An Evolutionary Framework for Replicating Neurophysiological Data with Spiking Neural Networks

Emily L. Rounds[1(✉)], Eric O. Scott[2], Andrew S. Alexander[3], Kenneth A. De Jong[2], Douglas A. Nitz[3], and Jeffrey L. Krichmar[1]

[1] Department of Cognitive Sciences, University of California, Irvine, Irvine, CA, USA
{roundse,jkrichma}@uci.edu
[2] Department of Computer Science, George Mason University, Fairfax, VA, USA
{escott8,kdejong}@gmu.edu
[3] Department of Cognitive Science,
University of California, San Diego, La Jolla, CA, USA
{aalexander,dnitz}@ucsd.edu

Abstract. Here we present a framework for the automatic tuning of spiking neural networks (SNNs) that utilizes an evolutionary algorithm featuring indirect encoding to achieve a drastic reduction in the dimensionality of the parameter space, combined with a GPU-accelerated SNN simulator that results in a considerable decrease in the time needed for fitness evaluation, despite the need for both a training and a testing phase. We tuned the parameters governing a learning rule called spike-timing-dependent plasticity (STDP), which was used to alter the synaptic weights of the network. We validated this framework by applying it to a case study in which synthetic neuronal firing rates were matched to electrophysiologically recorded neuronal firing rates in order to evolve network functionality. Our framework was not only able to match their firing rates, but also captured functional and behavioral aspects of the biological neuronal population, in roughly 50 generations.

Keywords: Spiking neural networks · Evolutionary algorithms · Indirect encoding · Neurophysiological recordings · Plasticity · Data matching · Parallel computing

1 Introduction

As the power and availability of high-performance computing resources grows, large and biologically realistic networks of spiking neurons are becoming increasingly relevant as a computational modeling tool. Networks consisting of on the order of hundreds or thousands of neurons allow researchers to formulate models that can represent how neural circuits give rise to cognition and behavior [12], and they allow engineers to prototype novel mechanisms that may prove useful in applications of neuromorphic hardware [9].

© Springer International Publishing AG 2016
J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 537–547, 2016.
DOI: 10.1007/978-3-319-45823-6_50

An important step in the design of these networks is the selection of parameter values that enable the model to perform a desired target function. Simulations of spiking neural networks (SNNs) tend to be very computationally expensive, and involve a large number of free parameters. For instance, even after a model of a neurological system has been constrained with the best available physiological data, it is not uncommon for an SNN to exhibit tens or hundreds of thousands of unknown synaptic weight parameters that must be specified by the model designer. Furthermore, SNN applications are often based on recurrent network topologies, where gradient-based optimization methods (such as backpropagation) are inapplicable. For these reasons, the task of parameterizing an SNN to solve a particular task, or to accurately model particular biological data, is an especially difficult kind of neural network optimization problem.

In this paper, we propose a two-pronged framework for tuning the parameters of spiking neural networks. First, we achieve a drastic reduction in the dimensionality of the parameter space by using a learning mechanism as an *indirect encoding* method for automatically adapting the weights of neural connections. This allows us to use an evolutionary algorithm (EA) to tune only the coarse-grained structure of the network and the global parameters of the learning method itself. Second, we use a GPU-based SNN simulator to accelerate fitness evaluation. This allows us to compensate for the increased computational effort that is required to train the networks through learning. To learn the synaptic weights, we apply a standard nearest neighbor implementation of spike-timing-dependent plasticity (STDP) [11], a widely-used and biologically realistic model of synaptic plasticity which has been studied experimentally [4] as well as computationally.

We demonstrate the functionality of this framework by applying it to a case study in which an SNN is tuned to match neural recordings from the rat retrosplenial cortex (RSC) [1]. To our knowledge, this is the first attempt to apply search algorithms to train SNNs to replicate neurophysiological data from awake, behaving animals. Existing work in the area of SNN synthesis has either trained recurrent networks to match high-level animal behavior in cognitive tasks [7,13,17], or it has focused on tuning the parameters of individual neuron models to match electrophysiological data [8,14–16]. However, in order to better understand the mechanisms underlying neurological circuits and to verify theoretical models of cognition, it is important that they are able to match neurological data in terms of neuronal firing rates as well as population functionality and behavior. Sometimes the choice of these parameters can be constrained by high-quality physiological data [20], but even with the best-understood brain regions we almost never know the precise value that these parameters should assume to best mimic nature. We show that this can be done effectively through the use of the present evolutionary parameter-tuning framework.

In general, neural networks have been successfully evolved using both direct and indirect encoding schemes. The NEAT and HyperNEAT algorithms [18,19] utilize an indirect encoding scheme in order to evolve increasingly complex network topologies, while Carlson et al. [5] used a similar approach to ours to

evolve SNNs whose neuronal responses gave rise to receptive fields similar to those found in neurons from the primary visual cortex. However, this study used artificial data and did not perform a behavioral task. Asher et al. [2] used a direct encoding scheme to train an artificial neural network (ANN) to perform visually- and memory-guided reaching tasks. However, this approach took thousands of generations to evolve, and yielded a network that had less biologically realistic neuronal units. To our knowledge, evolutionary algorithms utilizing indirect encoding have not been used to tune the parameters of networks containing realistic spiking neurons in order to perform a cognitive task.

To summarize, our approach is novel in three key ways: (1) We use biologically plausible spiking SNNs with realistic neural dynamics that not only reproduce the behavior of neural circuits, but also match empirical data at the neuron level while simultaneously capturing the holistic behavior of the circuit, (2) we use an indirect encoding approach evolutionary algorithm to tune SNNs, and (3) we use GPUs to run populations of SNNs simultaneously, thus speeding up the search process. This approach may be useful for replicating other neural datasets, and for creating biologically plausible SNNs.

2 Methodology

We test our STPD-based encoding method by fitting the activity of a network of 1,017 neurons to neurophysiological and behavioral data that have been previously collected by Alexander and Nitz from six male Long-Evans rats [1]. In neuroscience models, this topology is often loosely, manually specified based on the known, somewhat incomplete properties of a real structure in the brain. In the present case, we begin with a pre-specified network topology that defines the coarse-grained connectivity structure among several groups of neurons (Fig. 1). The goal of parameter tuning is to adjust the details of the network—such as synaptic weights, the number of connections between groups, and/or the behavioral parameters of the neurons in each group—such that the network successfully produces the desired target behavior.

2.1 RSC Model

In the current study, each SNN contained three groups of neurons, shown in Fig. 1: 417 excitatory input neurons, which handled the encoding of the behavioral inputs; 480 regular-spiking excitatory Izhikevich neurons and 120 fast-spiking inhibitory Izhikevich neurons [10]. The network had four types of connections: inputs to excitatory (`Inp`→`Exc`), inputs to inhibitory (`Inp`→`Inh`), recurrent excitatory (`Exc`→`Exc`), and inhibitory to excitatory (`Inh`→`Exc`). All synaptic projections were random with a 10 % chance of connectivity. No topology was enforced. To train the network, a learning rule known as STDP was used to update the weights of the network [4]; specifically, a standard nearest-neighbor implementation [11]. Homeostatic synaptic scaling was incorporated into the STDP rule in order to keep the neuronal firing rates within a reasonable regime by scaling to a target firing rate (for more details see Carlson et al. [6]).

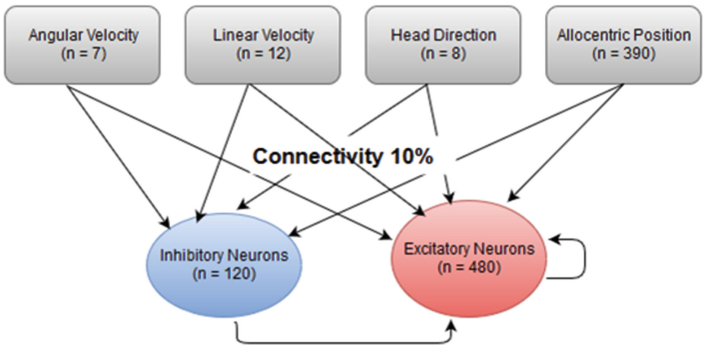


Fig. 1. The network topology used in the current study included four input groups, excitatory and inhibitory neuron populations, feedforward inhibition, and recurrent excitation, with 10 % connectivity between neurons.

2.2 Parameters and Training

The automated tuning framework was used to evolve a total of 18 parameters, which were related to plasticity, overall firing rates and weight ranges (see Table 1). These parameters were used as inputs to the CARLsim GPU-based simulation framework, which we used to run the SNN models [3,5]. Twelve parameters related to STDP were evolved, which correspond to three types of STDP curves. The remaining parameters control the homeostatic target base firing rates for the excitatory and inhibitory populations, and the initial and maximum weight values for each set of inter-group connections.

Table 1. Parameters initialized via the ECJ framework

Parameter	A^+	A^-	τ^+	τ^-	Base FR (exc)	Base FR (inh)	Inp-Exc Init.	Inp-Inh Init.	EE Init.	IE Init.
Minimum	−0.0002	−0.0002	5.0	5.0	5.0	5.0	0.01	0.01	0.001	0.001
Maximum	0.004	0.004	100.0	100.0	20.0	20.0	0.5	0.5	0.5	0.5
Std. dev	−0.00042	−0.00042	9.5	9.5	1.5	1.5	0.049	0.049	0.0499	0.0499

Each network was trained on a subset of trials from the Alexander and Nitz experiments [1]. In the present study, each generation underwent a training session and a testing session. Both consisted of 150 behavioral trials, which were drawn randomly from separate subsets of the total number of trials recorded to ensure that there was no overlap. In the testing phase, STDP was disabled in order to keep the synaptic weights fixed following training.

Following testing, the population was evaluated by summing the best correlations between the experimentally observed and simulated neurons for each SNN. The best correlations were found by first correlating every simulated neuron ($n = 600$) against every experimentally observed neuron ($n = 228$). Next, a match was chosen based on highest correlation value between each experimentally observed neuron and the corresponding simulated neuron (a neuron could only be chosen once). After all experimentally observed neurons had a

match, the fitness score for that individual SNN was computed by summing the correlations ρ between each pair (1). A maximum mean firing rate threshold was also incorporated into the fitness function to ensure that simulated firing rates were reasonable and realistic. The firing rate of each neuron in the network was averaged across all trials, and the highest observed value was considered the maximum mean. If the observed maximum mean firing rate `maxFR` exceeded the threshold, then the fitness score was penalized by subtracting the difference between the threshold and the observed firing rate (2):

$$f(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{n} \rho(\texttt{realFR}_i, \texttt{synFR}_{\texttt{match}}) & \text{if } \texttt{maxFR} < \texttt{FR}_{\texttt{target}}, \\ \sum_{i=1}^{n} \rho(\texttt{realFR}_i, \texttt{synFR}_{\texttt{match}}) - \texttt{FR}_{\texttt{error}} & \text{otherwise,} \end{cases} \tag{1}$$

where

$$\texttt{FR}_{\texttt{error}} = \texttt{FR}_{\texttt{max}} - \texttt{FR}_{\texttt{target}}, \tag{2}$$

and $\texttt{FR}_{\texttt{target}} = 250\,\text{Hz}$ was the maximum mean firing rate allowed for any given neuron.

After a generation, the fitness scores were sent to ECJ via the PTI for evaluation and constructing a new population. The simulations proceeded for 50 generations. The complete process was repeated 10 times to ensure repeatability. It is important to reiterate that the use of GPU processing speeds up the fitness function significantly. In this case, the fitness function runs 136,800 Pearson's r correlations (600 synthetic neurons multiplied by 228 neurophysiological neurons) per each individual, which is computationally very expensive. This complexity could increase considerably with the size of the dataset being replicated, the size of the network being run, and/or the number of individuals in the population, making it very important that the fitness function can be calculated in parallel on GPU.

2.3 Evolutionary Algorithm

We represented the parameter space of the RSC model as vectors in $\mathbb{R}^{18}$, and then applied a $(\mu + \lambda)$-style, overlapping-generations EA with truncation selection to maximize $f(\boldsymbol{x})$. We used a mutation operator that takes each parameter and adds 1-dimensional Gaussian noise with probability 0.5. The width of the Gaussian mutation operator was fixed at 10 % of the range that each parameter was allowed to vary within. The values of μ and λ were fixed at 3 and 15, respectively. It was straightforward to combine the SNN simulator with the ECJ evolutionary computation system [21] to create a unified parameter-tuning framework.

These decisions result in an algorithm with a small population and a strong selection pressure. This simple EA proved sufficient for our purpose, which is provide a proof of the feasibility of evolving SNNs with an STDP-based indirect encoding. We leave the problem of customizing EA design decisions to maximize performance for future work.

3 Results

3.1 Fitness Values and Firing Rate Correlations

Each of the 10 independent runs of the EA were executed for a small number of generations. Thanks to the indirect encoding, the best fitness found tended to be very high after just 50 generations (see Fig. 2(a)), with a mean of 105.93 ± 0.91. The highest observed fitness was 107.79. A total of 228 experimentally correlated neurons were matched, thus the average firing rate correlation was about 0.47 per neuron. The lowest observed fitness was 104.7, resulting in a correlation of about 0.46 per neuron (strong correlations by experimental standards). At the start of each evolutionary run, the average maximum fitness score was 84.57 ± 19.78.

Each of the ten evolutionary runs took 3.13 ± 1.26 days to complete. A breakdown of how long a generation took can be seen in Table 2. In the beginning, the population ran very slowly, taking approximately four hours to complete (slightly under two hours for training, and slightly more for testing). By the tenth generation, the population took roughly an hour to complete, which stayed relatively constant across the remaining generations (breaking down to about 20 min for training and 30 for testing). However, there was considerable variance in how long a generation could take at each point (each generation had a standard deviation of about one hour) because of the different firing rates of individual SNNs. Although the fitness increased during the evolutionary run, the selection strategy tended to include high and low firing SNNs in the population, which affects runtime performance in CARLsim.

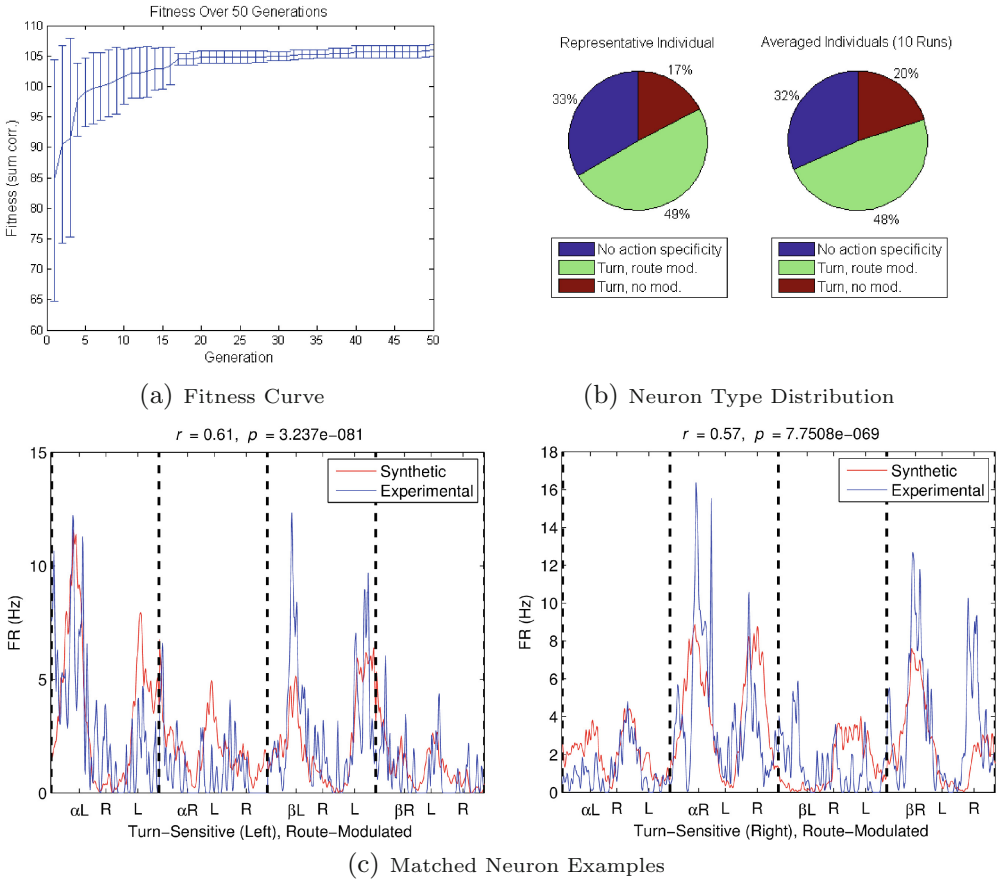
The tuning framework was able to closely match experimental neural activity to simulated neural activity. Figure 2(c) shows two representative examples of matched neurons with high correlations. Note that the correlation values in the figure are not much higher than the average correlation value, suggesting that they are typical examples of matched neurons indicative of the network's overall fitness. Thus the EA was able to generate networks whose neuronal firing rates were able to match those of the experimental dataset.

Table 2. Average runtimes in minutes (mean/std. dev)

Generation	1	10	20	30	40	50
Training	115.09/24.86	21.85/20.71	22.23/21.54	20.29/17.62	18.5/11.83	21.17/16.79
Testing	126.28/39.51	42.43/40.55	42.91/30.89	39.39/28.34	32.1/17.32	39.65/37.13
Total	240.48/59.24	64.42/59.57	65.3/50.4	59.94/45.25	50.66/28.38	60.91/49.86

3.2 Replicating Empirical Data

The evolved networks were also able to capture functional aspects of the neurons observed in the electrophysiological data. In agreement with Alexander and Nitz [1], we found neurons that were active when the animal was turning left or right (turn, no mod. in Fig. 2(b)) and turn cells that were route modulated (i.e.,



(a) Fitness Curve

(b) Neuron Type Distribution

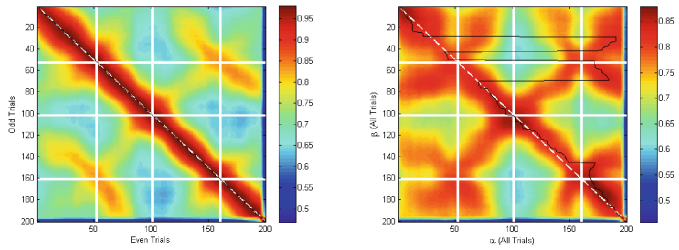
(c) Matched Neuron Examples

Fig. 2. (a) Network fitness rapidly and consistently converged over 50 generations. (b) All evolved networks yielded consistent distributions of neuron types. (c) Two representative matched neuron examples are provided, demonstrating that firing rate correlations between synthetic and experimentally-observed neurons were generally quite high.

preferring one turn over another on the same route; e.g., the first left instead of the second left on an LRL route; see turn, mod. in Fig. 2(b)), as well as neurons that were turn-insensitive (see no action specificity in Fig. 2(b)). In agreement with the experimental data, we found that approximately 20 % of the population were turn-sensitive, but were not route modulated. The ratios of turn-sensitive and route modulated cells found in the evolved SNNs were comparable to those found experimentally (compare our Fig. 2(b) with Fig. 3(a) in Alexander and Nitz [1]). However, we found a higher proportion of neurons that were route modulated (48 % as opposed to 26 %), and surprisingly, fewer of our neurons were turn-insensitive (32 % as opposed to 50 %). This may be because our SNN inputs, which were derived from recorded behavioral metrics, were less noisy than the sensorimotor signals that the rat uses to navigate.

3.3 Replicating Population Functionality

Lastly, the EA produced network behavior that was quite similar to the empirical findings, which is important because it suggests that the network functions similarly to the biological RSC, and thus has the ability to capture population dynamics as well as replicate biological neuronal firing rates. The evolved agent's position along a track could be discerned from the simulated population activity (see Fig. 3). Positional reconstruction was computed by cross-correlating mean neural activity across even vs. odd trials. Similar to that observed experimentally, the positional ensemble firing rate reconstructions from a representative evolved SNN clearly showed that the neural activity at positions on even trials was highly correlated with neural activity at the same positions on odd trials (Fig. 3(a)), thus very accurate reconstructions could be determined from population activity when the subject was in the same environment. That is, the highest correlation values occurred between the same bin numbers across even and odd trials, as shown by the white dashed line, from the top left corner to the bottom right corner. The reconstruction process was also applied to trials when the tracks were in different locations (α and β). Figure 3(b) shows a correlation matrix between positions in β and positions in α for the LRL trajectory. These reconstructions indicated that the position of the agent could be inferred between the track positions as well, but with less accuracy than for the even vs. odd reconstructions. This is consistent with the results reported in [1] (compare our Fig. 3(a) and (b) with Fig. 3(e) and 6(a) in Alexander and Nitz [1]), suggesting that the evolved simulated network is capable of conjunctively encoding allocentric and route-centric information similar to the biological RSC. These results were consistent across all evolved SNNs.



(a) αLRL (b) $\alpha\beta$LRL

Fig. 3. (a) Positional ensemble firing rate correlations (even vs. odd trials) which were highest fell along a 'perfect prediction line' suggesting that the network was able to infer its position along any given route so long as that route was in the same allocentric position. (b) Positional ensemble firing rate correlations for all trials at position α vs. position β deviated from the perfect prediction line, suggesting that the network discriminated routes that existed in different allocentric positions.

4 Discussion

In the present study, we introduced an automated tuning framework that leverages the search power of evolutionary algorithms combined with the parallelization of GPUs, which can result in a speedup of up to 60 times faster than a CPU in CARLsim [3]. This results in an efficient method for searching the SNN parameter space by drastically reducing its dimensionality via an indirect encoding scheme in which a learning rule, STDP, was used to specify the synaptic weights of each network. Performing fitness evaluation on each network in parallel further reduced the time necessary to tune the SNNs, even though every individual in the population was subjected to both a training and a testing phase. We successfully applied this framework to a case study in which it was used to evolve a model of the brain region RSC using electrophysiologically recorded neurons. Rather than altering the synaptic weights of each SNN directly, an evolutionary algorithm was used to alter the learning parameters of each SNN until a close match between synthetic and recorded neuronal firing rates was found, which resulted in a reduction of the number of parameters to be tuned from thousands to only 18. Furthermore, the evolutionary algorithm took only 50 generations to converge, demonstrating the framework was able to efficiently evolve a solution. This is in stark contrast to direct encoding methods of evolving neural networks, which can take thousands of generations to converge [2].

The phenomenological results of this case study suggest that the approach of using STDP as an indirect encoding scheme will generalize to other types of SNN tuning problems, and can be used to match other neurophysiological datasets, since many electrophysiological recordings are collected under conditions similar to the present dataset. First, the SNNs successfully captured the underlying network activity, which was reflected in the fitness score of each evolved network. Secondly, the SNNs captured neuronal function observed in the data, which was reflected in empirically observed distributions of non-route modulated turn-sensitive neurons and route modulated turn-sensitive neurons, respectively. Thirdly, the ensemble activity of the synthetic neurons captured behavioral functionality, such as position and route reconstruction.

The capacity to efficiently synthesize networks that reproduce neuron and network functionality across these three levels is of considerable importance as we attempt to move toward a greater understanding of brain function. We have demonstrated that we have created a powerful tool with this capacity by applying our framework to this case study of the RSC, which may be applied to a variety of modeling efforts and tuning problems involving SNNs. Further experiments are underway to investigate how the network responds to manipulations of its inputs, and to predict how neural activity in the retrosplenial cortex might change depending on environmental context. These predictions can then be tested by conducting new electrophysiological experiments, the results of which could lead to a better understanding of how neural responses give rise to behavior.

Acknowledgments. Supported by the National Science Foundation (Award IIS-1302125).

References

1. Alexander, A.S., Nitz, D.A.: Retrosplenial cortex maps the conjunction of internal and external spaces. Nat. Neurosci. **18**(8), 1143–1151 (2015)
2. Asher, D.E., Krichmar, J.L., Oros, N.: Evolution of biologically plausible neural networks performing a visually guided reaching task. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO 2014), pp. 145–152. ACM, New York (2014)
3. Beyeler, M., Carlson, K.D., Chou, T.-S., Dutt, N., Krichmar, J.L.: CARLsim 3: a user-friendly and highly optimized library for thecreation of neurobiologically detailed spiking neural networks. In: 2015 International Joint Conference on Neural Networks (IJCNN 2015), pp. 1–8. IEEE (2015)
4. Bi, G.-Q., Poo, M.-M.: Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. J. Neurosci. **18**(24), 10464–10472 (1998)
5. Carlson, K.D., Nageswaran, J.M., Dutt, N., Krichmar, J.L.: An efficient automated parameter tuning framework for spiking neuralnetworks. Front. Neurosci. **8** (2014)
6. Carlson, K.D., Richert, M., Dutt, N., Krichmar, J.L.: Biologically plausible models of homeostasis and STDP: stabilityand learning in spiking neural networks. In: The 2013 International Joint Conference on Neural Networks (IJCNN 2013), pp. 1–8. IEEE (2013)
7. Carnevale, F., deLafuente, V., Romo, R., Barak, O., Parga, N.: Dynamic control of response criterion in premotor cortex during perceptual detection under temporal uncertainty. Neuron **86**(4), 1067–1077 (2015)
8. Fountas, Z., Shanahan, M.: GPU-based fast parameter optimization for phenomenological spikingneural models. In: 2015 International Joint Conference on Neural Networks (IJCNN 2015), pp. 1–8, July 2015
9. Hu, M., Li, H., Chen, Y., Wu, Q., Rose, G.S., Linderman, R.W.: Memristor crossbar-based neuromorphic computing system: a case study. IEEE Trans. Neural Networks Learn. Syst. **25**(10), 1864–1878 (2014)
10. Izhikevich, E.M.: Simple model of spiking neurons. IEEE Trans. Neural Networks **14**(6), 1569–1572 (2003)
11. Izhikevich, E.M., Desai, N.S.: Relating STDP to BCM. Neural Comput. **15**(7), 1511–1523 (2003)
12. Krichmar, J.L., Coussy, P., Dutt, N.: Large-scale spiking neural networks using neuromorphic hardwarecompatible models. ACM J. Emerging Technol. Comput. Syst. (JETC), **11**(4) (2015). Article no. 36
13. Mante, V., Sussillo, D., Shenoy, K.V., Newsome, W.T.: Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature **503**(7474), 78–84 (2013)
14. Prinz, A.A., Billimoria, C.P., Marder, E.: Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. J. Neurophysiol. **90**(6), 3998–4015 (2003)
15. Prinz, A.A., Bucher, D., Marder, E.: Similar network activity from disparate circuit parameters. Nat. Neurosci. **7**(12), 1345–1352 (2004)
16. Rossant, C., Goodman, D.F.M., Fontaine, B., Platkiewicz, J., Magnusson, A.K., Brette, R.: Fitting neuron models to spike trains. Front. Neurosci. **5**(9) (2011)

17. Song, H.F., Yang, G.R., Wang, X.J.: Training excitatory-inhibitory recurrent neural networks forcognitive tasks: a simple and flexible framework. PLoS Comput. Biol. **12**(2), e1004792 (2016)
18. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artif. Life **15**(2), 185–212 (2009)
19. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
20. Tripathy, S.J., Savitskaya, J., Burton, S.D., Urban, N.N., Gerkin, R.C.: Neuroelectro: a window to the world's neuron electrophysiology data. Front. Neuroinf. **8** (2014)
21. White, D.R.: Software review: the ECJ toolkit. Genet. Program. Evolvable Mach. **13**(1), 65–67 (2012)