CARLsim 6: An Open Source Library for Large-Scale, Biologically Detailed Spiking Neural Network Simulation

Lars Niedermeier*, Kexin Chen†, Jinwei Xing†, Anup Das§, Jeffrey Kopsick¶,
Eric Scott‖, Nate Sutton**, Killian Weber‡, Nikil Dutt†‡ and Jeffrey L. Krichmar†‡
*Niedermeier Consulting, Zurich, ZH, Switzerland
†Department of Cognitive Sciences, University of California, Irvine, CA, USA
‡Department of Computer Science, University of California, Irvine, CA, USA
§Electrical and Computer Engineering, Drexel University, Philadelphia, USA
¶Interdepartmental Program in Neuroscience, George Mason University, Fairfax, VA, USA
‖Evolutionary Computation Laboratory, George Mason University, Fairfax, VA, USA
**Bioengineering Department, George Mason University, Fairfax, VA, USA
Correspondence Email: jkrichma@uci.edu

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892644

***Abstract*—Mature simulation systems for Spiking Neural Networks (SNNs) become more relevant than ever for understanding the brain and supporting neuromorphic computing. The CARLsim SNN platform is one of the first Open Source simulation systems that utilized CUDA GPUs to address the tremendous parallel processing demands of natural brains. It has evolved over almost a decade in numerous scientific research projects requiring efficient biologically plausible modeling at scale. With its sixth major release, CARLsim 6 respects this legacy by supporting the latest versions of operating systems, development tool chains, multi-core computers, and of course GPUs. It runs on a range of platforms; from Notebooks up to the NVIDIA DGX-A100 supercomputer, and is used in biologically plausible simulations of the hippocampus and neocortex. The latest version has added flexibility for incorporating long-term and short-term synaptic plasticity. Neuromodulation is an important property of neurobiology that can lead to rapid few shot learning, network rewiring, and neural activity modulation. Because of this, CARLsim 6 now supports four multiple neuromodulators for simulating neural excitability and synaptic plasticity.**

***Index Terms*—spiking neural network (SNN); neuromodulation; STP; STDP; GPCR; PKA/ PLC; LTP/ LTD; DA; ACh; NE; 5-HT;**

I. Introduction

Spiking Neural Networks (SNNs) provide a powerful method for modeling the dynamics of biological neural networks. Active research in the community calls for the development of more flexible and accessible tools for building and simulating SNNs. Such tools would provide an easy-to-use user interface, efficient and scalable network processing ability, and a rich set of native features that supports a wide variety of simulation needs.

CARLsim is a biologically detailed and large-scale SNN simulator. In previous iterations, CARLsim included a number of features that supported the simulation of complex networks, including synaptic conductances, spike-timing-dependent plasticity (STDP) that can be applied to either glutamatergic synapses (E-STDP) or to GABAergic synapses (I-STDP), Dopamine-modulated STDP (DA-STDP), and short-term plasticity (STP) [1]. CARLsim provides real-time and offline data analysis tools and an automated parameter tuning interface (PTI) to accelerate the creation and analysis of SNN models [1]. CARLsim also supports the utilization of multiple GPUs and multiple CPUs concurrently in a heterogeneous computing cluster, which greatly improves the efficiency in network simulations [2]. Recently, a Python interface to CARLsim was introduced, which retains the efficiency through the C++ core developments, and also further increases its accessibility to the computational neuroscience and machine learning communities.

In this release, new features in CARLsim 6 can be divided into system maintenance and functional enhancements. The first group is motivated to keep CARLsim up-to-date, e.g., by supporting the latest hardware and software stacks. Also, defects were addressed, documentation extended, and other general improvements such as the implementation of cMake to support all relevant target platforms. Furthermore, CARLsim 6 provides a Python version for the Offline Analysis Tool (OAT).

The functional improvements include the support of multiple neuromodulatory features which enables the implementation of the neuromodulatory system of mammals as defined by [3], [4], configuration of spike-timing dependent plasticity (STDP) and short-term plasticity (STP) at the connection level, rather than the post-synaptic group level (Fig. 1).

These features allow CARLsim users to utilize different hardware configurations and incorporate greater biological fidelity into their models. In the remainder of the paper, we describe the new system maintenance and functional features in detail. As in previous releases, CARLsim 6 is open-source, extensible, backwards compatible with prior versions, and publicly available on GitHub (https://github.com/UCI-CARL/CARLsim6).

978-1-7281-8671-9/22/$31.00 ©2022 IEEE

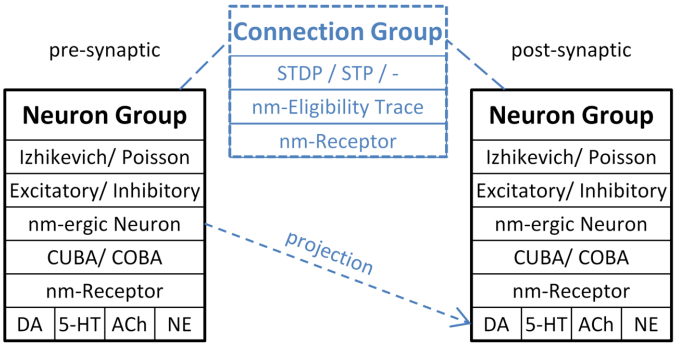


Fig. 1. Configuration of CARLsim 6 structural and neuromodulation features. CUBA/COBA can be configured at group level. Neuromodulatory neurons can project to other neuron groups. Plasticity is connection based. Receptors can be defined on both pre- and post-synapctic groups.

II. New Features

A. System Maintenance

1) cMake: cMake is recognized as the best practice build system for C++ projects by the open source community. cMake supports multiple operating systems like Linux, Mac OS, and Windows along with their distinct development environments like Microsoft Visual Studio, and Eclipse CDT for Linux as required by NVIDIA. Moreover, the features to be compiled and linked can be easily configured with build options in the cMake GUI or with the command line interface (CLI).

2) Windows Support: As cMake is able to generate project files for all current versions of Microsoft Visual Studio, CARLsim 6 restores the support of Microsoft Windows, which was deprecated in CARLsim 5 due to difficulty of maintenance. On Windows, CARLsim now deploys as optimized Dynamic Link Libraries (DLLs) that allow the integration as plug-ins, for instance in Qt. Additionally, CARLsim 6 provides extensions to PyNN that enables PyCARL [5] to run on Windows as well, which in turn enables working interactively with CARLsim in JupyterLab.

3) CUDA and GPU support: The support of all relevant CUDA versions and GPUs remains an important feature of CARLsim. CUDA was successfully tested on Linux and Windows up to the latest version, which was 11.5 at the time of writing. CARLsim 6 also adds supports for GPUs with the Ampere architecture, which is used by NVIDIA GForce RTX 3060 up to the NVIDIA DGX-A100. Due to the many possible combinations of operating systems, development stacks, CUDA versions, and GPUs, the CARLsim website lists tested systems and recommendations. This also includes performance benchmarks to serve as a guideline.

4) Binaries and Community Contributions: CARLsim 6 also provides binaries for the common platforms like Ubuntu 20.04 LTS and Windows 10/11. This is to facilitate evaluation by researchers and data scientists who want to evaluate CARLsim without the need to compile it from source, which is still a non-trivial task resulting from the strong dependency of CUDA on the target hardware platform. Contributions by the community are following the GitHub workflow. New features and fixes are organized in Git feature branches and are committed by the contributor as a GitHub Pull Request (PR). After the PR is successfully reviewed by the CARLsim development team, it is merged into the master branch, which triggers the continuous integration and build pipeline (CI/CD). If all quality gates have been passed, the binaries are created under the release tagged ”LATEST” and will be available for download.

5) Docker support: Docker images of Ubuntu 20.04 with CARLsim 6 installed and configured are also provided to help users switch from previous versions to CARLsim 6. Rather than installing the software on a bare machine, users have the option to create containers with the provided Docker images and quickly start using CARLsim 6 without worrying about the installation and configuration. This feature not only provides easier access to CARLsim 6 but also helps to deploy CARLsim 6 projects on cloud computing platforms (e.g., Kubernetes) that run containerized workloads and services.

6) Python OAT: Previous versions of CARLsim supported a MATLAB Offline Analysis Toolbox (OAT) that provided scripts to analyze the spike monitor and connection monitor output from SNN simulations. Because the MATLAB requirement was restrictive, CARLsim 6 introduces a more accessible Python OAT for these monitors.

7) Parameter Tuning with Evolutionary Computation Packages: CARLsim 6 provides an automated parameter tuning interface (PTI), which is integrated with two powerful evolutionary computation (EC) packages written in Java (ECJ) [6] and in Python (LEAP) [7]. The PTI assists in finding optimal parameters for the network simulation at different levels, from the level of single neurons to the level of the entire network, with a fitness function flexibly defined by the user that describes the need of their task. The evolutionary search process is accelerated with the support of parallel execution of multiple SNN instances in CARLsim. CARLsim in combination with ECJ has demonstrated success in reproducing neural dynamics observed in a number of brain regions [8]–[10]. As a newly integrated package, LEAP provides easy-to-use syntax and powerful visualization features.

CARLsim 6 introduces several functional enhancements that allow for SNN simulations with high biological fidelity. The additional parameters may add complexity to the parameter tuning process. The automated PTI can be used to tackle this challenge and efficiently search for appropriate parameter values.

B. Functional Enhancements

1) Group based Configuration of Input Current: In addition to the current based synaptic input (CUBA) that linearly combines the presynaptic spikes, the more realistic conductance based method (COBA) simulates the ionotropic receptors and its underlying neurotransmitters like gamma-aminobutyric acid (GABA) and glutamate. In previous versions of CARLsim, the choice of CUBA versus COBA was applied to the entire

simulation. In CARLsim 6 the calculation method of the input current can be individually defined for each neuron group to provide greater flexibility. The interface is backward compatible so that former models still work as do the network global defaults. Furthermore, this group based configuration was extended to implement metabotropic receptors that intrinsically rely on neuromodulators and which are also known as G-Protein Coupled Receptors (GPCRs) [11], [12], [13] like dopaminergic D1/2, norepinephrine alpha1, or the muscarinic for acetylcholine ACh. The neuromodulation will be described in more detail below.

2) Configurable STDP per Connection: In the previous releases, STDP was specified on the post-synaptic group, and all connections to this group with the same connection type (i.e., E-STDP or I-STDP) shared the same STDP parameters. The present release improves the flexibility in STDP setting and enables users to assign separate STDP parameters for each inter-group connection. This feature adds biological plausibility to the SNN model, and the additional parameters also increase the capacity of the model.

The code snippet below provides an example of specifying different STDP configurations for the two incoming excitatory projections into the output neuron group:

```
// connect neuron groups
sim.connect(gIn, gOut, ...
sim.connect(gOut, gOut, ...

// set E-STDP for the input connection
sim.setESTDP(gIn, gOut, true, STANDARD,
    ExpCurve(alpha_LTP_in, tau_LTP_in,
        -alpha_LTD_in, tau_LTP_in));

// set E-STDP for the recurrent connection
sim.setESTDP(gOut, gOut, true, STANDARD,
    ExpCurve(alpha_LTP_out, tau_LTP_out,
        -alpha_LTD_out, tau_LTP_out));
```

3) Configurable STP per Connection: In addition to STDP, CARLsim 6 enables users to configure STP for each inter-group connection. Various forms of STP have been observed in neural circuits for distinct connection types [14]; this updated feature allows for further variability to be included in an SNN by allowing STP variables u, τ_u, τ_v, and τ_d to be set for each connection.

The code snippet below demonstrates how STP is configured on an inter-group connection between Basket (inhibitory) and Pyramidal (excitatory) cells, two well documented neuron types in the Hippocampus:

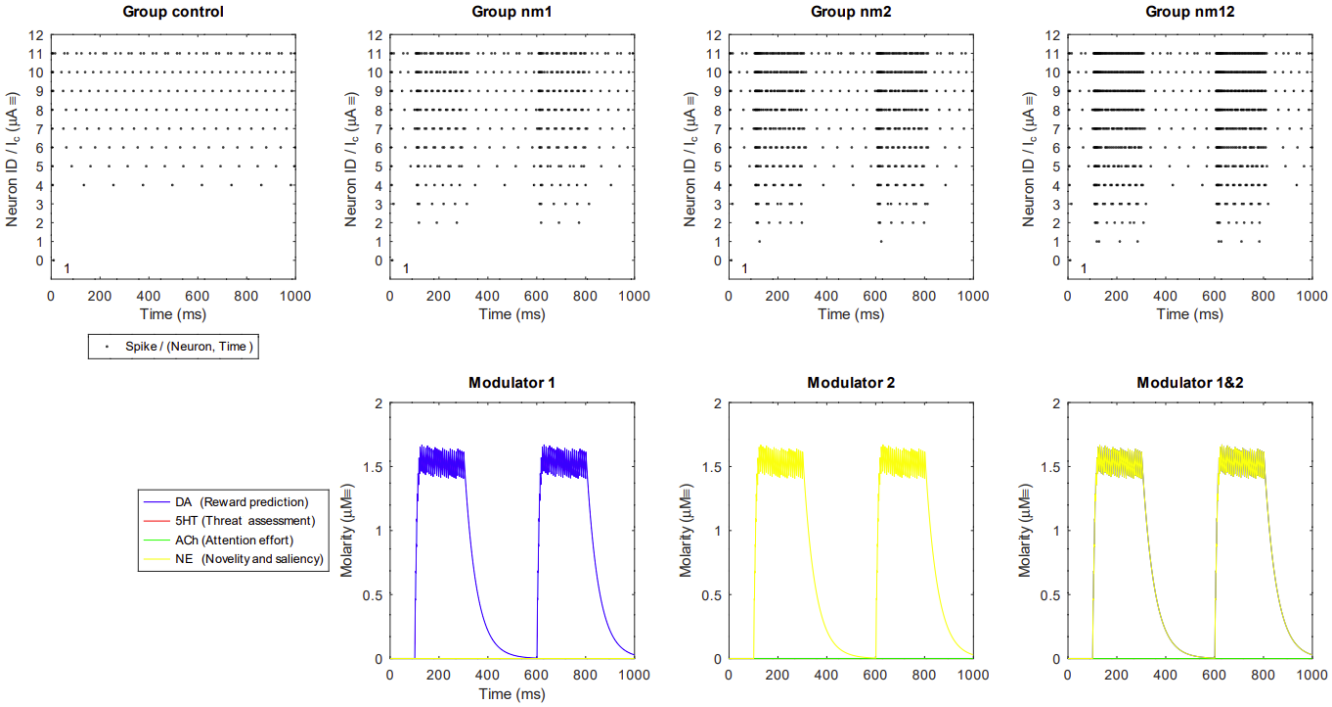
```
// configure STP on an inter-group
// connection between CA3 Basket
// and CA3 Pyramidal groups
sim.setSTP(CA3_Basket, CA3_Pyramidal,
           true,
           STPu(0.23f, 0.04f),
           STPtauU(16.74f, 2.0f),
           STPtauX(384.34f, 50.0f),
           STPtdAMPA(5.0f, 0.0f),
           STPtdNMDA(150.0f, 0.0f),
           STPtdGABAa(7.64f, 1.5f),
           STPtdGABAb(150.0f, 0.0f),
           STPtrNMDA(0.0f, 0.0f),
           STPtrGABAb(0.0f, 0.0f));
```

Currently a mean and standard deviation can be set for each STP variable. The example above includes a mean and standard deviation set for u, τ_u, τ_v, and for $AMPA$ and $GABA_A$ τ_d as these derived parameter estimates were based on receptors with fast synaptic properties.

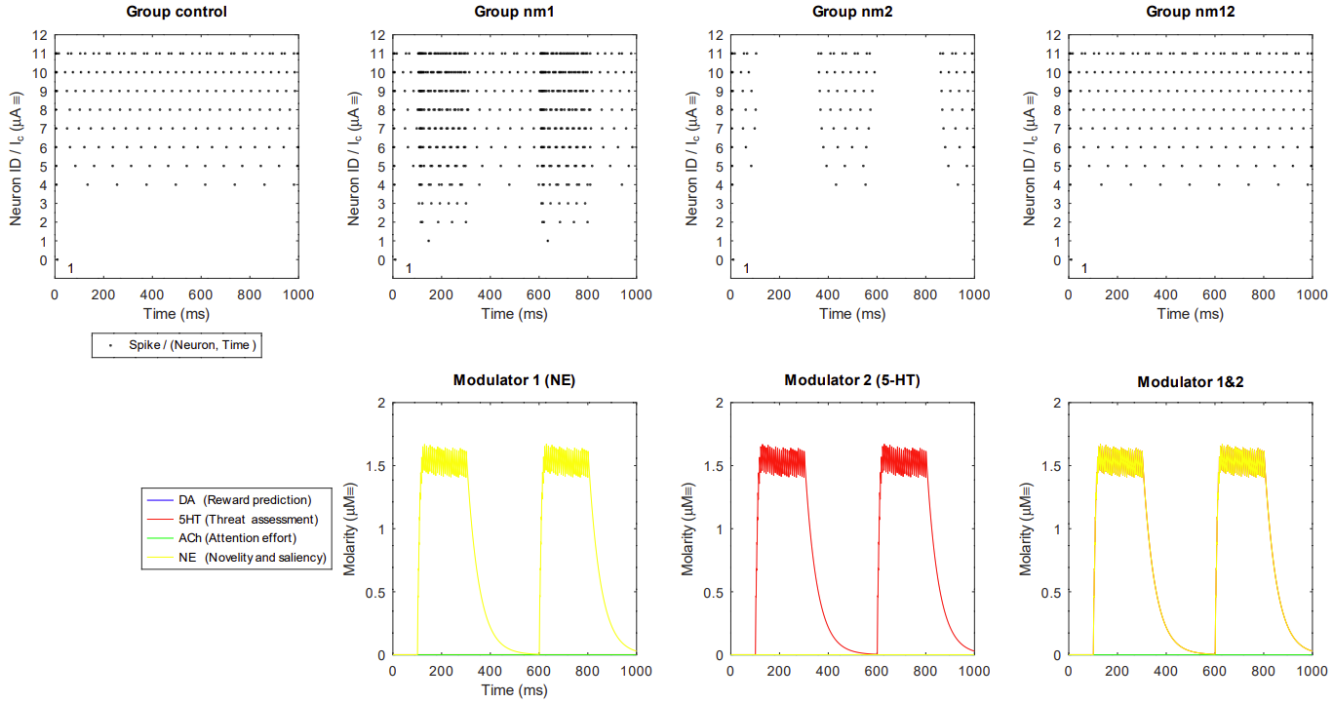
4) Neuromodulatory Effects on Target Groups: Unlike many neurotransmitters, neuromodulators (NMs) can have broad, long lasting effects on downstream neurons [15]. While single pathways of distinct NMs are well understood [16], it is clear that NMs also heavily influence each other [3], [4]. Therefore, CARLsim 6 provides equivalent support for four major neuromodulators (NM4): dopamine (DA), serotonin (5-HT), acetylcholine (ACh), and norepinephrine (NE). Their concentrations are captured as an NM4 molarity vector in the neuron group, which is targeted by the projection of neuromodulatory neurons. Consequently, the CARLsim offline analysis toolkit (OAT) was extended to visualize the molarity in the distinct target groups as shown in Fig. 2. A target group accumulates the molarity of all contained postsynaptic neurons of neuromodulatory neurons. The NM4 molarity vector is utilized as input for multivariate functions to implement neuromodulated excitability and plasticity.

In a previous custom extension for CARLsim, dopamine modulated post-synaptic facilitation (DA-PSF) was implemented within a model of the insular cortex [17]. However, multiple modulators can act on the same synapse to modify its strength, presumably depending on the behavioral need. Such effects can be drastic: 5-HT can functionally silence synapses, whereas dopamine can unmask synapses that are normally silent. The combined action of multiple neuromodulators on synapses can be more than simply additive, and the same neuromodulator can have opposing actions on synaptic strength [18].

With CARLsim 6, the calculation method of the input current can be individually set for a neuron group and parameterized by utilizing an extended form of linear combination that can be seen as a generalization of DA-PSF mentioned above. Fig. 2 presents current based examples for synergistic and antagonistic receptors, on which the concentrations of the neuromodulators of the post synaptic group individually strengthen or weaken the input current. The NM4 molarity vector is applied as input for the multivariate current calculation. The following code snippet demonstrates the configuration of an antagonistic receptor that is silenced for 5-HT, enhanced for NE, and neutral when both NM are at equilibrium, see also Fig. 2 (b). The synergistic receptor of Fig. 2 (a) is configured in the same way but with different parameters.



(a) Synergistic impact of NE on DA resulting in tonic to phasic excitability



(b) Antagonistic impact of 5-HT on NE with suppression and equilibrium

Fig. 2. CARLsim 6 allows the configuration of Izhikevich neurons with neuromodulated synergistic and antagonistic receptors, that exhibit non-linear excitability with an arbitrary crossover from tonic to phasic firing. The left column shows the firing in CARLsim's SpikeMonitor of such neurons without neuromodulation, labeled as control group. Each neuron was mapped to an increasing input current, displayed on the y-axis, with a designed crossover at about 10 μA ($\mapsto$ NeuronID 10). The simulation time is displayed on the x-axis in ms for each neuron group. (a) Synergistic effect on the neurons of the neuromodulators NE and DA. The molarity of the target groups are displayed in the CARLsim's GroupMonitor in the second row, and have the same values over time to demonstrate the synergistic effect on the receptor which is the changed crossover point of the firing from tonic to phasic. DA lowers the crossover down to 6 μA, while NE lowers it even more down to 3 μA, and with both NMs present, the phasic firing is almost instantaneous. (b) Antagonistic effects of 5-HT on NE, which is suppression for 5-HT and neutralization at equilibrium. The SpikeMonitor of group nm1 shows, that the antagonistic receptor has the same excitability for NE as above. However, when 5-HT is present, it acts as antagonist, and the SpikeMonitor of group nm2 displays gaps of spikes, when the neuron was silenced by 5-HT. The right column shows, that if both NM are present at the same level, the NM-effect is neutralized and group nm12 expose the same behaviour as the control group.

```
// target neuron group
sim.setNeuromodulator(g_nm,
 baseDA, tauDA, releaseDA, true,
 base5HT, tau5HT, release5HT, true,
 0.001f, 1.f, 0.f, false,
 baseNE, tauNE, releaseNE, true
);

// antagonistic receptor
sim.setNM4weighted(g_nm, NM4W_LN21,
 0.0f,       // DA
-1.0f/1.5f,  // 5-HT normalize and weight
 0.0f,       // ACh
 1.0f/1.5f,  // NE normalize and weight
 4.0f/2.0f,  // normalize all and boost
 2.0f        // unmodulated baseline
);

// synergistic receptor
sim.setNM4weighted(g_nm, NM4W_LN21,
 1.0f/1.5f,  // DA normalize and weight
 0.0f,       // 5-HT
 0.0f,       // ACh
 2.0f/1.5f,  // NE normalize and weight
 4.0f / (1.0f + 2.0f),
             // normalize all and boost
 1.0f        // unmodulated baseline
);
```

CARLsim allows Izhikevich neurons to exhibit non-linear excitability with crossover from tonic to phasic expression. Fig. 2 shows the firing of such a neuron in the control group for an increasing input current with a designed crossover at about 10 μA. For the synergistic receptor, DA lowers the crossover down to 6 μA, with NE even below 3 μA, and with both NMs present, the phasic firing is almost instantaneous, see Fig. 2 (a). In contrast, the receptor in Fig. 2 (b) has the same excitability for NE, but has 5-HT as antagonist that can silence the neuron if present alone or neutralize the effect of NE if both levels are the same.

5) Conductance Modulation of Receptors: CARLsim 6 implements the neuromodulated receptors NE$\alpha 1/\alpha 2A$ and DA$D1/D2$ of the prefrontal cortex working memory model as defined by [19], [20]. As shown in the code snippet below, the receptors are configured utilizing generic interface functions that are parameterized with the specific type of the receptor (e.g., *setConnectionModulation*). The pre- and post-synaptic groups are used to identify the connection group and assign the receptor settings (e.g., recurrent on Layer 3, MD/SC to Layer 5 for column c).

```
// noadrenergic alpha2A receptor
sim.setConnectionModulation(
    g_L3e[c], g_L3e[c], alpha2A_ADK13);
// dopaminergic D1 receptor
sim.setConnectionModulation(
    g_L3e_npref[d], g_L3e[c], D1_ADK13);
// dopaminergic D2 receptors
sim.setConnectionModulation(
    g_L5e[c], g_L5e[c], D2_AK15);
sim.setConnectionModulation(
    g_MD_SC, g_L5e[c], D2_AK15);
// alpha1 receptor (DA, NE, lambda)
sim.setNM4weighted(
    g_L3e[c], alpha1_ADK13,
    1.f, 0.f, 0.f, 1.0f, 1.f,
    -1.0f / 6.0f / log(1.0f / 3.0f));
```

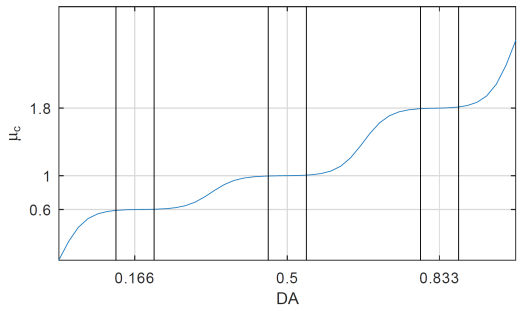
Fig. 3 shows the underlying continuous mapping of NE, DA that matches the discrete levels and implements the sensitivity dependency of the interaction between NE and DA and its effect on working memory as described in [19], [20]. Memory is properly maintained when these neuromodulators are balanced, but detrimental effects can occur due to concentration imbalances (see [19], [20] for details). According to the generic interface mentioned above, the internal implementation provides a framework that allows to implement other GPCRs in the same manner.

6) Neuromodulated STP: CARLsim 6 allows STP of an arbitrary neuron group to be influenced by neuromodulators. If the presynaptic neuron is repetitively active, STP can act as a gain-control mechanism, modifying synaptic strength as a function of the frequency of presynaptic activity, or in some cases, even switching the sign of synaptic dynamics from depression to facilitation and vice versa [18]. The following code snippet shows how the STP variables u, τ_u, τ_v [21] can be configured to be modulated by a multivariate function on the NM4 concentration to express facilitation as standard STP at a certain level. The modulated STP was validated to yield the same plasticity as standard STP. This new feature can be applied to elucidate synaptic gain and plasticity operations in the hippocampal subregion CA3 [22].

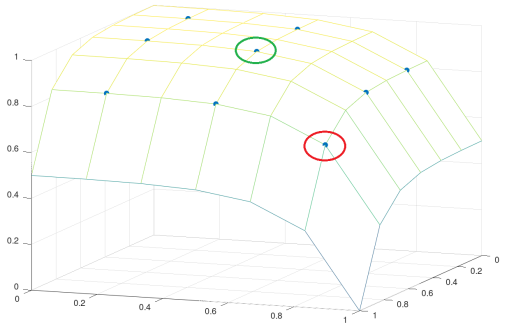
```
// facilitative STP
sim.setSTP(g1,true, 0.15f, 750.0f, 50.0f);

// modulated STP by serotonin
float u[]     = { 0.0f, 1.0f, 0.0f, 0.0f,
                  0.30f / 0.15f, 1.0f };
float tau_u[] = { 0.0f, 1.0f, 0.0f, 0.0f,
                  -700.0f / 750.0f, 1.0f };
float tau_x[] = { 0.0f, 1.0f, 0.0f, 0.0f,
                  700.0f / 50.0f, 1.0f };
sim.setNM4STP(g1, u, tau_u, tau_x);
sim.setNeuromodulator(g1,
        0.001f, 1.f, 0.f, false,
        1.0f; 1.f, 0.f, true, // base5HT
        0.001f, 1.f, 0.f, false,
        0.001f, 1.f, 0.f, false);
```

7) Eligibility trace based STDP: CARLsim 6 extends the eligibility trace based STDP for dopamine (DA-STDP) of former versions [17] equally to the other neuromodulators:



(a) Neuromodulatory factor $\mu_c(DA)$ of receptor D2



(b) Neuromodulatory factor $\mu(DA, NE)$ of receptor $\alpha 1$

Fig. 3. The working memory sensitively depends on a well-balanced concentration of neuromodulators. CARLsim 6 provides a continuous mapping of NE, DA to the neuromodulatory factor μ that matches the discrete levels investigated in [19], [20]. (a) Continous mapping of DA to the connection based neuromodulatory factor μ_c of the D2 receptor fitting the discrete levels (inside columns) given by [20]. (b) Continuous bivariate mapping of DA, NE to the group based neuromodulatory factor μ of the $\alpha 1$ receptor fitting the discrete bivariate levels (marked by blue dots) given by [19].

5-HT, ACh, and NE (see code snippet below). As STDP is connection based with CARLsim 6, multiple connection groups with different neuromodulators are now possible which is a precondition to implement the neuromodulatory system as defined by [3]. Also the offline analysis tool (OAT) was extended to support the monitoring of the NM level of the eligibility trace based STDP.

```
// set up dopamine modulated STDP
sim.setSTDP(gin, gPFC, true, DA_MOD,
 alphaPlus,tauPlus,alphaMinus,tauMinus);
// or for other modulators
sim.setSTDP(gin2, gPFC, true, SE_MOD,..
sim.setSTDP(gin3, gPFC, true, AC_MOD,..
sim.setSTDP(gin4, gPFC, true, NE_MOD,..
```

8) PKA/PLC modulated LTP/LTD of STDP: Neuromodulators play an important role in long-term potentiation (LTP) and depression (LTD) of mammalian central synapses. Different neuromodulators can change the balance of LTP and LTD and the effects on STDP reveal a simple rule: the activation of the protein kinase A (PKA) pathway, e.g., by beta-adrenergic receptors, promotes and gates LTP, whereas the activation of the phospholipase C (PLC) pathway, e.g., by M1 muscarinic receptors, promotes LTD. Also the activation of each pathway suppresses the other, suggesting a push-pull rule for the neuromodulation of long-term synaptic plasticity that seems to be independent of the underlying mechanisms of LTP and LTD [18].

With CARLsim 6, the PKA/PLC modulation extends the described scenario for any arbitrary neuromodulator pair, for instance NE and ACh. Fig. 4 shows a bivariate dynamic adaptation of the learning with seamless transformation of the usually statically configured parameters of STDP $(\alpha^+_{LTP}, \tau^+_{LTP}, \alpha^-_{LTD}, \tau^-_{LTD})$ [23]. With this implementation, SNNs can adapt not only the learning rate depending on the neuromodulators but also determine unlearning, which might be most relevant for learning by rewiring as described in [24]–[26]. The configuration is shown in the following excerpt of the unit test, that validates the PKA/PLC modulated STDP with the fixed values of standard STDP (see Fig. 4).

```
float ALPHA_LTP = 0.08f;
float ALPHA_LTD = -0.12f;
float TAU_LTP = 8.0f;
float TAU_LTD = 8.0f;
//enum FigNr  { _a,   _b,  _c,  ..
//float ne []={ 1.0f,  1.0f, 0.0f, ..
//float ach[]={ 1.0f,  0.0f, 1.0f, ..
//float a_p[]={ 0.08f, 0.16f, -0.08f, ..
//float a_m[]={-0.12f, 0.12f, -0.24f, ..

// set STDP group
g1 = sim->createGroup("excit", 1,
        EXCITATORY_NEURON);
sim->setNeuronParameters(g1,
        0.02f, 0.2f, -65.0f, 8.0f);

// set modulated STDP group
//g2 = sim->createGroup(..
//sim->setNeuronParameters(g2,..
sim->setNeuromodulator(g2,
 1.0f, 1.0f, 1.0f, false, // DA
 1.0f, 1.0f, 1.0f, false, // 5HT
 ach[fig], 1000000, 0.000001f, true,
 ne[fig],  1000000, 0.000001f, true);

// set ESTDP with reference values
sim->setConductances(false);
sim->setESTDP(gex2, g1, true, STANDARD,
                ExpCurve(
                  a_p[fig], TAU_LTP,
                  a_m[fig], TAU_LTP));

// set PKA_PLC modulated ESTDP
```

```
sim->setConductances(false);
sim->setESTDP(gex2, g2, true, PKA_PLC_MOD,
                ExpCurve(
                  ALPHA_LTP, TAU_LTP,
                  ALPHA_LTD, TAU_LTP),
                PkaPlcModulation(
                  NM_NE,  1.0f,   // pka
                  NM_ACh, 1.0f)); // plc
```

III. Benchmarks and Example Applications

A. Computational Performance

A strict objective for CARLsim 6 was to keep the same relative performance compared to its predecessor versions, despite the many kernel changes that were necessary for the new features described above. As shown in Fig. 5, CARLsim 6 achieves a similar performance as CARLsim 5 and also keeps the benefits of utilizing multiple GPUs as the size of spiking neural networks increases.

Moreover, due to the achieved backward compatibility and the thorough application of cMake, the latest GPUs are supported on all target platforms (Linux, Windows, MacOS). With the continuing increase in computing power by orders of magnitude every few years, we are able to exploit performance improvements with retail GPU cards that outperform former Tesla cards. For instance, the performance of Golosio [27] running on a Tesla K80 can be significantly improved on a desktop NVIDIA Geforce RTX 3090. By utilizing the NVIDIA Geforce RTX 3090 supported by CARLsim 6, we double the RAM (24GB vs. 12GB), quadruple the memory bandwidth (936 GB/s vs. 240 GB/s), triple the core clock speed (1400 MHz vs. 562 MHz), quadruple the number of transistors (28 mil. vs. 7 mil.), and quadruple the number of CUDA cores (10496 vs. 2496), which are critical for highly parallel single precision float operations that are necessary for numerical integration in some SNNs.

B. Compatibility with Neuromorphic Hardware

One of NVIDIA's keys to success was the cost reduction of GPU chips through mass production of their graphics cards for computer games, which ultimately led to their leading role today in supercomputing and artificial intelligence. As a consequence, GPUs still outperform neuromorphic hardware, especially for large scale models [28], but this comes at a significant energy cost. On the other hand, low power neuromorphic chips (e.g., Intel's Loihi 2 and BrainChip's Akida) have attained product level maturity, which makes them the primary choice for cloud edge devices and autonomous robots. The contribution of PyCARL [5] allows not only the development of models for such neuromorphic hardware but also to ensure spike timing when the SNNs are transferred to neuromorphic chips.

C. Large-scale model of a hippocampal subregion

Neural circuit models in the past commonly involved the modeling of a few neuron types and their corresponding connections to elucidate a particular function. To foster real-scale, more biologically realistic neural circuit models, one can think of different conceptual elements to build a network. These can include the counts and input-output relationships of each neuron type, the plasticity (short and long term) and probabilities of connection between neuron types, neuromodulatory effects on synaptic dynamics, along with how neurons and their connections are oriented in space. CARLsim 6 supports all of these components, and recently a real-scale neural circuit model (8 neuron types and 51 connection types; $\sim$90K neurons and $\sim$250M synapses) of mouse hippocampal subregion CA3 was built incorporating the majority of these conceptual elements utilizing parameters derived from Hippocampome.org [14], [22], [29], [30]. Leveraging CARLsim 6, a second of simulation time took 4 minutes of real time, and generated beta oscillations that have been observed in this hippocampal subregion during stationary behaviors [22]. Fig. 6 highlights some of the resultant activity that the model produces using the SNN Analysis Toolbox of [22], e.g., representative voltage traces from individual neurons from each excitatory and inhibitory neuron type, or the power spectrum of the local field potential (LFP) from pyramidal cells. CARLsim 6 therefore can be adapted and extended to support biologically realistic, data-driven neural circuit models of various brain areas, making it an attractive and effective simulation environment for its speed, scale, and complexity of modeling at the neural circuit level.

IV. Related Work

A. Deep Learning simulation environments

Deep Learning (DL) simulation environments for Artificial Neural Networks (ANNs) like PyTorch [31], and TensorFlow [32] are well developed and have been adopted by NVIDIA for GPU optimization [33], [34]. The widely accepted IDE is JupyterLab/Notebooks which also allows rapid prototyping and the ANN is entangled algorithmically within the Python code. The underlying neuron model is computationally friendly and hardware accelerated by sophisticated optimization based on NVIDIA TensorCores [35]. While Deep Learning (DL) based products have reached industrial standards in certain areas, there are intrinsic properties that also denote some of their shortcomings. For instance, DL models require "big data" and thousands of iterations to be trained properly. Especially, DL models are trained first and applied afterwards (inference). To overcome learning from scratch each time, a technique called "transfer learning" is utilized [36]. This contradicts multiple evidence, that both humans and rodents are capable to learn from one single experience [37]. Furthermore, in DL the information are encoded in the model structure and a new output (e.g. new classification item) usually induces a change of the network structure (output layer). This stands in contrast to biological organisms, whose learning is incremental from the beginning and continues during life [26]. Also the issue of catastrophic forgetting is an active area of research [38].

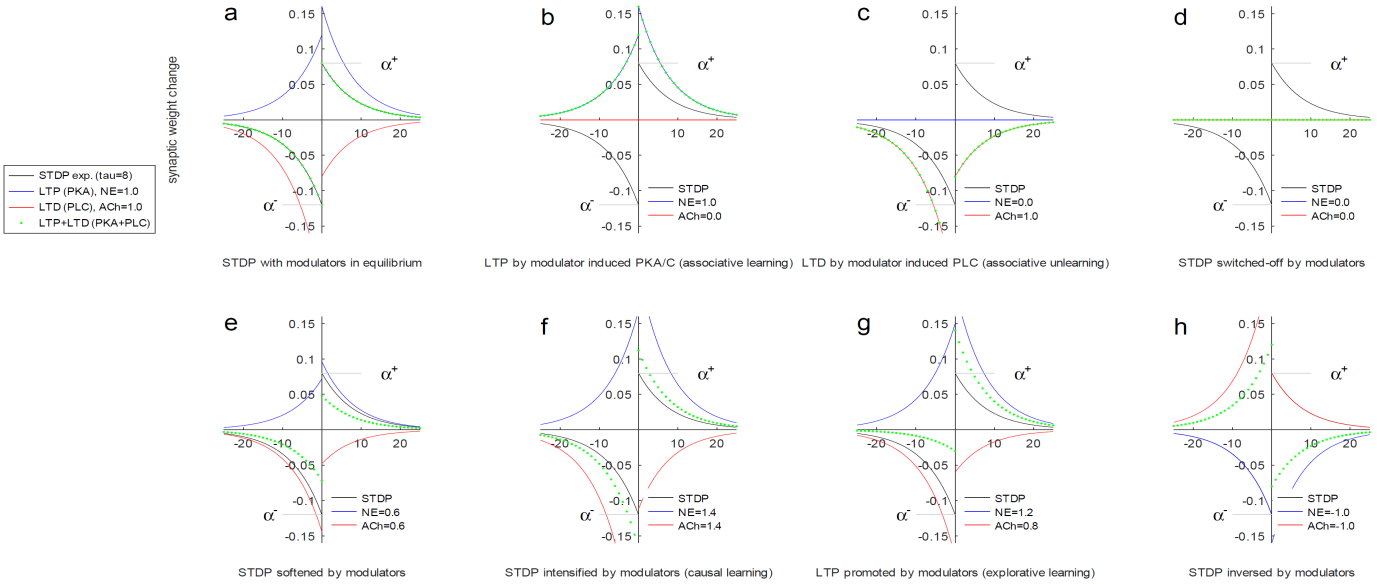


Fig. 4. LTP/LTD is modulated by NE and ACh and can adapt STDP based learning to environmental needs.

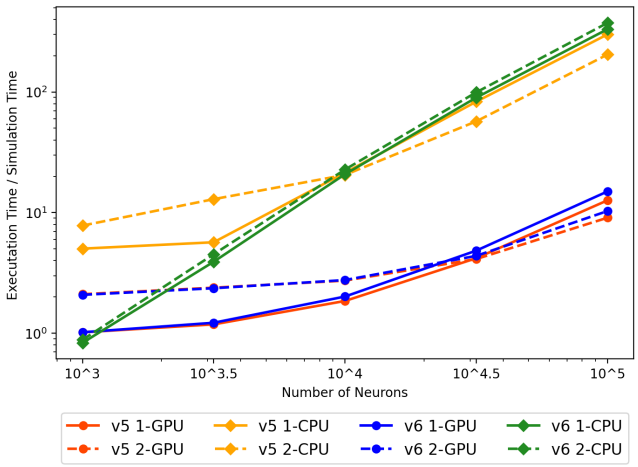


Fig. 5. Benchmark performance of CARLsim5 and CARLsim 6.

SNNs are biologically plausible neural networks that are based on the Hodgkin/Huxley differential equation system [39] and often classified as "third generation" NN. However, even with today's computing technology, this equation system is still too computational intensive and therefore not suited for large scale network simulations. To overcome this, Izhikevich simplified the equation system while preserving the 20 fundamental neurocomputational properties of biological neurons [40], [41], and found the simplest possible model capable of spiking, bursting, being an integrator or resonator that is therefore suited for large-scale bio-realistic artificial neural networks [42]. CARLsim 6 SNNs support temporal information, population coding, sparse coding and dimensionality reduction [43], as well as neuromodulation of neural circuits, which provide biologically detailed models and powerful computing tools that may benefit both the neuroscience and computer science communities.

B. Comparison to SNN simulation environments

We compare CARLsim 6 with the latest versions of open source SNN simulators that support parallel execution of SNN simulations, conductance based synapses, and synaptic plasticity. These simulators include[1]: Brian2 [44], GeNN [45], Nengo [46], NEST [47], NEURON [48], NeuronGPU [27]. Table I presents a comparison of supported features and software/ hardware requirements of different SNN simulation platforms.

Compared to other simulators, CARLsim 6 is advantageous in that it implements a wide variety of synaptic learning rules and biologically plausible model features, and is also highly optimized for large scale SNN simulations leveraging parallel execution with different computing hardware.

As a recently developed simulator, NeuronGPU targets GPUs as backend [27], and can be seen as a challenger in the area of performance as it optimizes the spike delivery algorithm. Although NeuronGPU has the similar asymptotic performance as CARLsim 6, its feature development seems still to be in an early stage as NeuronGPU has only nearest-neighbor STDP implemented as synaptic plasticity and no neuromodulation [27].

We also compare the support of neuromodulation in different SNN simulators, which is an important feature introduced in CARLsim 6. NEST can generate code for DA-STDP utilizing NESTML [47]. Brian provides an improved generator syntax for synaptic plasticity. Nengo applies BMC as unsupervised learning as equivalent for tripplet based STDP. The others like NCS or GeNN have not been changed in this functional area.

[1]To the best of our knowledge, NeMo and PCSIM, that have been included in earlier comparison to CARLsim [2], are not under active development.

Fig. 6. Representative voltage traces of individual neurons from each neuron type in the CA3 SNN (left), firing phase histograms of each neuron of each type relative to the beta oscillation the CA3 SNN produced (middle), and power spectra of CA3 Pyramidal neurons (right), produced from the SNN Analysis toolbox [22].

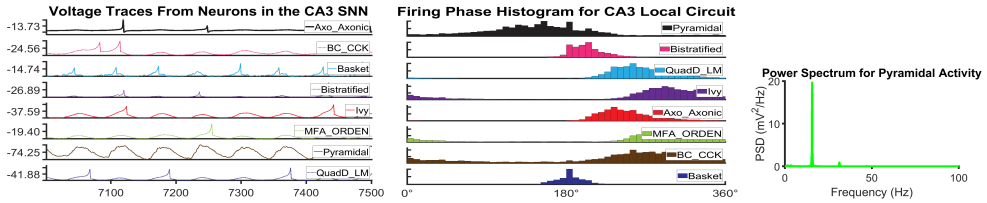


TABLE I

COMPARISON OF SNN SIMULATORS BY RECENT FEATURES. AN 'X' DENOTES THAT THE FEATURE IS DIRECTLY SUPPORTED BY THE SIMULATOR, WHILE A '/' MEANS THAT THE USER HAS TO IMPLEMENT CUSTOM CODE, RESPECTIVELY THAT THE FEATURE IS ONLY PARTIALLY IMPLEMENTED, AND A BLANK ' ' THAT FEATURE IS NOT AVAILABLE IN GENERAL. GRAY COLORED CELLS HIGHLIGHT FEATURES NEW IN CARLSIM 6 COMPARED TO PREVIOUS VERSIONS.

	CARLsim 6	Brian 2.5	NEURON	GeNN 4.6	NCS 6	Nengo 3.2	NEST 3.2	NeuronGPU
Synaptic plasticity								
DA-STDP	X	/	X			X	X	
5HT-/ ACh-/ NE-STDP	X	/	/			X		
Modulated LTP/ LTD	X							
Connection specific STDP	X	X	X				X	X
Neuromodulated STP	X	/	/					
Connection specific STP	X	X	X				X	
Synapse model								
Group level CUBA	X	X			X		X	
Group level COBA	X	X			X		X	
Neuromodulation	X	X	X	/		X	X	
Tools								
Parameter tuning (JAVA)	X					X		
Parameter tuning (Python)	X	/				X		
Analysis/ visualization	X	/	X		/	X		
Front-ends								
Python/ PyNN	X	X	X	X	X	X	X	X
C/ C++	X		X	X	X			X
Back-ends								
Single-threaded CPU	X	X	X	X	X	X	X	X
Multi-threaded CPU	X	X	X		X	X	X	X
Distributed		/	X		X	X	X	X
Single GPU	X	/		X	X	X		X
Multi-GPU	X				X			X
Hybrid (Multi-CPU/ GPU)	X				X			

V. CONCLUSION

The new features of CARLsim 6 enable the biorealistic implementation of SNNs. This latest version supports multiple neuromodulators and their effects on neuronal excitability and plasticity. STDP and STP are now at the connection level allowing greater flexibility. The introduction of the Python OAT and the Python based parameter tuning with LEAP allows more users to access analysis tools. Therefore CARLsim 6 has the potential to be an important tool for neuromorphic computing and the construction of neurobiologically inspired cognitive machines [26].

CARLsim 6 continues to be computationally efficient on multiple compute platforms. As CARLsim 6 is implemented in C++ and CUDA, the core library requires only a few MBytes of space and runs as native machine code on off-the-shelf robotics hardware like the NVIDIA Jetson. As the provided framework is highly backward compatible, CARLsim 6 can be utilized by CARLsim 4 and CARLsim 5 users to consolidate their implementations and to benefit from the latest hardware like the NVIDIA DGX-A100. CARLsim 6 is positioned to span a new ecosystem by motivating new projects to be able to build on a matured core library with a powerful support system. As Open Source under MIT License, it is available on GitHub (https://github.com/UCI-CARL/CARLsim6).

ACKNOWLEDGMENT

This work was supported by NSF award IIS-1813785, and by the United States Air Force Award FA9550-19-1-0306. The authors are thankful for the computing resources provided by CHASE-CI under NSF Grant CNS-1730158.

REFERENCES

[1] 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," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2015-Septe, 2015.

[2] T. S. Chou, H. J. Kashyap, J. Xing, S. Listopad, E. L. Rounds, M. Beyeler, N. Dutt, and J. L. Krichmar, "CARLsim 4: An Open Source Library for Large Scale, Biologically Detailed Spiking Neural Network Simulation using Heterogeneous Clusters," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2018-July, pp. 1158–1165, 2018.

[3] J. L. Krichmar, "The neuromodulatory system: A framework for survival and adaptive behavior in a challenging world," *Adaptive Behavior*, vol. 16, no. 6, pp. 385–399, 2008. [Online]. Available: https://doi.org/10.1177/1059712308095775

[4] M. C. Avery and J. L. Krichmar, "Neuromodulatory systems and their interactions: A review of models, theories, and experiments," *Frontiers in Neural Circuits*, vol. 11, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncir.2017.00108

[5] A. Balaji, P. Adiraju, H. J. Kashyap, A. Das, J. L. Krichmar, N. D. Dutt, and F. Catthoor, "Pycarl: A pynn interface for hardware-software co-simulation of spiking neural network," *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9207142

[6] S. Luke, "ECJ Then and Now," *CCS CONCEPTS Mathematics of computing*, vol. 8, 2017. [Online]. Available: http://dx.doi.org/10.1145/3067695.3082467

[7] M. A. Coletti, E. O. Scott, and J. K. Bassett, "Library for evolutionary algorithms in python (leap)," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1571–1579. [Online]. Available: https://doi.org/10.1145/3377929.3398147

[8] 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. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2014.00010/full

[9] E. L. Rounds, A. S. Alexander, D. A. Nitz, and J. L. Krichmar, "Conjunctive Coding in an Evolved Spiking Model of Retrosplenial Cortex," *Behavioral Neuroscience*, 2018.

[10] K. Chen, A. Johnson, E. O. Scott, X. Zou, K. A. De Jong, D. A. Nitz, and J. L. Krichmar, "Differential spatial representations in hippocampal ca1 and subiculum emerge in evolved spiking neural networks," *2021 International Joint Conference on Neural Networks (IJCNN)*, Jul 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9534139

[11] D. Bucher and E. Marder, "SnapShot: Neuromodulation," *Cell*, vol. 155, no. 2, pp. 482–482.e1, Oct. 2013.

[12] G. Pándy-Szekeres, M. Esguerra, A. S. Hauser, J. Caroli, C. Munk, S. Pilger, G. M. Keserű, A. J. Kooistra, and D. E. Gloriam, "The G protein database, GproteinDb," *Nucleic Acids Res.*, vol. 50, no. D1, pp. D518–D525, Jan. 2022.

[13] A. J. Kooistra, S. Mordalski, G. Pándy-Szekeres, M. Esguerra, A. Mamyrbekov, C. Munk, G. M. Keserű, and D. Gloriam, "GPCRdb in 2021: integrating GPCR sequence, structure and function," *Nucleic Acids Research*, vol. 49, no. D1, pp. D335–D343, 12 2020. [Online]. Available: https://doi.org/10.1093/nar/gkaa1080

[14] K. Moradi and G. A. Ascoli, "A comprehensive knowledge base of synaptic electrophysiology in the rodent hippocampal formation," *Hippocampus*, vol. 30, no. 4, pp. 314–331, Aug. 2019.

[15] Lakna, "What is the difference between Neurotransmitter and Neuromodulator," *PEDIAA*, 2019. [Online]. Available: https://pediaa.com/what-is-the-difference-between-neurotransmitter-and-neuromodulator

[16] M. R. Rosenzweig, S. M. Breedlove, and N. V. Watson, *Biological psychology*, 5th ed. Sunderland, MA: Sinauer Associates, Jun. 2007.

[17] T.-S. Chou, L. Bucci, and J. Krichmar, "Learning touch preferences with a tactile robot using dopamine modulated stdp in a model of insular cortex," *Frontiers in Neurorobotics*, vol. 9, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2015.00006

[18] F. Nadim and D. Bucher, "Neuromodulation of neurons and synapses," *Current Opinion in Neurobiology*, vol. 29, pp. 48–56, Dec. 2014, funding Information: We thank Isabel Soffer, Diana Martinez and Jorge Golowasch for their helpful comments. This work was supported in part by NIH grants NS083319 and MH060605 .

[19] M. Avery, N. Dutt, and J. Krichmar, "A large-scale neural network model of the influence of neuromodulatory levels on working memory and behavior," *Frontiers in Computational Neuroscience*, vol. 7, 2013. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2013.00133

[20] M. C. Avery and J. L. Krichmar, "Improper activation of d1 and d2 receptors leads to excess noise in prefrontal cortex," *Frontiers in Computational Neuroscience*, vol. 9, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2015.00031

[21] Beyeler, Michael, *CARLsim5 User Guide, 5.1 Short-Term Plasticity (STP)*, Cognitive Anteater Robotics Laboratory, University of Calfornia, Irvine. [Online]. Available: https://uci-carl.github.io/CARLsim5/ch5_synaptic_plasticity.html

[22] J. D. Kopsick, C. Tecuatl, K. Moradi, S. M. Attili, H. J. Kashyap, J. Xing, K. Chen, J. L. Krichmar, and G. A. Ascoli, "Robust resting-state dynamics in a large-scale spiking neural network model of area ca3 in the mouse hippocampus," *Cognitive Computation*, Jan. 2022.

[23] Carlson, Kristofor and Chou, Ting-Shuo, *CARLsim5 User Guide, 5.2 Spike-Timing Dependent Plasticity (STDP)*, Cognitive Anteater Robotics Laboratory, University of Calfornia, Irvine. [Online]. Available: https://uci-carl.github.io/CARLsim5/ch5_synaptic_plasticity.html

[24] J. Krichmar, "Design principles for biologically inspired cognitive robotics," *Biologically Inspired Cognitive Architectures*, vol. 1, p. 73–81, 07 2012.

[25] D. Bucher and E. Marder, "SnapShot: Neuromodulation," *Cell*, vol. 155, no. 2, pp. 482–482.e1, Oct. 2013.

[26] J. L. Krichmar, "A neurobiologically inspired plan towards cognitive machines," in *AAAI Spring Symposium: Towards Conscious AI Systems*, 2019.

[27] B. Golosio, G. Tiddia, C. De Luca, E. Pastorelli, F. Simula, and P. S. Paolucci, "Fast simulations of highly-connected spiking cortical models using gpus," *Frontiers in Computational Neuroscience*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2021.627620

[28] J. C. Knight and T. Nowotny, "Gpus outperform current hpc and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model," *Frontiers in Neuroscience*, vol. 12, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00941

[29] C. Tecuatl, D. W. Wheeler, N. Sutton, and G. A. Ascoli, "Comprehensive estimates of potential synaptic connections in local circuits of the rodent hippocampal formation by axonal-dendritic overlap," *Journal of Neuroscience*, vol. 41, no. 8, pp. 1665–1683, Feb. 2021.

[30] A. O. Komendantov, D. W. Wheeler, D. J. Hamilton, and G. A. Ascoli, "Simple models of quantitative firing phenotypes in hippocampal neurons: Comprehensive coverage of intrinsic diversity," *PLOS Computational Biology*, vol. 15, no. 10, p. e1007462, Oct. 2019.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[33] "Nvidia PyTorch release." [Online]. Available: https://docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes

[34] "GPU accelerated TensorFlow." [Online]. Available: https://www.nvidia.com/en-sg/data-center/gpu-accelerated-applications/tensorflow

[35] "Nvidia Tensor Cores." [Online]. Available: https://www.nvidia.com/en-us/data-center/tensor-cores

[36] "NVIDIA Transfer Learning Toolkit." [Online]. Available: https://developer.nvidia.com/tao-toolkit

[37] A. Wagatsuma, T. Okuyama, C. Sun, L. M. Smith, K. Abe, and S. Tonegawa, "Locus coeruleus input to hippocampal CA3 drives single-trial learning of a novel context," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, no. 2, pp. E310–E316, Jan. 2018.

[38] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: https://www.pnas.org/content/114/13/3521

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

[40] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, 2003.

[41] ——, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[42] ——, *Dynamical systems in neuroscience*, ser. Computational Neuroscience Series. London, England: MIT Press, Jan. 2010.

[43] M. Beyeler, E. L. Rounds, K. D. Carlson, N. Dutt, and J. L. Krichmar, "Neural correlates of sparse coding and dimensionality reduction," *PLOS Computational Biology*, vol. 15, no. 6, pp. 1–33, 06 2019. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1006908

[44] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, p. e47314, Aug. 2019.

[45] E. Yavuz, J. Turner, and T. Nowotny, "GeNN: A code generation framework for accelerated brain simulations," *Scientific Reports*, vol. 6, no. 1, p. 18854, Jan. 2016.

[46] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: A Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, 2014.

[47] R. de Schepper, J. M. Eppler, A. Kurth, P. Nagendra Babu, R. Deepu, S. Spreizer, G. Trensch, J. Pronold, S. B. Vennemo, S. Graber, A. Morales-Gregorio, C. Linssen, M. A. Benelhedi, H. Mørk, A. Morrison, D. Terhorst, J. Mitchell, S. Diaz, I. Kitayama, M. Enan, N. L. Kamiji, and H. E. Plesser, "NEST 3.2," Zenodo, Jan. 2022.

[48] N. T. Carnevale and M. L. Hines, *The NEURON Book*. Cambridge: Cambridge University Press, 2006.