PLOS ONE

RESEARCH ARTICLE

BioNet: A Python interface to NEURON for modeling large-scale networks

Sergey L. Gratiy[1], Yazan N. Billeh[1], Kael Dai[1], Catalin Mitelut[2], David Feng[1], Nathan W. Gouwens[1], Nicholas Cain[1], Christof Koch[1], Costas A. Anastassiou[1], Anton Arkhipov[1]*

1 Allen Institute, Seattle, WA, United States of America, **2** University of British Columbia, Vancouver, BC, Canada

* antona@alleninstitute.org

Citation: Gratiy SL, Billeh YN, Dai K, Mitelut C, Feng D, Gouwens NW, et al. (2018) *BioNet*: A Python interface to NEURON for modeling large-scale networks. PLoS ONE 13(8): e0201630. https://doi.org/10.1371/journal.pone.0201630

Editor: Jordi Garcia-Ojalvo, Universitat Pompeu Fabra, SPAIN

Received: April 17, 2018

Accepted: July 18, 2018

Published: August 2, 2018

Data Availability Statement: The source code and test examples are available from the https://github.com/AllenInstitute/bmtk.

Funding: The study was funded by the Allen Institute and the University of British Columbia. The authors Sergey L. Gratiy, Yazan N. Billeh, Kael Dai, David Feng, Nathan W. Gouwens, Nicholas Cain, Christof Koch, Costas A. Anastassiou, and Anton Arkhipov received a salary from the Allen Institute, and the author Catalin Mitelut received a salary from the University of British Columbia. The funders had no role in study design, data collection

Abstract

There is a significant interest in the neuroscience community in the development of large-scale network models that would integrate diverse sets of experimental data to help elucidate mechanisms underlying neuronal activity and computations. Although powerful numerical simulators (e.g., NEURON, NEST) exist, data-driven large-scale modeling remains challenging due to difficulties involved in setting up and running network simulations. We developed a high-level application programming interface (API) in Python that facilitates building large-scale biophysically detailed networks and simulating them with NEURON on parallel computer architecture. This tool, termed "BioNet", is designed to support a modular workflow whereby the description of a constructed model is saved as files that could be subsequently loaded for further refinement and/or simulation. The API supports both NEURON's built-in as well as user-defined models of cells and synapses. It is capable of simulating a variety of observables directly supported by NEURON (e.g., spikes, membrane voltage, intracellular [Ca++]), as well as plugging in modules for computing additional observables (e.g. extracellular potential). The high-level API platform obviates the time-consuming development of custom code for implementing individual models, and enables easy model sharing via standardized files. This tool will help refocus neuroscientists on addressing outstanding scientific questions rather than developing narrow-purpose modeling code.

Introduction

Neuroscience is undergoing a profound revolution in the sensitivity and throughput of its experimental methods, which are beginning to yield large, systematic datasets relating to the structure and activity of brain circuits [1–4]. This revolution has spurred a growing interest in the neuroscience community towards the development of large-scale biophysically detailed network models as a means for integrating diverse sets of experimental data and discovering the principles that underlie functions of neuronal circuits. Although several such large-scale models have been developed [5–9], data-driven modeling studies at large scale remain very challenging, largely because adequate software tools are not yet available.

The development of simulation environments such as NEURON [10,11], GENESIS [12] and more recently Arbor [13] has greatly facilitated the process of setting up and running

and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

biophysically detailed network simulations. These environments allow users to develop models using neurophysiological concepts (e.g., dendritic sections, synapses, ionic conductances), while seamlessly handling numerical tasks such as solving the cable equation with ionic membrane mechanisms and tracking spike event arrivals to cells in the network. In particular, NEURON has become a widely used tool, due to its capability to simulate the activity of biophysically detailed cells and their networks, support of simulations on parallel hardware [14,15], and the implementation of a Python interface [16].

Despite these advances, modeling large-scale networks of biophysically detailed cells with NEURON still requires a substantial coding effort and detailed knowledge of the simulation environment. This is due to NEURON providing users with a scripting rather than high-level functionality interface for setting up and running network simulations. It is thus left to individual modelers to write code in the NEURON environment to set up and run their simulations, which can be a very laborious task. Typically, computational neuroscientists implement this functionality tailored to a particular model and for specific questions. Adopting such tailored simulation codes to different models usually requires extensive modification, hindering sharing and reuse of models and code for running simulations.

The challenges of model sharing and reuse are well recognized in the computational neuroscience community and a number of software tools facilitating network modeling with NEURON have been introduced (e.g., NeuroConstruct [17], PyNN [18] and NetPyNE (neurosimlab.org/netpyne/)). Motivated by the specific applications and operational philosophies, each suite is best suited for addressing different kinds of modeling problems. In turn, modeling large-scale networks (~100,000 neurons) of biophysically detailed cells (with hundreds of compartments each) brings its own specific considerations arising from computational demand for simulating complex models.

Development of large-scale biophysically detailed models typically requires a multi-step incremental procedure (e.g., creating populations of cells, adding an external input, adding recurrent connectivity) in conjunction with intermediate simulation and analysis stages. Each of these stages may be computationally expensive (e.g., for a network of ~100,000 neurons with sophisticated connectivity rules building the adjacency matrix can take several days when executed on a single processor), and therefore for large networks it becomes imperative to be able to save intermediate results of each stage. This calls for a modular incremental workflow where different modeling stages–building, simulation, and analysis–can be performed independently. Thereby, a constructed model could be saved to files, if necessary modified and elaborated, and later used as input to the network simulator and analysis tools. Furthermore, biological networks exhibit highly heterogeneous properties of cells and connections, requiring the software to be flexible enough to support custom user functionality for defining the composition of the network. At the same time, it should allow inexperienced users to perform standard operations without having to become experts in a particular scripting environment. Finally, large-scale modeling requires a versatile, modular, query-friendly and computationally efficient network description format for storage of model parameters enabling an interface between different modeling stages.

With these aims in mind, we developed BioNet, a modular Python application programming interface (API) facilitating building and simulation of large-scale biophysically detailed networks. The builder component of the API provides functionality for creating networks and saving the detailed network description to files based on user specified properties of cell types and connectivity rules. The simulator component implements an interface to NEURON and provides the functionality for loading the network description from files, instantiating a model on parallel computer architecture, setting up the simulation parameters, computing the desired observables (e.g., membrane voltage, extracellular potential) using NEURON and

saving the simulation results. Thus, BioNet on the one hand lowers the barrier for conducting modeling studies of biophysically detailed networks to scientists with very basic programming skills, while also accelerates model development for seasoned computational neuroscientists. BioNet is publicly released as a component of a modeling tool suite called Brain Modeling Toolkit (version 0.0.5) with the code hosted at github.com/AllenInstitute/bmtk and documentation and tutorials hosted at alleninstitute.github.io/bmtk/.

Materials and methods

Modeling paradigm

Modeling large-scale biophysically detailed networks involves several consecutive stages. Modeling usually starts with a conceptual (high-level) definition of the network under study: which brain regions to include, what level of detail to implement for cells and synapses, and how to treat their interactions. Often one can distinguish the explicitly simulated network model, which may include recurrent connections, from the feed-forward external inputs that are not treated explicitly in the simulation. The external inputs are influencing the neurons in the network, but the network does not influence the external inputs. All information about the external input can be computed separately from the network simulation, such that, for instance, their spike trains are pre-computed and saved, and are then loaded from files to drive the network simulation.

For instance, when modeling activity in the mammalian visual system (Fig 1A) retinal ganglion cells can be viewed as providing feed-forward external input to the Lateral Geniculate Nucleus (LGN) of the thalamus, which in turn links to the primary visual cortex (V1). In this case both LGN and V1 cells must be included as a part of a simulated network in order to study their interactions. Alternatively, if the V1 to LGN feedback does not need to be modeled directly, we may treat the LGN as a feed-forward relay to the V1 network (Fig 1B). The LGN activity can then be pre-computed (or taken from experimental recordings, see, e.g., [19]) and treated as external input to the simulated V1 network model.

Subsequently, we proceed with defining the components (e.g. cell types/models, number of cells, connectivity rules and synaptic types/models) of the external inputs and the simulated network (Fig 1C). These definitions are used to create individual instances of cells, connections and their specific parameters constituting a detailed description of a network model. Once all the descriptions of a network are supplied, the model may be simulated to produce network activity. The network activity then may be analyzed in combination with the detailed description of the network.

In the case of large-scale biophysically detailed networks, the building stage typically involves a multi-step procedure which can be computationally expensive. It is thus highly desirable to explicitly separate the building and simulation stages of the modeling. Thus, the building stage produces an explicit description of the network and saves it to disk for subsequent simulations and analysis. This enables multiple simulations to be run for the same network build–but with different run-time parameters such as external input, duration of simulation or what data to be saved (e.g. membrane voltage, intracellular calcium or extracellular potentials on different electrode layouts).

Model components

To build a BioNet model, users must decide on the component building blocks: models of individual cells and synapses corresponding to different cell and synapse types, as well as the connectivity rules between different cell types. BioNet facilitates simulations of neuronal activity at many levels of biophysical detail, ranging from realistic morphologies with time- and

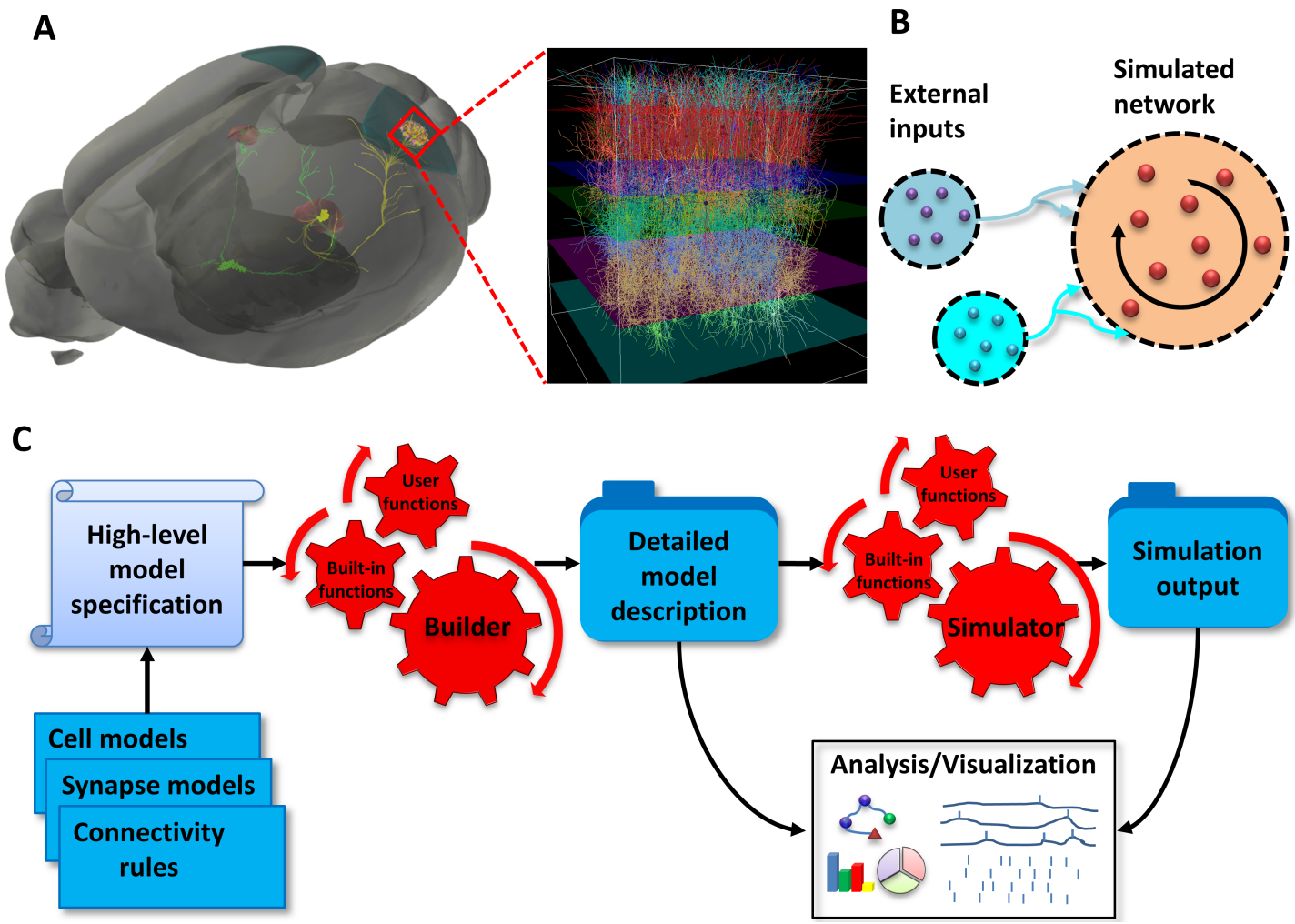


Fig 1. Modeling paradigm. (**A**) Modeling a large-scale network of neurons of the mouse primary visual cortex (area V1) as an example. Anatomically, retinal ganglion cells project (green), among other targets, to the Lateral Geniculate Nucleus (LGN) of the thalamus (red). The LGN cells project to V1 (zoomed-in view). In this example, the LGN activity is pre-generated and treated as external input, whereas V1 is simulated explicitly ("simulated network"). (**B**) Conceptual representation of the network model as a set of populations of external sources providing feedforward inputs (blue circles) and the simulated network (orange circle), which is recurrently interconnected. (**C**) Stages of the modeling workflow: The model components (blue stack) and the high-level model specification (scroll) are passed to the builder (red gears) to produce detailed model description as a set of files (blue folder). This network description serves as an input to the simulator (red gears) to produce the simulation output as a set of files (blue folder). In turn, the outputs from the simulator and builder serve as input for the analysis and visualization tools (white board).

https://doi.org/10.1371/journal.pone.0201630.g001

voltage-dependent ionic conductances to highly simplified integrate-and-fire point models (Fig 2). To utilize a particular model type of a cell/synapse, the user simply needs to specify a Python function returning an instance of a NEURON's cell or synapse. BioNet comes with several built-in functions that users may readily adopt to their models. Other custom models may be included by creating functions that may either fully implement biophysical and morphological properties, load them from files (e.g., JSON, SWC), or include a call to a NEURON's legacy HOC model. This capability gives users the flexibility of employing both built-in NEURON models of cells (e.g., IntFire) and synapses (e.g., Exp2Syn), as well as a variety of published models, such as those from ModelDB [20] and the Allen Cell Types Database [21–23].

Similarly, in order to specify the properties of connections between cells, the user needs to specify a function that determines connections between the cells based on user defined connectivity rules (e.g., distance dependence, orientation preference dependence, etc.). Generally,

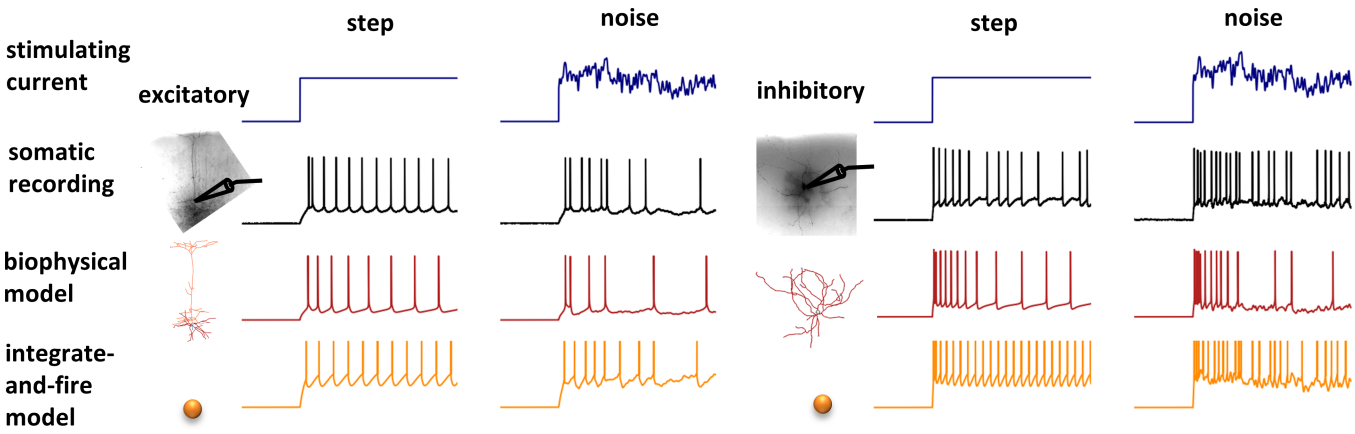


Fig 2. Examples of supported levels of detail for cell models. Experimental somatic recordings (black) in response to stimulating current (blue) can be modeled with BioNet at multiple levels of resolution (as long as they are supported by or can be implemented in NEURON). Here, two extremes are shown: biophysically detailed (red) and point leaky integrate-and-fire (LIF) models (orange). Example recordings and fits are from the Allen Cell Types Database.

https://doi.org/10.1371/journal.pone.0201630.g002

any combination of the cell's properties could be used to define the connectivity rules between two cells. For convenience, our builder toolkit provides a number of connectivity functions that are typically utilized in network modeling and serve as examples for creating custom connectivity rules.

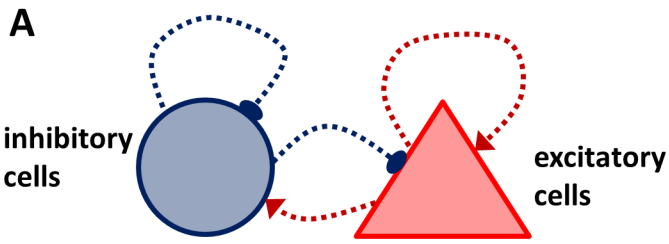
Results

Below we describe the operational workflow and functionality of our toolkits for building and simulation of large-scale biophysically detailed networks. We focus primarily on a user experience that does not require changing the source code, although, as the package is open-source, users can freely modify it or add functionality by consulting the online documentation.

Building networks

The builder toolkit facilitates constructing network models based on high-level user specification of a network composition from the available modeling components. In order to efficiently support the development of highly heterogeneous multi-purpose models at large scale, the builder was designed to satisfy several conditions. First, it provides functionality for assigning properties of cells and their connections according to user-defined rules. Second, it enables a step-by-step workflow where a portion of a model could be developed and saved to files and then subsequently loaded for further refinement. For instance, multiple feed-forward inputs and/or recurrent connections may be constructed in stages or even independently. This capability enables constructing different components of one model by a team of scientists and then merging them into a single interconnected network model. Third, an essential design consideration of a builder is that small changes to the model composition or parameters must require small effort on a user's part. For instance, changing biophysical properties of a cell type in an existing network will not affect connectivity properties of the network. Thus, when modifying such properties, the builder will not need to load or re-generate connectivity. This capability facilitates model refinement and adoption by different users.

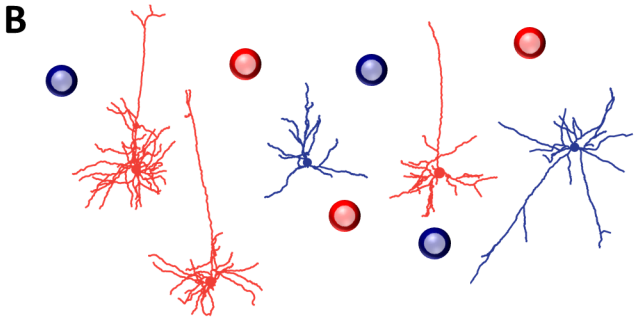
Building a network starts with specifying the high-level composition of a network: the cell types and the number of neurons of each type as well as the user functions defining properties of the cells and the connectivity rules (Fig 3A). The builder then interprets these rules to create a specific instantiation of a network that can be saved to files and subsequently loaded by



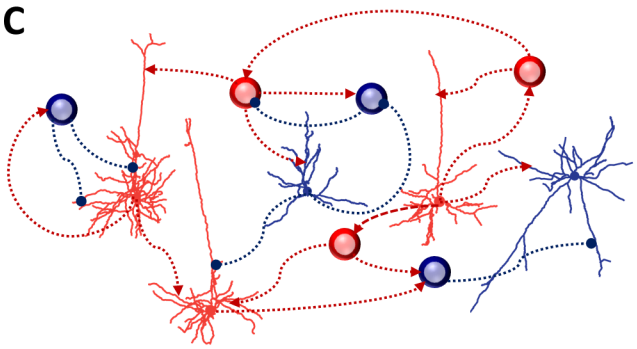

```
from bmtk.builder.networks import NetworkBuilder
import my_functions as myf
mynet = NetworkBuilder("my_network")

num_biophysical_exc = 3; num_lif_exc = 3
num_biophysical_inh = 2; num_lif_inh = 3

con_func = myf.my_connection_function
con_func_params  = {'property': value,...}
```




```
# ************ Create cells (nodes) *************
xyz_pos = myf.get_positions(num_biophysical_exc,
                **geometry_params)
par_files = ['params01.json',...]
morph_files = ['morph01.swc',...]
mynet.add_nodes(N = num_biophysical_exc,
                model_template = 'ctdb:Biophys1.hoc',
                pop_name = 'pyramids',
                model_type ='biophysical',
                positions = xyz_pos,
                morphology_file = morph_files,
                dynamics_params = par_files,...)
# similar for the remaining nodes, ...
```




```
# *********Create connections (edges) **********
mynet.add_edges(source={'model_type':'intfire'},
                target={'pop_name':'pyramids'},
                connection_rule = con_func,
                connection_params = con_func_params,
                target_sections=['somatic','basal'],
                distance_range=[0.0,150.0],
                syn_weight=0.009,
                weight_function='wmax',
                dynamics_params='inst.json',
                delay=2.0,
                model_template='exp2syn',...)
# similar for the remaining edges,...
```

Fig 3. Building networks. (**A**) High-level specification of a simple example network (left) and corresponding builder API commands (right). The model is composed of two cell types: inhibitory (blue) and excitatory (red), which exchange connections both across and between the cell types. The API commands define the number of cells of each type to be created, connectivity rule (con_func) to use and associated parameters (con_func_params) as well as additional edge parameters (edge_type_params). (**B**) Illustration of creating cells (left) where each cell type may include both biophysical (morphological reconstruction) and LIF models (circles). The corresponding API commands for adding nodes for the biophysically detailed subset of excitatory populations are illustrated on the right. Here we specify the number of nodes to be created (N), a type of a model (model_type), the dynamical cell models (model_template) and the corresponding model parameters (dynamics_params), morphologies (morphology_file), and positions of cell somata (positions) that were computed with a user-defined function. (**C**) Illustration of connecting the cells into a network (left) and the corresponding API commands for adding a particular subset of connections (right). Here, the cells satisfying the query for both the source and target nodes will be connected using a function (connection_rule) with parameters (connection_params). The additional edge_type attributes are shared across the added edges and include the synaptic strength (syn_weight), function modulating synaptic strength (weight_function), dynamical synaptic model (model_template) and corresponding parameters (dynamics_params), a conduction delay (delay), as well as the locations where synapses could be placed on a cell (target_sections, distance_range).

https://doi.org/10.1371/journal.pone.0201630.g003

BioNet to be simulated. Other instantiations can be generated by re-running the building stage with different random seeds. Recognizing that a model network is a graph, we can construct networks by first adding the graph nodes (cells) and then adding edges (connections) between the nodes.

The nodes are added to the network by invoking the `add_nodes(...)` command (Fig 3B). One can use it to add individual nodes, or groups of nodes that share common properties. The key-value pairs in the argument of a function are the node property names and the corresponding assigned property values. Each invocation of `add_nodes(...)` command creates a corresponding node type defined by the properties shared across the added nodes. Users can assign such common properties by assigning a single value to the property name (e.g., `pop_name = "pyramid"`). Alternatively, if a property has different values for each instance of the added node, user must specify an array of values with the size equal to the number of added nodes (e.g., `positions = xyz_pos`). The number of properties and their names are not restricted by the builder and the user is free to specify arbitrary key-value properties. However, to use the BioNet simulator, some parameters are mandatory. These include: (1) `model_type` (e.g., "biophysical", "intfire"); (2) `model_template` defining the dynamics of a model and the corresponding parameters `dynamics_params;` (3) `params_file/morphology_file` specifying the biophysical parameters of a model. Additional properties of the nodes can be added to be used when defining connectivity rules (e.g., `positions`) or for bookkeeping purposes (e.g., `population_name`).

Subsequently, the nodes are connected with the `add_edges(...)` command (Fig 3C). The cells involved in a connection are selected by specifying queries for the properties of the source and target nodes. Similar to nodes, each invocation of `add_edges(...)` command creates an edge type that can share some property defining the edge type (e.g., `dynamics_params = 'GABA_InhToExc.json'`). Alternatively, if the connection properties are specific to the individual connected pair, then the builder will find connected pairs using the `connection_rule` function with parameters in the `connection_params` dictionary and assign the desired properties to the individual connections. Users can specify any number of additional connection parameters using key-value pairs; however, to use with the simulator, some parameters are mandatory just as in the case of the nodes. These include: (1) `model_template` defining the dynamics of a model and the corresponding parameters `dynamics_params;` (2) `syn_weight` specifying the synaptic strength together with the function `weight_function` modulating synaptic strength; and (3) `a delay` specifying a conduction delay. In addition, depending on the way the connectivity is defined, the user may need to specify `target_sections` ("apical","basal","somatic" and"axonal") as well as `distance_range` (minimal and maximal distances from the soma on `target_sections`) for establishing the synapse placement.

Although previous community efforts resulted in the development of network description approaches (e.g., NeuroML [24] and NineML [25]), little effort has focused thus far on efficient description of large-scale networks, where data size, memory efficiency for simulations on parallel hardware and the ability to readily query and modify network parameters are the dominant determinants in the choice of the data structures. As a result, in the existing approaches the files describing network parameters (e.g., connectivity) tend to use unnecessarily large amounts of disk space and computer memory for representing realistic brain networks. Furthermore, modifying or adding network properties may be a computationally expensive task (for large networks), potentially requiring loading and rewriting the entire network description. BioNet offers a solution to these issues by storing parameters of the nodes and edges in separate but related binary tables (thus, saving disk space), enabling highly independent representation of network parameters. In addition, the parameters common to a group of nodes/edges can be grouped to define node/edge types and stored in separate node_type/edge_type tables enabling a parsimonious network description. Such a "type-instance" representation enables easy modification of the properties of the node/edge types without necessarily needing to iterate through individual nodes/edges, saving much computational cost for large networks.

For instance, one can easily change properties of synapse types by editing a small file containing edge type properties, without having to modify or regenerate the huge adjacency matrix file.

Running simulations

To run a simulation, the user first needs to prepare a configuration file specifying several run parameters (time step, simulation duration, etc.) as well as the paths to the files describing the network model, its components, recording electrodes and the desired output directories (Fig 4A). The configuration file (specified in JSON format,) is provided as a command line argument when executing the main Python script:

```
> python run_my_sim.py my_config.json
```

The main script calls the API's functions and custom user modules loading the network description, instantiating cells, connecting them, setting up desired recordings, and running the simulation (Fig 4B). More details for running simulations are provided in the online tutorial.

BioNet supports running simulations on parallel architecture via NEURON's Parallel Context library [15]. Briefly, the cells are distributed across M host processors in a circular (Round-robin scheduling) manner such that the k-th parallel process will simulate cells with IDs: k, k+M, k+2M, etc., thus hosting ~ 1/M-th of all cells. This pattern allows one to distribute the computational load approximately evenly, especially for networks with a large number of cells, even when they include highly diverse cell models (Fig 4C).

Simulation output

BioNet provides a flexible and customizable functionality for outputting variables computed by NEURON (e.g, transmembrane voltage) as well as computing custom output variables (e.g., extracellular potential) not provided by NEURON. The properties of the output variables are described in the configuration file that specifies sections, node_ids, the name of the output file as well as the Python classes that implements the output of each variable. BioNet provides a few built-in classes that implement the output of the somatic membrane voltage, intracellular somatic calcium concentration, spikes and the extracellular potential. This functionality is enabled in BioNet by interfacing into NEURON's standard run system allowing calls to user-defined classes at different stages of the simulation: at the beginning of the simulation, on each time step and when the simulation is completed. These classes serve as templates for users to define additional user-specific simulation outputs.

At each time step, simulation results are stored to a memory block and written into HDF5 files every N steps, as defined by the user in the configuration file. This gives users the ability to monitor the state of the simulation continuously (rather than waiting for the simulation to finish), and to analyze results while the simulation is in progress. Importantly this avoids filling the random-access-memory with recorded variables for long simulation runs.

During simulation, efficient data output is attained by each processor saving data independently into a temporary dedicated file in the centralized storage, and then, at the end of the simulation, combining data across the files generated by each processor. For spikes the data is simply merged into a single larger file. For extracellular potential recordings on a set of electrodes the contributions from all cells are added together in order to arrive at the cumulative extracellular potential.

Application Example: model of the Layer 4 of mouse V1

We demonstrate the application of BioNet for simulating activity of a network of the input layer 4 in mouse V1 receiving LGN input [9], which was constructed and simulated using an

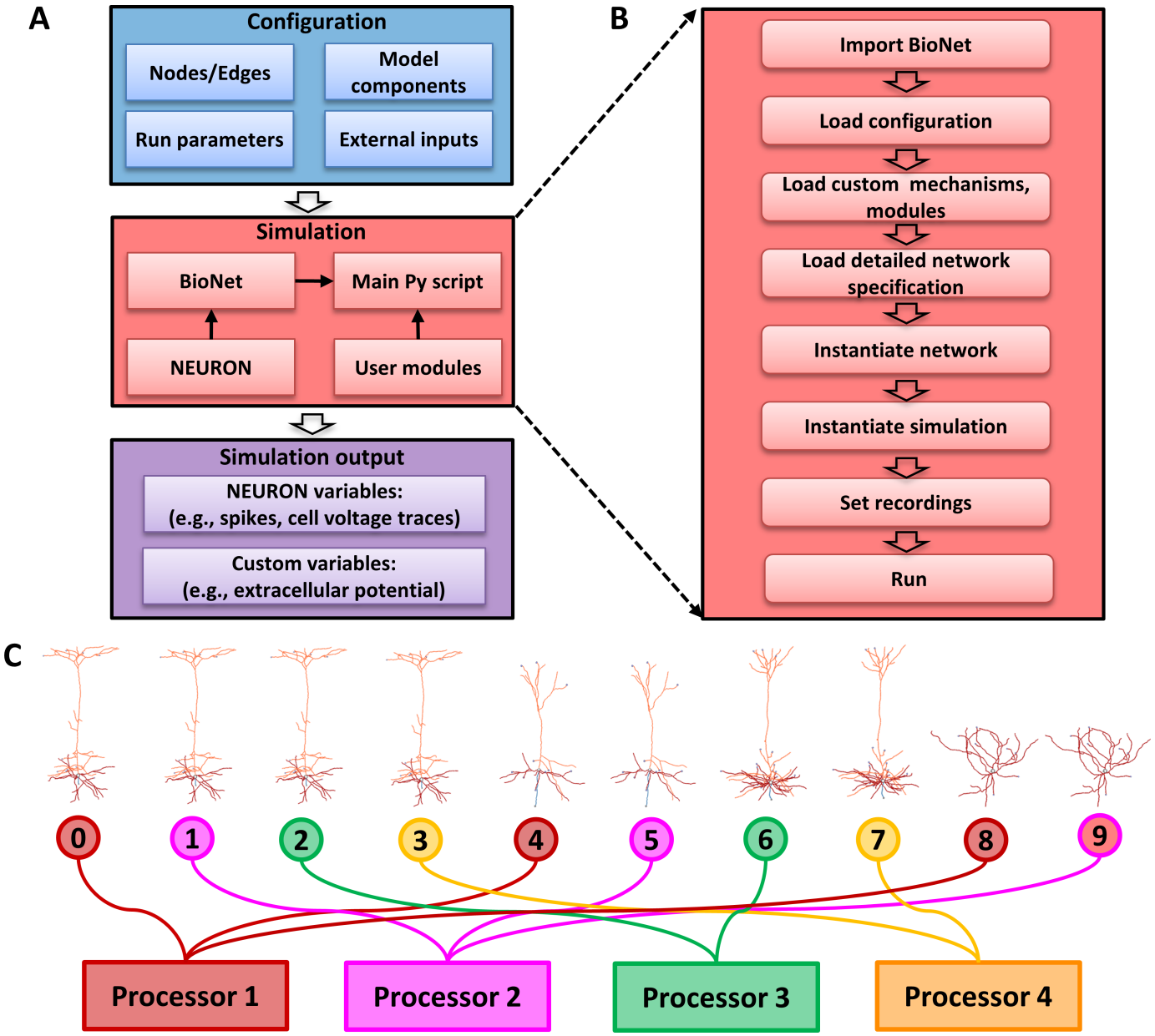


Fig 4. Running simulations. (**A**) Relationships among various elements involved in running simulations with BioNet. The pre-built network (blue), is passed to the main Python script (pink) that loads custom user modules and runs BioNet/NEURON to produce the simulation output (purple). (**B**) The stages of the simulation executed by the main Python script. (**C**) Algorithm for distributing the cells over a parallel architecture. This simple example shows 10 cells distributed across 4 parallel processes (typically each parallel process corresponds to a CPU core). Cells are assigned to each process in turn (a "round-robin" assignment).

https://doi.org/10.1371/journal.pone.0201630.g004

early development version of BioNet. The model is composed of a cylindrical "core" domain (radius 400 μm) within layer 4 (height 100 μm) in the central portion of V1 and includes 10,000 biophysically detailed cells (Fig 5A). The model also includes 35,000 leaky integrate-and-fire (LIF) cells within the "periphery" annulus (outer radius of 845 um), so that the biophysical cells at the border of the "core" have on average the same number of neighboring cells to connect with as those at the center. The biophysically detailed core includes 5 "cell types": three excitatory, as determined by major layer 4-specific Cre-lines (Scnn1a, Rorb, Nr5a1, 85%

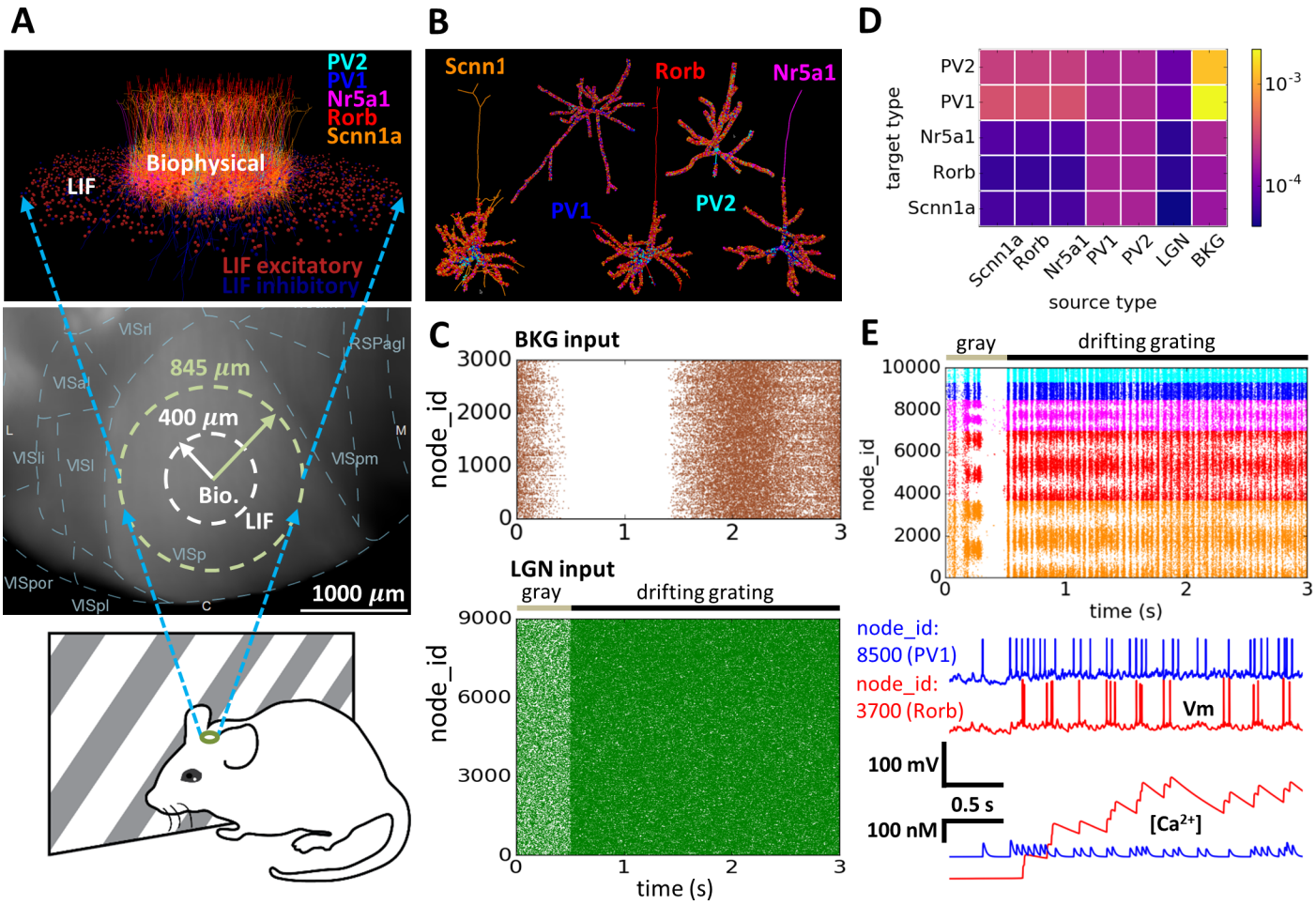


Fig 5. Application example: Model of the layer 4 in mouse V1. (**A**) The *in silico* study [9] mimicked *in vivo* visual physiology experiments (bottom), where a mouse watches visual stimuli such as, e.g., drifting gratings, while the activity of neurons in its cortex are recorded. (Center) The top view of the cortical surface, with boundaries of cortical areas delineated (VISp is V1). The inner boundary encloses part of the tissue that was modeled using biophysically detailed cells, whereas the tissue between the inner and outer circles was modeled using the simplified LIF cells. (Top) The 3D visualization of the layer 4 model (only 10% of cells are shown for clarity). (**B**) Example of synaptic innervation of the biophysically detailed cell models of each type. Synapses (depicted as spheres) are color coded according to their source cell type. (**C**) Rastergrams of the external inputs: (Top) "background" input (BKG, khaki) that switches between "ON" to "OFF" states, loosely representing different brain states; (Bottom) LGN input (green) corresponding to the visual response to 0.5 second gray screen (gray line) followed by 2.5 second drifting grating (black line). (**D**) The connection matrix showing the peak conductance strength for connections between each pair of cell types. (**E**) Simulation output: (Top) spike raster in the biophysical "core". The node_ids are ordered such that cells with similar ids have similar preferred orientation angle. In this example, cells preferring ~0, ~180, and ~360 degrees are responding strongly to a horizontal drifting grating. (Bottom) somatic voltage traces and the corresponding calcium traces for example excitatory (red) and inhibitory (blue) cells.

https://doi.org/10.1371/journal.pone.0201630.g005

of all cells), and two inhibitory, parvalbumin-positive fast-spiking interneurons (PV1 and PV2, 15% of all cells). Each "cell type" is modeled by a single cell model from the Allen Cell Types Database [21], and uses a compartmental model for the reconstructed dendritic morphology with active conductances at the soma and passive conductances at the dendrites (Fig 5B). The LIF cells on the "periphery" are modeled using one excitatory and one inhibitory "cell type" model. The connection weights and innervation domains depend on the pre- and post-synaptic types (Fig 5B and 5D). The connectivity in the network differed by cell type, distance, and orientation selectivity of neurons. In particular, one objective of this work was to test the role of the so-called like-to-like connectivity in the cortex, where cells preferring similar orientations of the grating stimuli are more likely to connect to each other than cells with different

preferred orientations [26–28]. Our builder functionality was important for enabling these complex and heterogeneous connectivity rules.

The network receives two kinds of feed-forward external inputs (that were pre-computed separately from the simulation of the V1 network): from the lateral geniculate nucleus (LGN) and a "background" (BKG) (Fig 5C). The LGN input is triggered by visual stimulation (first 0.5 second: gray screen; remaining time: a drifting grating like the example in Fig 5). The BKG input represents the input from the rest of the brain and alternates between "ON" and "OFF" states, accounting for different brain states. Both LGN and BKG activity are modeled with firing rate models that produce firing rates of individual cells that were subsequently converted to spikes utilizing Poisson process. A key advantage conferred by BioNet was that once a few instances of the V1, LGN, and BKG populations were constructed and saved to file, they could be easily mixed and matched to provide a large number of simulation configurations and trials, without the need to re-build connectivity or re-generate spikes from LGN and BKG for each new simulation trial.

The corresponding simulation output for the biophysical "core" of the layer 4 model with the spike responses as well as somatic voltage and $[Ca^{2+}]$ traces of several selected cells are shown in Fig 5E. The spiking activity demonstrates the prominent orientation selectivity of excitatory but not inhibitory cells, and oscillations in the ~20 Hz range, consistent with extracellular electrophysiological experiments [9]. Analysis of the simulation output conducted in the dedicated study [9] reveals that the model reproduces several findings from *in vivo* recordings, such as magnitude of responses to gratings, orientation selectivity, prominence of gamma oscillations, long-tailed distributions of firing rates, lifetime sparsity [29] for gratings and movies, the magnitude of cortical amplification, and others. Furthermore, the model makes predictions regarding the effect of optogenetic perturbations of the LGN or the Scnn1a population in layer 4 as well as the effects of the different connectivity rules on the observed activity [9].

Application Example: Computing extracellular potential

Extracellular potential is a major observable in electrophysiological experiments in vivo. The high-frequency component of the signal, the multi-unit activity (MUA), provides information about the spiking activity of neurons in the vicinity of the recording electrode contacts [30] and the low-frequency component, the local field potential (LFP), contains information about the collective coordinated activity of neuronal populations [31,32]. Recent advances in the multi-electrode fabrication for high density and high throughput recordings [33,34] led to increasing interest in the problem of interpreting extracellular potential recordings in terms of underlying neuronal activity, with modeling as an essential approach [35,36].

NEURON's built-in extracellular mechanism allows one to account for the properties of the extracellular space. However, this mechanism alone does not support computation of the extracellular potential at electrode sites receiving contributions from many nearby cells. A recently developed Python tool LFPy [37] allows calculating extracellular potential from biophysically detailed single cells and populations of unconnected cells simulated with NEURON. It was successfully applied for modeling extracellular potential recordings with multielectrode arrays *in vitro* [38] and *in vivo* [39,40] from the populations of biophysically detailed cells with the network activity pre-computed by an external simulator (e.g., NEST [41]).

BioNet enables further advances in the field by providing the capability for computing extracellular potentials from concurrent network simulations of biophysically detailed neurons in a self-consistent manner. The extracellular potential is computed using the line-source model [42]. It is further assumed that membrane current is uniformly distributed within individual computational compartments (i.e., NEURON segment) and that the extracellular

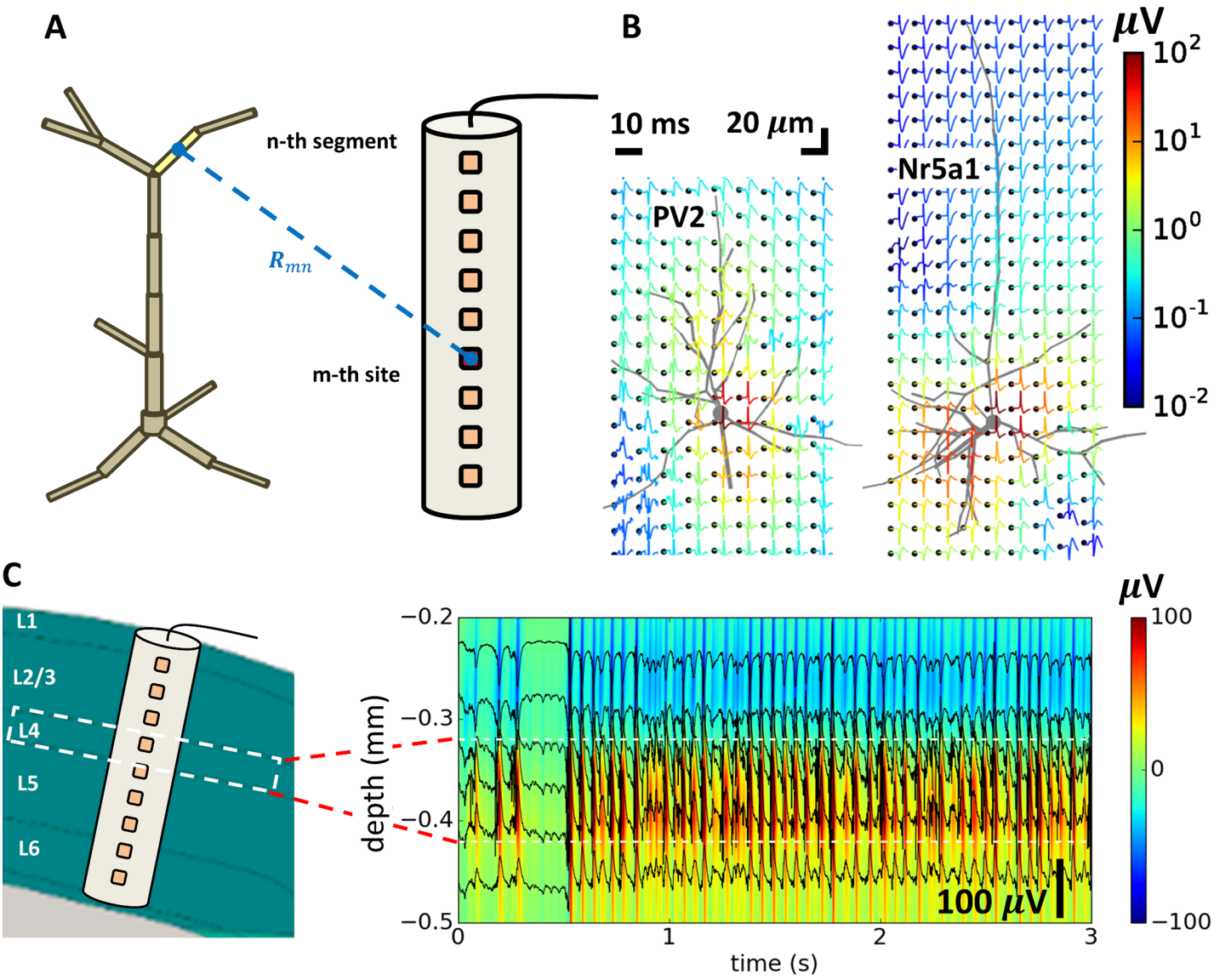


Fig 6. Computing extracellular potential. (**A**) Schematic of the compartmental model of a cell in relationship to the recording electrode. The calculation of the extracellular potential involves computing the transfer resistances R_{mn} between each n-th dendritic segment and m-th recording site on the electrode. (**B**) Extracellular spike "signatures" of individual cells recorded on the mesh electrode (black dots), using two single-cell models from the layer 4 network model as examples: PV2 (left) and Nr5a1 (right). (**C**) Modeled extracellular recordings with the linear electrode positioned along the axis of the cylinder in the layer 4 model (left). Extracellular potential responses (right) show all simulated data (color map) as well as from six select channels (black traces superimposed on the color map).

https://doi.org/10.1371/journal.pone.0201630.g006

medium is homogeneous and isotropic. The extracellular potential at the m-th location (i.e., electrode site) generated by a single cell is thus computed $\Phi_m = \Sigma R_{mn} I_n$, where I_n is the transmembrane current passing through the n-th segment and R_{mn} is the transfer resistance between n-th segment and the m-th electrode site (Fig 6A). The transfer resistances are computed using expressions that assume a uniform current distribution within individual segments [43,44]. The contributions of individual cells to the recordings (Fig 6B) are then summed up to find the total recorded potential from all cells at each recording electrode site. The example layer 4 model (see Sec. "Application Example: Layer 4 model of the mouse V1") presented above was also recorded with a virtual linear electrode positioned at the center of

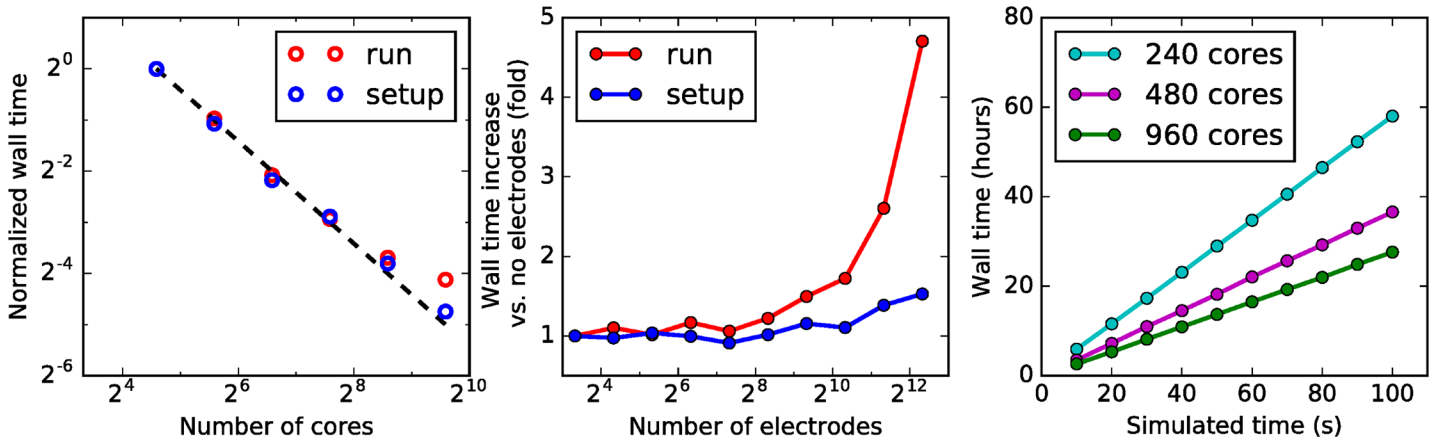


Fig 7. Computational performance. (**A**) Scaling of wall time duration (normalized by the duration on a single CPU core) with the number of CPU cores for the simulation set up (blue circles) and run (red circles) of the layer 4 model (see Fig 5). The ideal scaling is indicated by the dashed line. (**B**) Wall time increase when computing the extracellular potential for both set up (blue circles) and run (red circles) durations. (**C**) Scaling of the wall time with the simulated time for a long simulation. The non-ideal scaling with the increase in the number of cores corresponds to the deviations from the dashed line in (A).

https://doi.org/10.1371/journal.pone.0201630.g007

the network and perpendicular to the cortical surface and is shown in Fig 6C. Extracellular potential traces show the local field potential (LFP) manifested as low-frequency oscillations present across all electrode channels resulting from the collective neuronal activity in the network. Riding on top of the low-frequency LFP, traces also show the extracellular action potentials—manifested as sharp downward deflections—representing spikes from individual cells, whose somata are positioned sufficiently close to the recording sites. This capability of BioNet was utilized to simulate *in silico* ground truth extracellular recordings in order to test different electrode designs [34] and validate and improve spike-sorting algorithms [45].

Computational performance

We evaluated the run time performance of the simulations using the layer 4 model (see Sec. "Application Example: Layer 4 model of the mouse V1"). The simulations were performed on various numbers of nodes each including 24 hyper-threading cores (12 actual cores) of Dual Intel Xeon CPUs E5-2620 with clock speed of 2.00 GHz and 132GB RAM. Each node includes 2x 1000GB Hard Disks with CPUs connected via 1 Gb Ethernet. The wall time (that would be recorded by a wall-clock) for both set up and run scales nearly ideally for up to ~1000 CPU cores (Fig 7A). The slight loss of performance for large number of cores is in part attributed to NEURON's increased work load involved in the internal communication between processors [14].

The computation of extracellular potential adds only a small increase to the wall time for electrodes with up to ~1000 recording sites (Fig 7B). This is achieved by utilizing NEURON's cvode.use_fast_imem() function, which returns the transmembrane currents on each time step without needing to utilize NEURON's computationally expensive EXTRACELLULAR mechanism. However, for larger numbers of recording sites the matrix multiplication involved in computing the extracellular potential begins to dominate the computational burden. Thus, for number of sites greater than ~1000, this results in nearly doubling of the wall time when doubling the number of recordings sites.

The wall time scales linearly with the duration of the simulation, which allows longer simulations to be run (order of minutes of simulated time). As we show in Fig 7C, test simulations of the layer 4 model were carried out for 100 seconds of simulated time, while the compute time in these tests was strictly linear as a function of the simulation time.

Discussion

The Python-based API to NEURON developed here is designed to free neuroscientists' efforts from the need to write idiosyncratic, narrow-purpose modeling code and instead empower them to carry out systematic, large-scale high-throughput simulations to address scientific questions. BioNet enables functionality for building custom network models, described in a set of files and subsequently loading them to be seamlessly simulated with NEURON on parallel hardware. BioNet provides a Python interface facilitating the application of both NEURON's built-in and custom (e.g., previously published) models of cells and synapses. It provides users with the capability to execute custom run-time calculations similarly to how extracellular potential calculations were implemented (see Section "Computing extracellular potential"). BioNet is publicly available (github.com/AllenInstitute/bmtk) with accompanying documentation and tutorials hosted at alleninstitute.github.io/bmtk/.The challenges involved in constructing and simulating models with NEURON led to the introduction of a number of software tools designed to facilitating modeling neuronal networks. These include tool suites such as NeuroConstruct [17], PyNN [18] and NetPyNE (neurosimlab.org/netpyne/), each of which has been designed with a particular set of applications in mind. NeuroConstruct is a graphical user interface designed to simplify development of the biophysically-realistic small-scale models (that can be simulated on a single processor). It converts user specifications into simulation scripts that can be executed on a variety of simulator backends. In contrast, PyNN is a Python application programming interface designed to enable modeling at a high level of abstraction that supports parallel simulations for several simulator backends. While originally developed for simplified neuronal models, it is now expanding to also support the morphologically-detailed models. NetPyNE is designed as a dedicated suite of tools facilitating development, analysis, and parallel simulation of large biophysically detailed networks with NEURON. It utilizes hierarchical data structures for both the high-level specification of networks as well as for instantiated networks that can be stored to files for subsequent simulations.

In turn, BioNet was specifically designed to address the challenges faced when developing large-scale biophysically detailed networks that are very computationally expensive to construct and simulate. BioNet addresses these challenges by utilizing a modular approach to modeling, where building and simulation stages are distinctly separated. Thereby a network can be built in incremental steps and saved in files for subsequent refinement and simulation. This capability is supported in BioNet by describing properties of nodes and edges using a relational data model [46] that enables a highly independent and parsimonious representation of network parameters. Thus, small changes to a model require small effort on user's part, enabling rapid hypothesis testing regarding the role of various structural components on network function.

The modular interface design and capabilities of BioNet should help address the current challenges of model sharing and reuse present in the computational neuroscience community. We believe that the adoption of such a modular workflow—whereby the building, simulation and analysis tools are fully independent and interface via file exchange—will encourage wider application of biophysically detailed modelling. This approach has been very successful, for example, in the field of molecular dynamics simulations of proteins, where tools for model building, simulation, and visualization interface with each other via commonly supported file formats [47–51]. In turn, this will lead to the advancement and adoption of the common network descriptions interfacing these tools, which has been lagging behind (although some promising initiatives are underway, e.g., NeuroML [24]). Taken together, these efforts will empower neuroscientists to develop reusable large-scale, data-driven biophysically detailed

network models and make large-scale modeling accessible and appealing to a much broader range of neuroscientists.

Acknowledgments

We wish to thank the Allen Institute founders, Paul G. Allen and Jody Allen, for their vision, encouragement and support. We are grateful to Michael Hines for many discussions and suggestions related to the NEURON simulation environment. We are grateful to Peter Groblewski (Allen Institute) and Brandon Blanchard (Allen Institute) for help with illustrations. We also wish to thank our colleagues at the Blue Brain Project for their continuing collaboration on the development of the file formats for representing parameters of biophysically detailed networks.

Author Contributions

Conceptualization: Sergey L. Gratiy, Nicholas Cain, Christof Koch, Costas A. Anastassiou, Anton Arkhipov.

Funding acquisition: Christof Koch, Costas A. Anastassiou.

Investigation: Sergey L. Gratiy, Yazan N. Billeh, Kael Dai, Anton Arkhipov.

Methodology: Sergey L. Gratiy, Yazan N. Billeh, Kael Dai, David Feng, Nicholas Cain, Anton Arkhipov.

Project administration: Christof Koch.

Resources: Nathan W. Gouwens, Christof Koch.

Software: Sergey L. Gratiy, Yazan N. Billeh, Kael Dai, Catalin Mitelut, David Feng, Anton Arkhipov.

Supervision: Christof Koch, Costas A. Anastassiou, Anton Arkhipov.

Validation: Sergey L. Gratiy, Yazan N. Billeh, Kael Dai, Catalin Mitelut.

Visualization: Sergey L. Gratiy, Yazan N. Billeh, Catalin Mitelut, David Feng, Anton Arkhipov.

Writing – original draft: Sergey L. Gratiy, Yazan N. Billeh, Catalin Mitelut, Christof Koch, Anton Arkhipov.

Writing – review & editing: Sergey L. Gratiy, Yazan N. Billeh, Anton Arkhipov.

References

1. Amunts K, Ebell C, Muller J, Telefont M, Knoll A, Lippert T. The Human Brain Project: Creating a European Research Infrastructure to Decode the Human Brain. Neuron. 2016 Nov 2; 92(3):574–81. https://doi.org/10.1016/j.neuron.2016.10.046 PMID: 27809997

2. Hawrylycz M, Anastassiou C, Arkhipov A, Berg J, Buice M, Cain N, Gouwens NW, Gratiy S, Iyer R, Lee JH, Mihalas S, Mitelut C, Olsen S, Reid RC, Teeter C, Vries S de, Waters J, Zeng H, Koch C, MindScope Anastassiou C, Arkhipov A, Barber C, Becker L, Berg J, Berg B, Bernard A, Bertagnolli D, Bickley K, Bleckert A, Blesie N, Bodor A, Bohn P, Bowles N, Brouner K, Buice M, Bumbarger D, Cain N, Caldejon S, Casal L, Casper T, Cetin A, Chapin M, Chatterjee S, Cheng A, Costa N da, Cross S, Cuhaciyan C, Daigle T, Dang C, Danskin B, Desta T, Vries S de, Dee N, Denman D, Dolbeare T, Donimirski A, Dotson N, Durand S, Farrell C, Feng D, Fisher M, Fliss T, Garner A, Garrett M, Garwood M, Gaudreault N, Gilbert T, Gill H, Gliko O, Godfrey K, Goldy J, Gouwens N, Gratiy S, Gray L, Griffin F, Groblewski P, Gu H, Gu G, Habel C, Hadley K, Haradon Z, Harrington J, Harris J, Hawrylycz M, Henry A, Hejazinia N, Hill C, Hill D, Hirokawa K, Ho A, Howard R, Huffman J, Iyer R, Jarsky T, Johal J, Keenan T, Kim S, Knoblich U, Koch C, Kriedberg A, Kuan L, Lai F, Larsen R, Larsen R, Lau C, Ledochowitsch P, Lee B, Lee C-K, Lee J-H, Lee F, Li L, Li Y, Liu R, Liu X, Long B, Long F, Luviano J, Madisen L, Maldonado V, Mann R,

Mastan N, Melchor J, Menon V, Mills SMM, Mitelut C, Mizuseki K, Mortrud M, Ng L, Nguyen T, Nyhus J, Oh SW, Oldre A, Ollerenshaw D, Olsen S, Orlova N, Ouellette B, Parry S, Pendergraft J, Peng H, Perkins J, Phillips J, Potekhina L, Reading M, Reid C, Rogers B, Roll K, Rosen D, Saggau P, Sandman D, Shea-Brown E, Shai A, Shi S, Siegle J, Sjoquist N, Smith K, Sodt A, Soler-Llavina G, Sorensen S, Stoecklin M, Sunkin S, Szafer A, Tasic B, Taskin N, Teeter C, Thatra N, Thompson C, Tieu M, Tsyboulski D, Valley M, Wakeman W, Wang Q, Waters J, White C, Whitesell J, Williams D, Wong N, Wright V, Zhuang J, Yao Z, Young R, Youngstrom B, Zeng H, Zhou Z. Inferring cortical function in the mouse visual system through large-scale systems neuroscience. Proc Natl Acad Sci. 2016 Jul 5; 113 (27):7337–44. https://doi.org/10.1073/pnas.1512901113 PMID: 27382147

3. Koch C, Jones A. Big Science, Team Science, and Open Science for Neuroscience. Neuron. 2016; 92 (3):612–616. https://doi.org/10.1016/j.neuron.2016.10.019 PMID: 27810003
4. Martin CL, Chun M. The BRAIN Initiative: Building, Strengthening, and Sustaining. Neuron. 2016 Nov 2; 92(3):570–3. https://doi.org/10.1016/j.neuron.2016.10.039 PMID: 27809996
5. Traub RD, Contreras D, Cunningham MO, Murray H, LeBeau FEN, Roopun A, Bibbig A, Wilent WB, Higley MJ, Whittington MA. Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts. J Neurophysiol. 2005 Apr; 93(4):2194–232. https://doi.org/10.1152/jn.00983.2004 PMID: 15525801
6. Izhikevich EM, Edelman GM. Large-scale model of mammalian thalamocortical systems. Proc Natl Acad Sci. 2008 Mar 4; 105(9):3593–8. https://doi.org/10.1073/pnas.0712231105 PMID: 18292226
7. Markram H, Muller E, Ramaswamy S, Reimann MW, Abdellah M, Sanchez CA, Ailamaki A, Alonso-Nanclares L, Antille N, Arsever S, Kahou GAA, Berger TK, Bilgili A, Buncic N, Chalimourda A, Chindemi G, Courcol J-D, Delalondre F, Delattre V, Druckmann S, Dumusc R, Dynes J, Eilemann S, Gal E, Gevaert ME, Ghobril J-P, Gidon A, Graham JW, Gupta A, Haenel V, Hay E, Heinis T, Hernando JB, Hines M, Kanari L, Keller D, Kenyon J, Khazen G, Kim Y, King JG, Kisvarday Z, Kumbhar P, Lasserre S, Le Bé J-V, Magalhães BRC, Merchán-Pérez A, Meystre J, Morrice BR, Muller J, Muñoz-Céspedes A, Muralidhar S, Muthurasa K, Nachbaur D, Newton TH, Nolte M, Ovcharenko A, Palacios J, Pastor L, Perin R, Ranjan R, Riachi I, Rodríguez J-R, Riquelme JL, Rössert C, Sfyrakis K, Shi Y, Shillcock JC, Silberberg G, Silva R, Tauheed F, Telefont M, Toledo-Rodriguez M, Tränkler T, Van Geit W, Díaz JV, Walker R, Wang Y, Zaninetta SM, DeFelipe J, Hill SL, Segev I, Schürmann F. Reconstruction and Simulation of Neocortical Microcircuitry. Cell. 2015 Oct 8; 163(2):456–92. https://doi.org/10.1016/j.cell.2015.09.029 PMID: 26451489
8. Bezaire MJ, Raikov I, Burk K, Vyas D, Soltesz I. Interneuronal mechanisms of hippocampal theta oscillations in a full-scale model of the rodent CA1 circuit. eLife. 2016 Dec 23; 5:e18566. https://doi.org/10.7554/eLife.18566 PMID: 28009257
9. Arkhipov A, Gouwens NW, Billeh YN, Gratiy S, Iyer R, Wei Z, Xu Z, Berg J, Buice M, Cain N, Costa N da, Vries S de, Denman D, Durand S, Feng D, Jarsky T, Lecoq J, Lee B, Li L, Mihalas S, Ocker GK, Olsen SR, Reid RC, Soler-Llavina G, Sorensen SA, Wang Q, Waters J, Scanziani M, Koch C. Visual physiology of the Layer 4 cortical circuit in silico. bioRxiv. 2018 Mar 31;292839.
10. Hines ML, Carnevale NT. The NEURON Simulation Environment. Neural Comput. 1997 Aug 1; 9 (6):1179–209. PMID: 9248061
11. Carnevale NT, Hines ML. The NEURON Book. 1 edition. Cambridge University Press; 2009. 480 p.
12. Bower JM, Beeman D. The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System. Springer Science & Business Media; 2012.
13. Klijn W, Cumming B, Yates S, Karakasis V, Peyser A. Arbor: A morphologically detailed neural network simulator for modern high performance computer architectures. In 2017. Available from: https://github.com/eth-cscs/arbor
14. Migliore M, Cannia C, Lytton WW, Markram H, Hines ML. Parallel network simulations with NEURON. J Comput Neurosci. 2006 May 26; 21(2):119–29. https://doi.org/10.1007/s10827-006-7949-5 PMID: 16732488
15. Hines ML, Carnevale NT. Translating network models to parallel hardware in NEURON. J Neurosci Methods. 2008 Apr 30; 169(2):425–55. https://doi.org/10.1016/j.jneumeth.2007.09.010 PMID: 17997162
16. Hines ML, Davison AP, Muller E. NEURON and Python. Front Neuroinformatics. 2009 Jan 28; 3.
17. Gleeson P, Steuber V, Silver RA. neuroConstruct: a tool for modeling networks of neurons in 3D space. Neuron. 2007; 54(2):219–235. https://doi.org/10.1016/j.neuron.2007.03.025 PMID: 17442244
18. Davison A, Brüderle D, Kremkow J, Muller E, Pecevski D, Perrinet L, Yger P. PyNN: a common interface for neuronal network simulators. 2009;
19. Wang H-P, Spencer D, Fellous J-M, Sejnowski TJ. Synchrony of Thalamocortical Inputs Maximizes Cortical Reliability. Science. 2010 Apr 2; 328(5974):106–9. https://doi.org/10.1126/science.1183108 PMID: 20360111

20. Hines ML, Morse T, Migliore M, Carnevale NT, Shepherd GM. ModelDB: A Database to Support Computational Neuroscience. J Comput Neurosci. 2004 Jul 1; 17(1):7–11. https://doi.org/10.1023/B:JCNS.0000023869.22017.2e PMID: 15218350

21. Allen Cell Types Database [Internet]. 2015. Available from: http://celltypes.brain-map.org.

22. Gouwens N, Berg J, Kock C, Feng D, Sorensen S, Arkhipov A. Systematic, high-throughput generation of biophysically detailed models for diverse cortical neuron types. Nat Commun. 2018; 9(1):710. https://doi.org/10.1038/s41467-017-02718-3 PMID: 29459718

23. Teeter C, Iyer R, Menon V, Gouwens N, Feng D, Berg J, Cain N, Koch C, Mihalas S. Generalized Leaky Integrate-And-Fire Models Classify Multiple Neuron Types. Nat Commun. 2017 Jan 31; 9(1):709.

24. Gleeson P, Crook S, Cannon RC, Hines ML, Billings GO, Farinella M, Morse TM, Davison AP, Ray S, Bhalla US, others. NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail. PLoS Comput Biol. 2010; 6(6):e1000815. https://doi.org/10.1371/journal.pcbi.1000815 PMID: 20585541

25. Raikov I. NineML–a description language for spiking neuron network modeling: the abstraction layer. BMC Neurosci. 2010; 11(1):P66.

26. Ko H, Hofer SB, Pichler B, Buchanan KA, Sjöström PJ, Mrsic-Flogel TD. Functional specificity of local synaptic connections in neocortical networks. Nature. 2011 May; 473(7345):87. https://doi.org/10.1038/nature09880 PMID: 21478872

27. Cossell L, Iacaruso MF, Muir DR, Houlton R, Sader EN, Ko H, Hofer SB, Mrsic-Flogel TD. Functional organization of excitatory synaptic strength in primary visual cortex. Nature. 2015 Feb 19; 518 (7539):399–403. https://doi.org/10.1038/nature14182 PMID: 25652823

28. Lee W-CA, Bonin V, Reed M, Graham BJ, Hood G, Glattfelder K, Reid RC. Anatomy and function of an excitatory network in the visual cortex. Nature. 2016 Apr 21; 532(7599):370–4. https://doi.org/10.1038/nature17192 PMID: 27018655

29. Vinje WE, Gallant JL. Natural Stimulation of the Nonclassical Receptive Field Increases Information Transmission Efficiency in V1. J Neurosci. 2002 Apr 1; 22(7):2904–15. https://doi.org/20026216 PMID: 11923455

30. Lewicki MS. A review of methods for spike sorting: the detection and classification of neural action potentials. Netw Comput Neural Syst. 1998; 9(4):R53–R78.

31. Katzner S, Nauhaus I, Benucci A, Bonin V, Ringach DL, Carandini M. Local origin of field potentials in visual cortex. Neuron. 2009; 61(1):35–41. https://doi.org/10.1016/j.neuron.2008.11.016 PMID: 19146811

32. Lindén H, Tetzlaff T, Potjans TC, Pettersen KH, Grün S, Diesmann M, Einevoll GT. Modeling the spatial reach of the LFP. Neuron. 2011; 72(5):859–872. https://doi.org/10.1016/j.neuron.2011.11.006 PMID: 22153380

33. Buzsáki G. Large-scale recording of neuronal ensembles. Nat Neurosci. 2004; 7(5):446. https://doi.org/10.1038/nn1233 PMID: 15114356

34. Jun JJ, Steinmetz NA, Siegle JH, Denman DJ, Bauza M, Barbarits B, Lee AK, Anastassiou CA, Andrei A, Aydın Ç, Barbic M, Blanche TJ, Bonin V, Couto J, Dutta B, Gratiy SL, Gutnisky DA, Häusser M, Karsh B, Ledochowitsch P, Lopez CM, Mitelut C, Musa S, Okun M, Pachitariu M, Putzeys J, Rich PD, Rossant C, Sun W, Svoboda K, Carandini M, Harris KD, Koch C, O'Keefe J, Harris TD. Fully integrated silicon probes for high-density recording of neural activity. Nature. 2017 Nov; 551(7679):232. https://doi.org/10.1038/nature24636 PMID: 29120427

35. Buzsáki G, Anastassiou CA, Koch C. The origin of extracellular fields and currents—EEG, ECoG, LFP and spikes. Nat Rev Neurosci. 2012 Jun; 13(6):407–20. https://doi.org/10.1038/nrn3241 PMID: 22595786

36. Einevoll GT, Kayser C, Logothetis NK, Panzeri S. Modelling and analysis of local field potentials for studying the function of cortical circuits. Nat Rev Neurosci. 2013; 14(11):770. https://doi.org/10.1038/nrn3599 PMID: 24135696

37. Lindén H, Hagen E, Łęski S, Norheim ES, Pettersen KH, Einevoll GT. LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons. Front Neuroinformatics. 2014 Jan 16; 7.

38. Ness TV, Chintaluri C, Potworowski J, Łęski S, Głąbska H, Wójcik DK, Einevoll GT. Modelling and Analysis of Electrical Potentials Recorded in Microelectrode Arrays (MEAs). Neuroinformatics. 2015 Oct 1; 13(4):403–26. https://doi.org/10.1007/s12021-015-9265-6 PMID: 25822810

39. Hagen E, Ness TV, Khosrowshahi A, Sørensen C, Fyhn M, Hafting T, Franke F, Einevoll GT. ViSAPy: A Python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms. J Neurosci Methods. 2015 Apr 30; 245:182–204. https://doi.org/10.1016/j.jneumeth.2015.01.029 PMID: 25662445

40. Hagen E, Dahmen D, Stavrinou ML, Lindén H, Tetzlaff T, Albada V, J S, Grün S, Diesmann M, Einevoll GT. Hybrid Scheme for Modeling Local Field Potentials from Point-Neuron Networks. Cereb Cortex. 2016 Dec 1; 26(12):4461–96. https://doi.org/10.1093/cercor/bhw237 PMID: 27797828

41. Gewaltig M-O, Diesmann M. NEST (NEural Simulation Tool). Scholarpedia. 2007 Apr 5; 2(4):1430.

42. Plonsey R. The active fiber in a volume conductor. IEEE Trans Biomed Eng. 1974 Sep; BME-21 (5):371–81.

43. Holt GR. A critical reexamination of some assumptions and implications of cable theory in neurobiology. California Institute of Technology; 1998.

44. Holt GR, Koch C. Electrical Interactions via the Extracellular Potential Near Cell Bodies. J Comput Neurosci. 1999 Mar; 6(2):169–84. PMID: 10333161

45. Jun JJ, Mitelut C, Lai C, Gratiy S, Anastassiou C, Harris TD. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. bioRxiv. 2017 Jan 30;101030.

46. Codd EF. A relational model of data for large shared data banks. Commun ACM. 1970; 13(6):377–387.

47. Berendsen HJ, van der Spoel D, van Drunen R. GROMACS: a message-passing parallel molecular dynamics implementation. Comput Phys Commun. 1995; 91(1–3):43–56.

48. Humphrey W, Dalke A, Schulten K. VMD: visual molecular dynamics. J Mol Graph. 1996; 14(1):33–38. PMID: 8744570

49. Case DA, Cheatham TE, Darden T, Gohlke H, Luo R, Merz KM, Onufriev A, Simmerling C, Wang B, Woods RJ. The Amber biomolecular simulation programs. J Comput Chem. 2005; 26(16):1668–1688. https://doi.org/10.1002/jcc.20290 PMID: 16200636

50. Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel RD, Kalé L, Schulten K. Scalable molecular dynamics with NAMD. J Comput Chem. 2005 Dec 1; 26(16):1781–802. https://doi.org/10.1002/jcc.20289 PMID: 16222654

51. Brooks BR, Brooks CL, MacKerell AD, Nilsson L, Petrella RJ, Roux B, Won Y, Archontis G, Bartels C, Boresch S, Caflisch A, Caves L, Cui Q, Dinner AR, Feig M, Fischer S, Gao J, Hodoscek M, Im W, Kuczera K, Lazaridis T, Ma J, Ovchinnikov V, Paci E, Pastor RW, Post CB, Pu JZ, Schaefer M, Tidor B, Venable RM, Woodcock HL, Wu X, Yang W, York DM, Karplus M. CHARMM: The Biomolecular Simulation Program. J Comput Chem. 2009 Jul 30; 30(10):1545–614. https://doi.org/10.1002/jcc.21287 PMID: 19444816