CPU implementation, making CARLsim 4 well-suited for large-scale SNN models in the presence of real-time constraints. Additionally, the present release adds new features, such as leaky-integrate-and-fire (LIF), 9-parameter Izhikevich, multi-compartment neuron models, and fourth order Runge Kutta integration.**

I. Introduction

Insights from neural representation and organization in the brain play a major role in the design of algorithms to mimic intelligence using modern computers, e.g. models of visual cognition [1] and reinforcement learning [2]. Real neurons communicate using action potentials or spikes. The temporal order and frequency of spikes have been found to be fundamental to learning and memory [3]. In addition, SNNs in the brain are computationally efficient, fault tolerant, and robust to noise perturbations [4]. Therefore, it is important that large scale brain simulations use spiking neurons in order to understand the principles of neural computation. Moreover, neuromorphic hardwares have been developed to implement energy efficient SNNs for low power applications [5].

The existing SNN simulators model neural function at various levels of details. To be useful, the simulators need to support key neurobiological functions as well as be computationally efficient. Biophysically detailed simulators, such as NEURON [6] and GENESIS [7], require a large high-performance computing (HPC) cluster or a supercomputer to simulate only a few thousand neurons [8]. On the other hand, GeNN [9] and NCS [10] simulators achieve parallel execution of large SNNs using low cost off-the-shelf GPUs, however they support limited biological features. Since modern HPC clusters provide arrays of heterogeneous processors, the ability to simulate an SNN distributed across multiple CPUs and GPUs concurrently is highly advantageous.

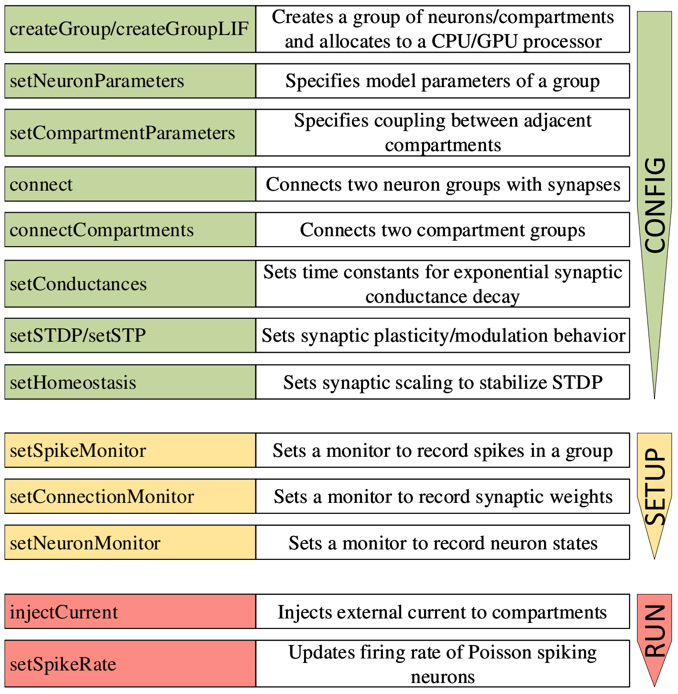
In this paper, we present CARLsim 4, the latest version of the SNN simulation library CARLsim [11], [12], [13], which improves upon the previous versions in terms of its support for multiple GPU and CPU based (hybrid) parallel simulation of large SNNs, multiple compartment models, 9-parameter Izhikevich and LIF spiking neuron models, and the fourth order Runge Kutta sub millisecond integration (RK4) [14] for improved numerical precision. In the current version, CARLsim 4 supports SNN simulations using up to 24 CPU cores and up to 8 GPUs, concurrently. The newly added features in CARLsim 4 greatly improve its usability for large scale brain simulations with biological details. The multi-compartment model allows for large scale biophysical simulations. The 9-parameter Izhikevich model and the RK4 integration are added for precise inter-compartment conductance dynamics. CARLsim 4 includes the LIF neuron, which many modeling studies use due to its simple dynamics. Moreover, LIF neurons are supported by most of the neuromorphic platforms, and CARLsim 4 can be used to develop large SNNs for these energy efficient hardwares.

CARLsim 4 retains all the modeling features and plug-in tools from the previous version. It provides a very simple programming interface and requires only rudimentary knowledge of C++. For easy dissemination and collaboration, CARLsim 4 is available publicly on Github (https://github.com/UCI-CARL/CARLsim4) and the Neuroscience Gateway (https://www.nsgportal.org). Neuroscience Gateway [15] provides access to clusters with state-of-the-art CPU/GPU processors, all of which can be concurrently utilized using CARLsim 4.

II. CARLsim 4 API

CARLsim 4 supports simulations on Linux, macOS, and Windows platforms using C/C++ and NVIDIA CUDA libraries. The POSIX threads library is used to realize parallel execution using multiple CPU cores on Linux and macOS platforms. Windows based simulations additionally require Visual Studio and openMP libraries. Instructions to set up CARLsim 4 are available on the user guide (http://uci-carl.github.io/CARLsim4/). All third party packages required by CARLsim 4 in all supported platforms are freely available.

A. Simulation workflow



createGroup/createGroupLIF	Creates a group of neurons/compartments and allocates to a CPU/GPU processor	CONFIG
setNeuronParameters	Specifies model parameters of a group	CONFIG
setCompartmentParameters	Specifies coupling between adjacent compartments	CONFIG
connect	Connects two neuron groups with synapses	CONFIG
connectCompartments	Connects two compartment groups	CONFIG
setConductances	Sets time constants for exponential synaptic conductance decay	CONFIG
setSTDP/setSTP	Sets synaptic plasticity/modulation behavior	CONFIG
setHomeostasis	Sets synaptic scaling to stabilize STDP	CONFIG
setSpikeMonitor	Sets a monitor to record spikes in a group	SETUP
setConnectionMonitor	Sets a monitor to record synaptic weights	SETUP
setNeuronMonitor	Sets a monitor to record neuron states	SETUP
injectCurrent	Injects external current to compartments	RUN
setSpikeRate	Updates firing rate of Poisson spiking neurons	RUN

Fig. 1: The sequence of states in a typical CARLsim 4 simulation and the API calls invoked in them. Green, yellow, and red boxes denote CONFIG, SETUP, and RUN states, respectively.

A CARLsim simulation process occupies CONFIG, SETUP, and RUN states during execution. In CONFIG state, the model architecture and dynamics are configured. As shown in Figure 1, the main operations performed in this state are creation of neuron or compartment groups, creation of synaptic connections among neuron groups and dendritic connections among compartment groups, and configuration of conductance decay and plasticity behaviors of the synapses. Moreover, the preferred CPU/GPU processor of each group is defined in this state. A call to the `setupNetwork` API transitions the simulation to SETUP state, in which data structures are generated internally for storing model parameters, and monitors are set to record parameters and activities, such as spikes (in address-event representation), synaptic weights, and neuron states. The constructed network is simulated for the desired amount of time steps by calling `runNetwork`, and the simulation enters RUN state. In this state, `runNetwork` can be called iteratively with desired input spike rates and/or external currents that can be altered during runtime. The API calls listed in Figure 1 are documented in the user guide and partly in [13].

B. Neuron types

CARLsim 4 provides a built-in implementation of spiking point neuron models as well as multi-compartment models. The LIF, 4-parameter Izhikevich [16], and 9-parameter Izhikevich [17] spiking neuron models are supported. These point neurons can be stimulated by external current or through synaptic input from other neurons. The 4-parameter Izhikevich neuron model is capable of producing a large set of spiking behavior observed in cortical neurons and is computationally highly efficient in regard to the more biologically accurate Hodgkin–Huxley model [18] with similar firing capabilities [16].

The 9-parameter Izhikevich model [17] was introduced to capture neuronal dynamics that the 4-parameter model could not and for better interpretability. The model is given by the following equations and properties:

$$C\frac{dv}{dt} = k(v - v_r)(v - v_t) - u + I, \tag{1}$$

$$\frac{du}{dt} = a\{b(v - v_r) - u\}, \tag{2}$$

$$if \quad v \geq v_{peak} \quad v \leftarrow c \quad and \quad u \leftarrow u + d,$$

where, v is the membrane potential, u is the recovery variable, and I is the cumulative stimulus from synaptic excitatory/inhibitory inputs, external current, and noise. The nine parameters of the model are the membrane capacitance C, the resting membrane potential v_r, the instantaneous threshold potential v_t, the rate constant of the membrane potential k, the recovery time constant a, a constant b, the reset potential c, the outward minus inward current during spike d, and the threshold potential for spike v_{peak}.

```
CARLsim sim("hello_world", GPU_MODE, USER);
// Create a 9 parameter Izhikevich neuron group
int grp1=sim.createGroup("Group1", Grid3D(3,3,1),
      INHIBITORY_NEURON, 0, GPU_CORES);
sim.setNeuronParameters(grp1, izh_C, izh_k, izh_vr,
      izh_vt, izh_a, izh_b, izh_vpeak, izh_c, izh_d);

// Create an LIF neuron group
int grp2=sim.createGroupLIF("Group2", Grid3D(13,9,1),
      EXCITATORY_NEURON, 1, CPU_CORES);
sim.setNeuronParametersLIF(grp2, tau_m, tau_ref, vTh,
      vReset, RangeRmem(minRmem, maxRmem));
```

The above code snippet demonstrates the creation of a 9-parameter Izhikevich neuron group and an LIF neuron group. The first group of neurons (`Group1`) is placed on a 3×3 grid and they are inhibitory, i.e. these neuron have only GABAergic synapses. Neurons with all glutamatergic synapses are specified by EXCITATORY_NEURON keyword. The final two arguments of the `createGroup` call specify that the preferred placement of the group is the GPU with ID 0 (0-indexed), The integer value (`grp1`) returned by the `createGroup` method serves as the ID for the neuron group to be used later by other API calls. The `setNeuronParameters` call sets the nine parameters of Izhikevich model. The second group of LIF neurons (`Group2`) is arranged on a 13×9 grid and are placed on CPU core 1 using `createGroupLIF` call. The `setNeuronParametersLIF` method sets up LIF parameters for the whole group, whereas the `RangeRmem` struct can be used to set a fixed or uniformly distributed membrane resistance values across the neuron population.

Neurons can be modeled with multiple compartments in a dendritic tree. In a dendritic tree, currents flow into a compartment from the up (away from soma) and down (toward soma) compartments. Figure 2 depicts a dendritic tree and the flow of dendritic currents into a compartment i.

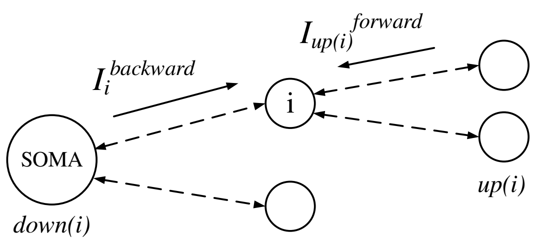


Fig. 2: An example dendritic tree with multiple compartments. The dashed arrows represent connections between the compartments. The forward and backward dendritic currents entering compartment i are depicted using solid directional arrows.

Each compartment can have multiple up-compartments and a single down-compartment in the tree. We term the dendritic current flowing toward the soma as forward current and the dendritic current flowing away from soma as backward current. For a compartment i and its down-compartment $down(i)$, the forward current from compartment i to compartment $down(i)$ is denoted as ($I_i^{forward}$) and the backward current from compartment $down(i)$ to compartment i is denoted as ($I_i^{backward}$). The total dendritic current that flows into a compartment i is given by the following equation.

$$I_i^{dendr} = I_i^{backward} + \sum_{j \in up(i)} I_j^{forward} \tag{3}$$

The asymmetric coupling strength between two compartments determines the directional flow of dendritic current. The coupling strength value can expressed in terms of conductance (G) and the degree of asymmetry in the coupling (P). A value of $P = 0.5$ signifies symmetric coupling. Between compartments i and $down(i)$, the forward current $I_i^{forward}$ and the backward current $I_i^{backward}$ are given by the following equations,

$$I_i^{forward} = G_i * P_i * (V_i - V_{down(i)}) \tag{4}$$

$$I_i^{backward} = G_i * (1 - P_i) * (V_i - V_{down(i)}) \tag{5}$$

The $G_i * P_i$ term in equation 4 is called the up-coupling constant for compartment $down(i)$ and the $G_i * (1 - P_i)$ term in equation 5 is called the down-coupling constant for compartment i.

```
// Create a soma and a dendritic compartment
int grpSP = sim.createGroup("Soma", 20,
      EXCITATORY_NEURON, 1, GPU_CORES);
sim.setNeuronParameters(grpSP, izh_a, izh_b,
      izh_c, izh_d);
int grpD1 = sim.createGroup("Dendrite1", 20,
      EXCITATORY_NEURON, 1, GPU_CORES);
sim.setNeuronParameters(grpD1, izh_a, izh_b,
      izh_c, izh_d);
sim.setCompartmentParameters(grpSP, 4.60f, 0.0f);
sim.setCompartmentParameters(grpD1, 0.0f, 28.39f);
sim.connectCompartments(grpSP, grpD1);
```

Each compartment is modeled using a point neuron model. In the above example, a soma compartment group (`grpSP`) and a dedritic compartment group (`grpD1`) are created using the 4-parameter Izhikevich neuron model. Coupling constants of each compartment group are set using the API call `setCompartmentParameters(grpId, gUp, gDown)`. This method indicates that the group is modeling compartments, instead of point neurons. The `connectCompartments` method connects two compartments bidirectionally.

C. Synapse types

CARLsim provides two types of synapse models, the current based model (CUBA) and the conductance based model (COBA). CUBA propagates action potential from the pre-synaptic neuron, multiplied by synapse weights, as synaptic current. In COBA mode, synaptic current is calculated using conductance variables of postsynaptic receptors. Exponentially decayed conductances of multiple synaptic-receptor types are used. For excitatory synapses $AMPA$ and $NMDA$ receptors are used, and for inhibitory synapses, $GABA_A$ and $GABA_B$ receptors are used. For each receptor k, the conductance (g) at time t is given by the following equation.

$$g = \sum_f \theta(t - t_f) e^{\frac{t - t_f}{\tau}} \tag{6}$$

In this equation, f is a pre-synaptic spike, t_f is the time of its occurance, θ is the Heaviside function, and τ is the decay time constant. The synaptic current flowing through each of the receptors is also a function of postsynaptic voltage and receptor specific reversal potential [13].

D. Plasticity types

Multiple forms of synaptic plasticity and modulation are supported, namely, short-term plasticity (STP) [19], spike timing dependent plasticity (STDP) [3], dopamine modulated STDP [20], and homeostatic synaptic scaling [21]. STP acts on a timescale of the order 100 ms and results in short-term facilitation (STF) and short-term depression (STD). STF happens due to influx of calcium into the axon terminal after spike and STD is caused by consumption of neurotransmitters for synaptic signaling at the axon terminal of pre-synaptic neuron. CARLsim implements the phenomenological model of STP [22].

Excitatory STDP and Inhibitory STDP are used based on whether the pre-synaptic neuron is excitatory or inhibitory. The weight change policy reverses for these two types. The nearest neighbor STDP rule [23] is implemented. CARLsim supports modulation of STDP using the dopamine neuromodulator, by allowing STDP to take place only in elevated dopamine concentrations. Additionally, homeostatic synaptic scaling is provided to stabilize STDP. The scaling mechanism is multiplicative in nature and preserves the relative weights of the synapses connecting a neuron. Plasticity features of CARLsim were discussed in details in [13].

E. Integration methods

CARLsim 4 supports sub millisecond integration using first order forward-Euler and RK4 methods [14]. A common integration method is used for all the neurons in the network. Although, irrespective of the integration time step length, the simulation time step in CARLsim 4 is always 1 millisecond, i.e. spikes can occur at 1 millisecond time resolution. It is recommended that a minimum of two forward-Euler integrations are performed per millisecond for the 4 parameter Izhikevich neuron for precision. RK4 with at least 10 integrations per millisecond should be used for the 9-parameter Izhikevich neurons and the multi-compartment neurons. Due to its simplicity, the integration of Izhikevich recovery variable is always performed using the forward-Euler method. Higher order integration and more integrations per millisecond require more processing time.

F. Utilities

CARLsim provides a set of library tools to support modeling. A built-in Parameter Tuning Interface (PTI) [24] is provided to tune parameters and hyper-parameters of a network using an evolutionary computing library [25]. This tool has been used to produce simple cell dynamics in the primary visual cortex with self-organizing receptive fields [24] and to tune neural responses to match behavior of the rat retrosplenial cortex [26]. A MATLAB visual stimulus toolbox is provided that can be used to generate 2D visual patterns to feed into a CARLsim network through a C++ plugin. Similarly, another MATLAB toolbox is provided for offline analysis and plotting of network activities from a simulation. Moreover, a regression suite and a continuous integration plug-in is provided for seamless integration of new features to the library.

III. Example models using CARLsim

A. Existing models

CARLsim has been used to model a broad range of biologically plausible SNNs [13]. These include, models of primate visual cortex, rat retrosplenial cortex, motor planning in posterior parietal cortex, dopamine modulation in insular cortex, synaptic plasticity, working memory, and attention. These models have been able to reproduce data observed in neurobiology and psychophysics literature. More importantly, CARLsim models with up to 100K neurons were run in real time using GPU. One such model of dorsal visual pathway [27] was used to visually navigate a robot in real environment using live camera data.

B. Examples of new CARLsim 4 functionalities

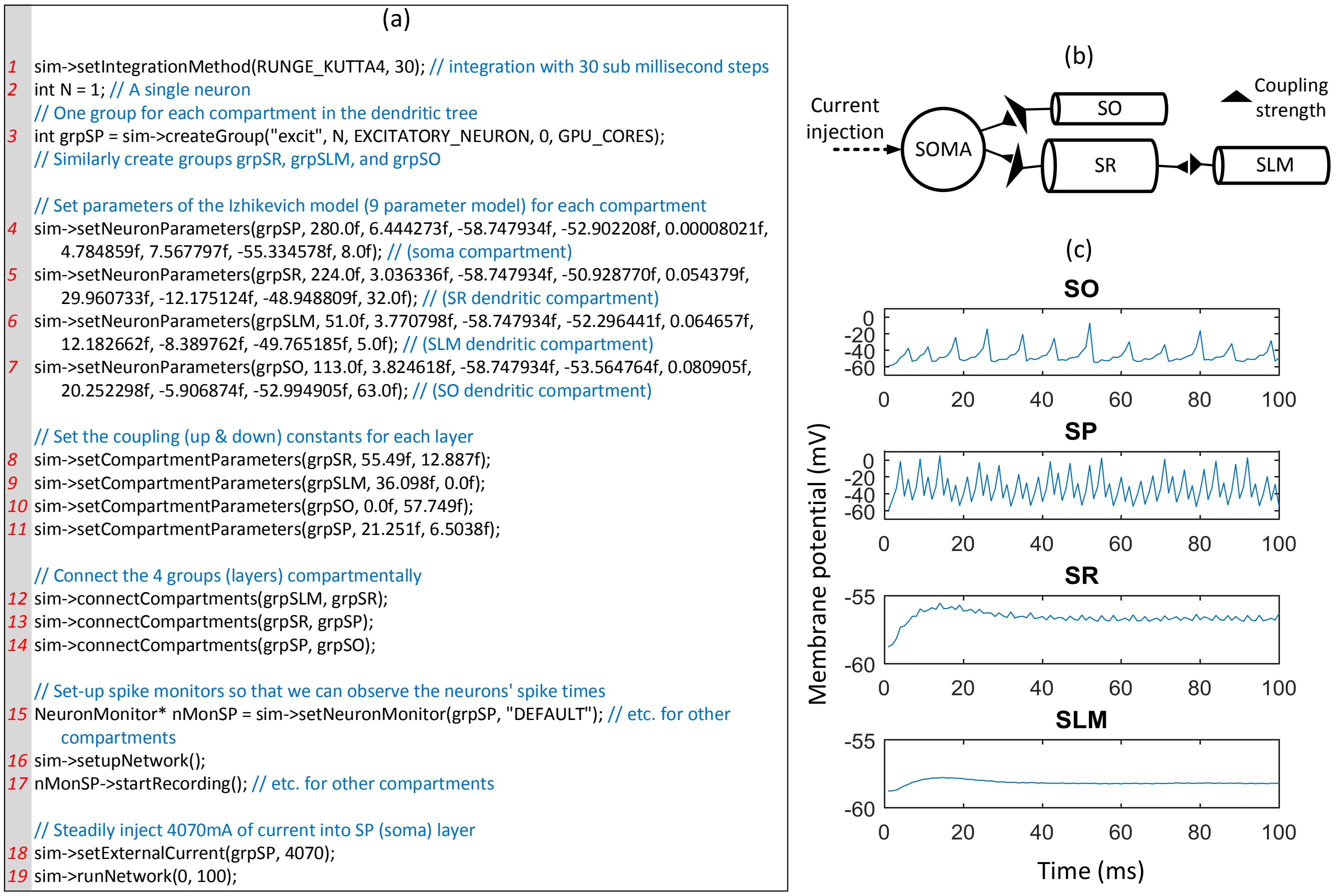
1) A fast spiking CA3 Basket cell with four compartments: The multi-compartment neuron feature is demonstrated using a four compartment model of a fast spiking basket cell found in CA3 region of hippocampus. The cell is a type of perisomatic inhibitory interneurons that contribute to the widely studied gamma frequency oscillations in hippocampal slices [28]. Although these neurons contain numerous dendritic branches, recording studies usually decompose these neurons into four slices or compartments for investigation of local field potentials and spiking activities, termed as SO, SP (soma), SR, and SLM [28]. Figure 3 (b) depicts the dendritic structure of the cell. The example demonstrates an experiment to observe the membrane potential dynamics of each compartment, when a steady external current is applied to SP.

Figure 3 (a) shows the C++ source code for simulating a CA3 basket cell with four compartments. Each compartment is created using the 9-parameter Izhikevich neuron model (line 3). The model parameters of each compartment are chosen to reflect the geometry of the neuron in each layer (lines 4-7). The coupling strength values between compartments are chosen to match inter-layer polarizing behaviors observed in recording studies [28] (lines 8-11). Neuron monitors are set to track membrane potential values of each compartment during simulation (line 15).

The neuron is simulated for 100 ms with steady injection of 4070 mA external current to the soma (lines 18-19). External and dendritic currents are integrated 30 times per millisecond using RK4. The membrane potential dynamics of each compartment during the simulation are depicted in Figure 3 (c). Depolarization of the soma causes strong spiking activities in the SP compartment, which are propagated to the SO compartment due to strong coupling from SP to SO, which results in SO spikes. However, the coupling from SP to SR is not strong and depolarization in SR and SLM compartments are not substantial.

Currently, all connected compartment groups must be allocated to the same processor. In this example, all four compartments are placed on the same GPU (GPU 0), since they are all connected in a dendritic chain.

2) An 80-20 network with Izhikevich and LIF neurons implemented using heterogeneous processors: We implement the random spiking 80-20 network containing 80% excitatory and 20% inhibitory neurons [29] using 2 GPUs and 2 CPUs. The network exhibits sleep-like oscillations observed in the mammalian cortex. The ratio of number of excitatory neurons



```
sim->setIntegrationMethod(RUNGE_KUTTA4, 30); // integration with 30 sub millisecond steps
int N = 1; // A single neuron
// One group for each compartment in the dendritic tree
int grpSP = sim->createGroup("excit", N, EXCITATORY_NEURON, 0, GPU_CORES);
// Similarly create groups grpSR, grpSLM, and grpSO

// Set parameters of the Izhikevich model (9 parameter model) for each compartment
sim->setNeuronParameters(grpSP, 280.0f, 6.444273f, -58.747934f, -52.902208f, 0.00008021f,
    4.784859f, 7.567797f, -55.334578f, 8.0f); // (soma compartment)
sim->setNeuronParameters(grpSR, 224.0f, 3.036336f, -58.747934f, -50.928770f, 0.054379f,
    29.960733f, -12.175124f, -48.948809f, 32.0f); // (SR dendritic compartment)
sim->setNeuronParameters(grpSLM, 51.0f, 3.770798f, -58.747934f, -52.296441f, 0.064657f,
    12.182662f, -8.389762f, -49.765185f, 5.0f); // (SLM dendritic compartment)
sim->setNeuronParameters(grpSO, 113.0f, 3.824618f, -58.747934f, -53.564764f, 0.080905f,
    20.252298f, -5.906874f, -52.994905f, 63.0f); // (SO dendritic compartment)

// Set the coupling (up & down) constants for each layer
sim->setCompartmentParameters(grpSR, 55.49f, 12.887f);
sim->setCompartmentParameters(grpSLM, 36.098f, 0.0f);
sim->setCompartmentParameters(grpSO, 0.0f, 57.749f);
sim->setCompartmentParameters(grpSP, 21.251f, 6.5038f);

// Connect the 4 groups (layers) compartmentally
sim->connectCompartments(grpSLM, grpSR);
sim->connectCompartments(grpSR, grpSP);
sim->connectCompartments(grpSP, grpSO);

// Set-up spike monitors so that we can observe the neurons' spike times
NeuronMonitor* nMonSP = sim->setNeuronMonitor(grpSP, "DEFAULT"); // etc. for other
    compartments
sim->setupNetwork();
nMonSP->startRecording(); // etc. for other compartments

// Steadily inject 4070mA of current into SP (soma) layer
sim->setExternalCurrent(grpSP, 4070);
sim->runNetwork(0, 100);
```

Fig. 3: Multi-compartment experiment of a CA3 basket cell with four slices (compartments). (a) CARLsim 4 source code to implement the experiment, (b) Coupling among the four compartments of the neuron; the coupling strength between compartments is unidirectional and is proportional to conductance, (c) the membrane potential values of each of the compartments during the experiment.

to the number of inhibitory neurons in the SNN preserves the ratio found in the mammalian cortex [30]. Similarly, the inhibitory synapses are stronger than the excitatory synapses. The excitatory neurons receive inputs from both excitatory and inhibitory neurons, whereas the inhibitory neurons receive inputs from only excitatory neurons. All neurons receive inputs from approximately 100 other neurons. Additionally, at each time step, a randomly selected neuron receives a thalamic input.

Since we use a total of four processors to simulate the SNN, we can better utilize all available CPU/GPU processors by dividing the excitatory group into three subgroups of equal size (i.e. each excitatory subgroup contains 26.67% of the neurons in the whole network). Therefore, we subdivide the original 80-20 network [29] to demonstrate a CARLsim 4 simulation with heterogeneous neuron models using heterogeneous computing processors, as shown in Figure 4(a). Each neuron group is placed in a dedicated processor and connections between the groups are made to mimic the 80-20 network architecture.

The excitatory groups are modeled using LIF neurons and the inhibitory group is modeled using fast spiking 4-parameter Izhikevich neurons [16]. We tune the LIF neuron parameters to obtain regular spiking behavior. Fig. 4(b) depicts the rhythmic oscillations in spiking activity of the inhibitory group and one excitatory subgroup, with stronger activities in the inhibitory group. The C++ source code used to implement this experiment is available in the projects directory.

IV. Kernel: Improvements and Benchmark

A. *Kernel Improvements*

In order to achieve SNN simulations using multiple CPU Cores and GPUs concurrently, we re-wrote the kernel to support: 1) a common runtime data structure that is used by different computing modules (currently CPU and GPU backend), 2) a standard interface to CPU and GPU computing modules, 3) a basic network partition algorithm with a layer managing mappings between the global network and subnetworks, and 4) a universal spike routing function among different computing modules.

Fig. 5 illustrates the improvement in memory management over the previous release. In CARLsim 3, the kernel could manage only one copy of runtime data (i.e., voltage v(t), current i(t), and spikes s(t) for each neuron) and calculated the results using either CPU or GPU computing functions. In CARLsim 4, the kernel can first partition runtime data into several subsets of the original runtime data and then instantiate

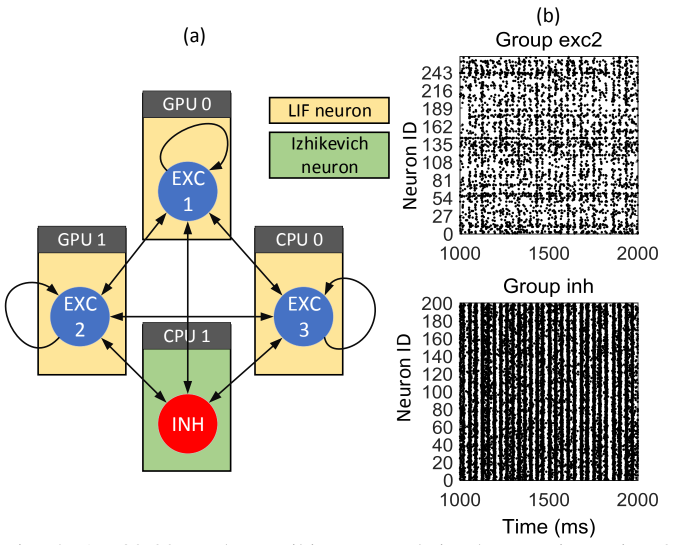


Fig. 4: An 80-20 random spiking network implementation using 2 CPUs and 2 GPUs. (a) Architecture of the 80-20 network used in this example. The blue colored circles are excitatory neuron groups (EXC1, EXC2, EXC3) and the red colored circle is the inhibitory neuron group (INH). Incoming synapses to the inhibitory group are fixed. All other synapses are updated using STDP. The neurons inside the yellow rectangles follow LIF dynamics and the neurons inside the green rectangle follow 4-parameter Izhikevich dynamics. (b) Spike monitor outputs for the four EXC2 and INH groups.

multiple CPU and/or GPU modules to calculate the results. Currently, the kernel of CARLsim 4 can manage 8 copies of partitioned runtime data on GPU memory and 24 copies on main memory, which means the support of 8 GPU cards and CPU with 24 cores.

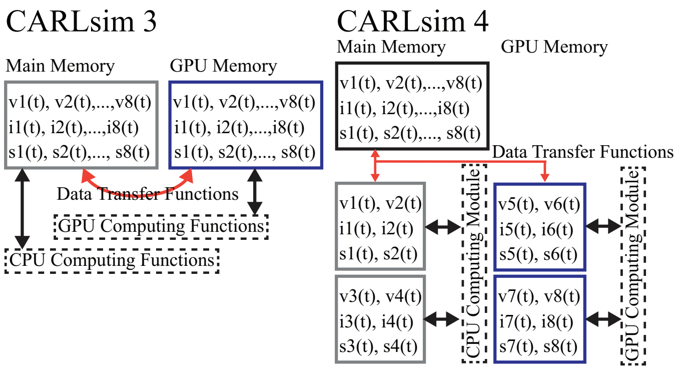


Fig. 5: Memory organization

B. Benchmark

We designed benchmarks for multi-CPU acceleration, multi-GPU acceleration, scalability, to show CARLsim 4's improvements over CARLsim 3. We ran all benchmarks on a HPC node with 4 NVIDIA Tesla K80 GPUs with 10 GB DRAM per GPU, an Intel Xeon E5-2680v3 processor containing 24 CPU cores, and 128 GB CPU DRAM.

1) Benchmark 1 - Multi-GPU and Multi-GPU acceleration: In this benchmark, we evaluate the acceleration achieved by multiple CPU cores (we will refer to as CPUs) and multiple GPUs. The network contains 8 subnetworks with 100 synapses per neuron. Each subnetwork consists of 80% excitatory (Group E) and 20% inhibitory (Group I) neurons. An additional Poisson spike generator group (Group P) of equal size as the excitatory group is included to drive subnetwork activity. In each subnetwork, 2 out of 4 connections are between neuron groups allocated to different CPU or GPU. The subnetworks are evenly partitioned among 1, 2, 4 GPUs or 1, 2, 4, 8 CPUs in separate runs, as shown in Figure 6. In all cases, subnetwork allocation are done to have equal number of neurons on each partition. We vary the number of E+I neurons in each 80-20 subnetwork from 10^3 to 10^5, i.e. total number of neurons in the whole network varies from 1.44×10^4 to 1.44×10^6. All excitatory synapse weights are updated using STDP. The mean firing rate of the network is approximately 16 Hz in all cases.

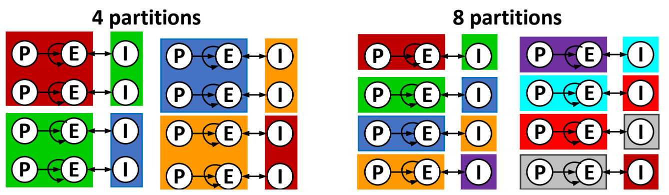


Fig. 6: The network allocation scheme used for evaluation of multiple CPU/GPU acceleration. Each color denotes a unique CPU or GPU partition. The 8 partitions allocation is done for only CPU case. 1 and 2 partitions allocation schemes are not shown. For 2, 4, and 8 partitions cases, 50% connections are between neuron groups on different CPU or GPU.

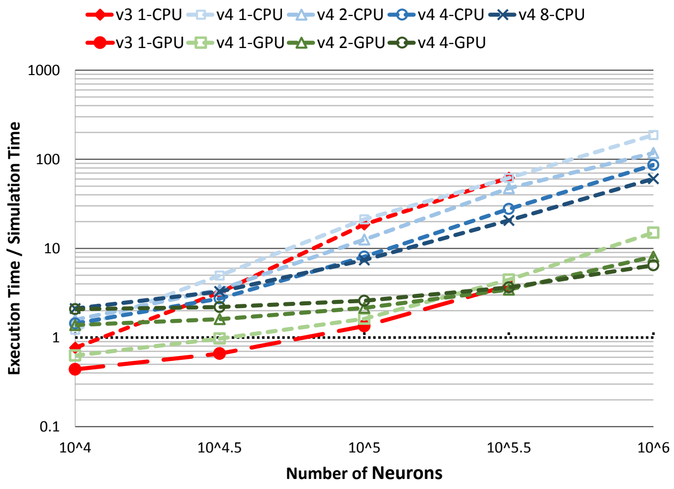


Fig. 7: Execution time per second of simulation versus number of neurons and number of CPUs or GPUs used. The vertical axis is shown in logscale, in which a value of 1 means real time performance. v3 and v4 refer to CARLsim 3 and CARLsim 4, respectively.

Fig. 7 shows that multiple CPUs and multiple GPUs accelerate simulation of large SNNs. We compare CARLsim 4 against CARLsim 3, which is able to run up to order of $10^{5.5}$ neurons using one CPU or GPU. It can be seen that for smaller networks, fewer partitions (also CARLsim 3 with 1 partition) run faster. This is expected because for smaller networks, inter hardware communications throttle any acceleration due to parallel execution. However, as the network size increases, CARLsim 4 outperformes CARLsim 3 in terms of both speed and maximum network size. When comparing GPU against

CPU in terms of execution time, GPU simulation yields 10x to 60x speed up over CPU.

2) Benchmark 2 - Scalability: To estimate the largest network that can be realized by CARLsim 4 using the hardware at our disposal as a reference, we scale the number of neurons to maximum possible with the 4-partitions allocation scheme shown in Figure 6 using 4 GPUs. The synapse weights are tuned to achieve a mean firing rate of approximately 20 Hz and held fixed. The total size of the largest network realized is 4.3×10^6 neurons and approximately 2.4×10^8 synapses. The memory footprint is 9.2 GB DRAM per GPU, which is the major limiting factor on network size. However, when we change the allocation scheme to assign one subnetwork per GPU, without any cross GPU synapses, the largest network realized is 8.6×10^6 neurons and 4.8×10^8 synapses, with 9.02 GB memory footprint per GPU DRAM.

TABLE I: Comparison of spiking neural network simulators regarding supported features. An 'X' denotes that the feature is supported by the simulator, a '/' denotes that the feature is partially implemented or requires substantial effort to use, and a blank ' ' denotes that the feature is not supported. Gray colored rows denote new features added in CARLsim 4 and not available in previous versions.

	CARLsim 4	Brian 2	NEURON 7.5	GeNN 3	NCS 6	Nemo 0.7	Nengo 2.6	NEST 2.14	PCSIM 0.5
Neuron model									
Leaky Integrate-and-fire	X	X	X	/	X		X	X	X
Izhikevich 4-param	X	X		X	X	X	X	X	X
Izhikevich 9-param	X	X		/					
Multi-compartment	X	X	X					X	
Hodgkin-Huxley		X	X	X	X		X	X	X
Synapse Model									
Current-based	X	X	X	X	X		X	X	X
Conductance-based	X	X	X	X	X	X	X	X	X
AMPA, NMDA, GABA	X	X	X	X	/		X	X	X
Neuromodulation	/	/	X	/		/	X	/	/
Synaptic Plasticity									
Short-term plasticity	X	X	X	/	X	X	X	X	X
E-STDP	X	X	X	X	X	X	/	X	X
I-STDP	X	/	X	/		X		X	
DA-STDP	X	/	X	/			/	X	X
Homeostasis	X	/	X	/			X	X	X
Tools									
Parameter tuning	X	/					X		
Analysis/visualization	X	/	X		/	X	X		/
Regression suite	X	X		X		X	X	X	X
Integration methods									
First-order/exponential	X	X	X	X	X	X		X	X
Exact/Crank-Nicholson		X	X					X	
Runge-Kutta	X	X					X	X	
Computing hardware									
Single-threaded CPU	X	X	X	X	X	X	X	X	X
Multi-threaded CPU	X	X	X		X	X	X	X	X
Distributed		/	X		X		X	X	X
Single GPU	X	/		X	X	X	X		
Multi-GPU	X				X				
Hybrid (Multi-CPU/GPU)	X				X				

V. Related works

We compare CARLsim 4 with existing large-scale SNN simulators that satisfy these criteria: (i) open source, (ii) support parallel simulation of spiking point neurons in a network, (iii) support conductance based synapses, and (iv) support synaptic plasticity. Accordingly, we select Brian 2 [31], NEURON 7.5 [6], GeNN 3 [9], NCS 6 [10], Nemo 0.7 [32], Nengo 2.6 [33], NEST 2.14 [34], and PCSIM 0.5 [35]. Table I presents a side-by-side comparison of these simulators in terms of supported features, which is a compiled from the documentations, articles, source code, and release notes of the simulators. Table I obviously does not compare across all the features supported by the simulators. However, the features considered herein are the ones desired for modeling of large-scale biologically plausible SNNs [36].

Compared to other simulators, CARLsim balances between flexibility and performance by providing optimized CUDA/C++ implementations of a large number of biologically plausible model features. Moreover, its modularity allows users to add a new feature with minimal effort. CARLsim provides neuron, synapse, and plasticity models that are highly relevant for simulating brain regions as well as for theoretical understanding of network level coding principles.

All of the considered simulators support some type of spiking point neuron models. However, multi-compartment models provide a way to model realistic membrane potential dynamics in dendrite branches and cell body. Morphologically realistic neuron simulators, such as NEURON [6] and Genesis [7], can be used to model multiple compartment models with desired geometry. Other simulators have added multi-compartment support with less concern about precise geometry and more about modeling the interaction among the compartments. CARLsim provides built-in implementation of multi-compartment models with GPU acceleration. CARLsim provides built-in methods to implement dopaminergic neuromodulation.

CARLsim is among the few simulators to provide an integrated automatic parameter tuning tool for spiking neural networks. CARLsim finds open parameters of an SNN that best fit a given response behavior. The parameters to fit can be diverse, such as synaptic weights, neuron dynamics, or network topologies.

Modern high-performance computing clusters and laboratory workstations contain arrays of CPUs and GPUs. To support these heterogeneous clusters, CARLsim 4 has been improved to support parallel network simulations using multiple GPUs and CPUs concurrently. To the best of our knowledge, NCS and CARLsim 4 are the only available SNN simulators to support simulations using heterogeneous CPU/GPU processors. NEST and PCSIM support traditional distributed computing using CPU based clusters. Brian, GeNN, Nemo, and Nengo allow parallel network simulations using GPUs.

VI. Conclusion

For large-scale simulation and analysis of biologically realistic neural networks using off-the-shelf computing resources, CARLsim has advantages over other simulators. In particular, CARLsim's support for parallel simulations using multiple GPUs and CPUs, built-in biologically realistic neuron, synapse, and learning models, minimal third-party dependencies, compatibility with popular OS platforms, integrated visualization and analysis tools, rigorous documentation of code and features (including tutorials), and continuous integration and testing, make it an easy to use and powerful simulator of biologically plausible neural network models. Moreover, the automated parameter tuning interface integrated onto CARLsim simplifies design of large networks to mimic complex brain functions. Finally, CARLsim provides an easy to use programming interface, that has been used by researchers with diverse expertise.

Acknowledgment

This research was supported by National Science Foundation Award IIS-1302125, Intel Corporation, and Northrop Grumman Aerospace Systems. MB was supported by Washington Research Foundation Funds for Innovation in Neuroengineering and Data-Intensive Discovery. We thank Amitava Majumdar of San Diego Supercomputing Center for providing access to their HPC resources. We also acknowledge the contributions to CARLsim 4 by Andrey Krainyak.

References

[1] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] Y. Dan and M.-m. Poo, "Spike timing-dependent plasticity of neural circuits," *Neuron*, vol. 44, no. 1, pp. 23–30, 2004.

[4] W. Maass, "On the computational power of noisy spiking neurons," in *Advances in neural information processing systems*, 1996, pp. 211–217.

[5] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[6] M. L. Hines and N. T. Carnevale, "The NEURON simulation environment," *NEURON*, vol. 9, no. 6, 2006.

[7] J. M. Bower, D. Beeman, and M. Hucka, "The GENESIS simulation system," 2003.

[8] H. Markram, E. Muller, S. Ramaswamy, M. W. Reimann, M. Abdellah, C. A. Sanchez, A. Ailamaki, L. Alonso-Nanclares, N. Antille, S. Arsever *et al.*, "Reconstruction and simulation of neocortical microcircuitry," *Cell*, vol. 163, no. 2, pp. 456–492, 2015.

[9] E. Yavuz, J. Turner, and T. Nowotny, "GeNN: a code generation framework for accelerated brain simulations," *Scientific reports*, vol. 6, 2016.

[10] R. V. Hoang, D. Tanna, L. C. J. Bray, S. M. Dascalu, and F. C. Harris Jr, "A novel CPU/GPU simulation environment for large-scale biologically realistic neural modeling," *Frontiers in neuroinformatics*, vol. 7, 2013.

[11] J. M. Nageswaran, N. Dutt, J. L. Krichmar, A. Nicolau, and A. V. Veidenbaum, "A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors," *Neural networks*, vol. 22, no. 5, pp. 791–800, 2009.

[12] M. Richert, J. M. Nageswaran, N. Dutt, and J. L. Krichmar, "An efficient simulation environment for modeling large-scale cortical processing," *Frontiers in neuroinformatics*, vol. 5, 2011.

[13] M. Beyeler, K. D. Carlson, T.-S. Chou, N. Dutt, and J. L. Krichmar, "CARLsim 3: A user-friendly and highly optimized library for the creation of neurobiologically detailed spiking neural networks," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[14] J. C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, 1987.

[15] S. Sivagnanam, A. Majumdar, K. Yoshimoto, V. Astakhov, A. Bandrowski, M. E. Martone, and N. T. Carnevale, "Introducing the neuroscience gateway." in *IWSG*, 2013.

[16] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[17] Izhikevich, Eugene M, *Dynamical systems in neuroscience*. MIT press, 2007.

[18] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[19] C. F. Stevens and Y. Wang, "Facilitation and depression at single central synapses," *Neuron*, vol. 14, no. 4, pp. 795–802, 1995.

[20] J. Seamans and D. Durstewitz, "Dopamine modulation," *Scholarpedia*, vol. 3, no. 4, p. 2711, 2008, revision #90663.

[21] G. G. Turrigiano, "The self-tuning neuron: synaptic scaling of excitatory synapses," *Cell*, vol. 135, no. 3, pp. 422–435, 2008.

[22] G. Mongillo, O. Barak, and M. Tsodyks, "Synaptic theory of working memory," *Science*, vol. 319, no. 5869, pp. 1543–1546, 2008.

[23] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biological cybernetics*, vol. 98, no. 6, pp. 459–478, 2008.

[24] K. D. Carlson, J. M. Nageswaran, N. Dutt, and J. L. Krichmar, "An efficient automated parameter tuning framework for spiking neural networks," *Frontiers in neuroscience*, vol. 8, 2014.

[25] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, J. Bassett, R. Hubley, and A. Chircop, "ECJ: A Java-based evolutionary computation research system," *Downloadable versions and documentation can be found at the following url: http://cs. gmu. edu/eclab/projects/ecj*, 2006.

[26] E. L. Rounds, E. O. Scott, A. S. Alexander, K. A. De Jong, D. A. Nitz, and J. L. Krichmar, "An evolutionary framework for replicating neurophysiological data with spiking neural networks," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 537–547.

[27] M. Beyeler, N. Oros, N. Dutt, and J. L. Krichmar, "A GPU-accelerated cortical neural network model for visually guided robot navigation," *Neural Networks*, vol. 72, pp. 75–87, 2015.

[28] A. I. Gulyás, G. G. Szabó, I. Ulbert, N. Holderith, H. Monyer, F. Erdélyi, G. Szabó, T. F. Freund, and N. Hájos, "Parvalbumin-containing fast-spiking basket cells generate the field potential oscillations induced by cholinergic receptor activation in the hippocampus," *Journal of Neuroscience*, vol. 30, no. 45, pp. 15 134–15 145, 2010.

[29] E. M. Izhikevich, "Polychronization: computation with spikes," *Neural computation*, vol. 18, no. 2, pp. 245–282, 2006.

[30] V. Braitenberg and A. Schüz, *Anatomy of the Cortex: Statist. and Geometry*. Springer, 1991.

[31] D. F. Goodman and R. Brette, "The Brian simulator," *Frontiers in neuroscience*, vol. 3, no. 2, p. 192, 2009.

[32] A. K. Fidjeland, E. B. Roesch, M. P. Shanahan, and W. Luk, "Nemo: a platform for neural modelling of spiking neurons using gpus," in *Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on*. IEEE, 2009, pp. 137–144.

[33] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. R. Voelker, and C. Eliasmith, "Nengo: A python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, no. 48, 2014.

[34] S. Kunkel, A. Morrison, P. Weidel, J. M. Eppler, A. Sinha, W. Schenck, M. Schmidt, S. B. Vennemo, J. Jordan, A. Peyser, D. Plotnikov, S. Graber, T. Fardet, D. Terhorst, H. Mørk, G. Trensch, A. Seeholzer, R. Deepu, J. Hahne, I. Blundell, T. Ippen, J. Schuecker, H. Bos, S. Diaz, E. Hagen, S. Mahmoudian, C. Bachmann, M. E. Lepperød, O. Breitwieser, B. Golosio, H. Rothe, H. Setareh, M. Djurfeldt, T. Schumann, A. Shusharin, J. Garrido, E. B. Muller, A. Rao, J. H. Vieites, and H. E. Plesser, "NEST 2.12.0," Mar. 2017.

[35] D. Pecevski, T. Natschläger, and K. Schuch, "PCSIM: a parallel simulation environment for neural circuits fully integrated with Python," *Frontiers in neuroinformatics*, vol. 3, 2009.

[36] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris *et al.*, "Simulation of networks of spiking neurons: a review of tools and strategies," *Journal of computational neuroscience*, vol. 23, no. 3, pp. 349–398, 2007.