Neuroscience

DendroTweaks: An interactive approach for unraveling dendritic dynamics

Reviewed Preprint
v2 • October 30, 2025
Revised by authors

Reviewed Preprint
v1 • November 11, 2024

Roman Makarov, Spyridon Chavlis, Panayiota Poirazi

Institute of Molecular Biology and Biotechnology (IMBB), Foundation for Research and Technology-Hellas (FORTH), Heraklion, Greece • Department of Biology, University of Crete, Heraklion, Greece

eLife Assessment

Computational simulation of neuron function depends on a collection of morphological properties and ion channel biophysics. This manuscript introduces DendroTweaks, a **valuable** web application and Python library that eases interactive exploration, development, and validation of single-neuron models in an easily installable and well-documented package. The authors provide a **convincing** demonstration that their software aids with building intuition and rapid prototyping of biophysical models of neurons, which improves the accessibility of dendritic simulation.

https://doi.org/10.7554/eLife.103324.2.sa3

Abstract

Neurons rely on the interplay between two critical components, dendritic morphology and ion channels, to transform synaptic inputs into a sequence of somatic spikes. Detailed biophysical models with active dendrites have been instrumental in exploring this interaction. However, such models can be challenging to understand and validate due to the large number of parameters involved. In this work, we introduce *DendroTweaks*, a toolbox designed to make detailed biophysical models with active dendrites more intuitive and more interactive. *DendroTweaks* features a web-based graphical interface, where users can explore single-cell neuronal models and adjust their morphological and biophysical parameters with real-time visual feedback. In particular, *DendroTweaks* focuses on subcellular properties, such as kinetics and distribution of ion channels, as well as the dynamics and placement of synaptic inputs. The toolbox supports various experimental protocols designed to illuminate how morpho-electric properties map to dendritic events and how these dendritic events shape neuronal output, thereby enhancing model validation. It helps users build high-level, modular model representations and includes a rich set of tools for parsing, generating, and standardizing commonly used neuronal data formats. Finally, it enables model simplification through a built-in morphology reduction algorithm, allowing users to export models for further use in faster, more interpretable networks. By combining extensive visualization capabilities and comprehensive data management functionality, *DendroTweaks* introduces a novel interactive approach for unraveling dendritic dynamics. This approach will accelerate

research on dendritic computations, their underlying mechanisms, and their fundamental role in brain function.

1 Introduction

Neurons are the most well-studied brain cells, known for their key role in processing and storing information. Information travels from neuron to neuron via synaptic connections, which are typically formed on dendrites. These extensive branching processes of a neuron actively shape and transform synaptic inputs on their way to the soma, endowing neurons with a wide range of input-output transformations. Since Rall's pioneering work on signal propagation within dendrites [Rall, 1959], our understanding of dendritic dynamics has expanded with the discovery of local regenerative events, such as Na^+ dendritic spikes [Spencer and Kandel, 1961, Golding and Spruston, 1998], and NMDA and Ca^{2+} plateau potentials [Schiller and Schiller, 2001, Llinas and Nicholson, 1971, Llinás and Hess, 1976]. These active properties of dendrites largely depend on the interaction between their branching morphology and ion channel composition. Multicompartmental biophysical models with active dendrites have been instrumental in exploring this relationship. However, the large number of parameters in such models complicates their interpretability, making them challenging to build, examine, and validate. In addition, the lack of standardization, along with their high computational complexity, makes these models less attractive for use in large-scale network simulations. As a result, state-of-the-art network models still consider dendrites as passive cables [Markram et al., 2015, Billeh et al., 2020], greatly underestimating their computational power [Tran-Van-Minh et al., 2015]. At the same time, with the advent of new techniques like genetic tracking of ion channels and high-resolution imaging of neuronal activity using voltage-sensitive dyes, multicompartmental biophysical modeling is experiencing its renaissance. Detailed biophysical models, although mathematically complex and computationally inefficient, are ideal for capturing and explaining data produced, for example, through simultaneous voltage and synaptic input imaging *in vivo*.

These challenges and demands highlight the growing need to make biophysical neuronal models more accessible. Model accessibility can be considered on two complementary levels. First, there is *conceptual* accessibility, which refers to how well we can understand the system being modeled (e.g., *If I change parameter A, will outcome X follow?*). Second, there is *implementational* accessibility, which concerns the model as a computational artifact (e.g., *Can I easily change parameter A and measure X?*). Accessibility at the conceptual level can be improved through interactive hypothesis testing. This approach implies converting complex models into interactive visualizations that provide real-time feedback on how changes in morpho-electric parameters affect neuronal behavior. In particular, such functionality would help clarify how the ion channel kinetics and distribution shape dendritic events and somatic output. To our understanding, this is one of the biggest gaps in current neuronal modeling software. Interactive visualizations would also enhance model validation, shifting the focus from somatic spiking to activity throughout the cell. Finally, this approach would help identify which dendritic properties are essential for neuronal function and which can be discarded, enabling the simplification of single-cell models and their integration into faster, more interpretable networks. Accessibility at the implementational level can be improved by developing high-level, simulator-agnostic model representations that are capable of capturing complex dendritic properties. Such representations should be modular, enabling users to easily switch between stimulation protocols, morphologies, or parameter sets. In addition, accessibility at this level involves adopting best practices for data management and standardization. Together, these approaches result in more reusable, easier-to-understand models that are customizable according to users' needs.

In this work, we introduce *DendroTweaks*, a toolbox designed to make detailed biophysical models with active dendrites more accessible at both conceptual and implementational levels. Building on existing methods, we have developed a comprehensive workflow for developing single-cell

models, including tuning their morphological and biophysical parameters, running stimulations, and analyzing the results. *DendroTweaks* is implemented as a Python package with an intuitive web-based graphical user interface (GUI). The GUI allows users to visually explore and fine-tune any parameter of the model, providing real-time feedback through interactive plots. The toolbox helps users build high-level, modular model representations that effectively capture the complex properties of active dendrites. Additionally, it includes a rich set of tools for parsing, generating, and standardizing commonly used neuronal data formats, facilitating interoperability with other neuronal modeling software. With *DendroTweaks*, users can better understand and control their models while exploring how dendritic properties shape neuronal activity. By making complex models more interpretable and interactive, *DendroTweaks* will advance research on the role of dendrites in brain function and deepen our understanding of these remarkable structures.

2 Results

2.1 Implementation and user interface

DendroTweaks is a Python toolbox designed for developing single-cell detailed biophysical models with active dendrites. It is inspired by the exploratory data analysis approach and allows any user, from naive to expert, to gain an in-depth understanding of the model through interactive visualization. Using the Model-View-Presenter (MVP) architecture, a single-cell neuronal model is presented through a web-based GUI built with the Bokeh library for data visualization [Bokeh Development Team, 2014]. The model provides a high-level simulator-agnostic representation of a cell, whereas the numerical simulation is delegated to an external simulator such as NEURON [Hines and Carnevale, 2001] or Jaxley [Deistler et al., 2024]. The toolbox is available as both a standalone Python package and a web-based application. The application GUI can be accessed via the online platform (*https://dendrotweaks.dendrites.gr*) or a locally hosted Bokeh server. The Python package for direct interaction with the software's core functionality is distributed via the Python Package Index (PyPI) (*https://pypi.org/project/dendrotweaks*).

The GUI is organized into three main components (**Figure 1**): (1) the left menu for file import and export operations, simulation control, and application settings, (2) the main workspace with interactive plots, and (3) the right menu with widgets and auxiliary plots. The main workspace contains top panels representing the cell and bottom panels representing monitors for the cell's activity. In the upper left corner, there is a morphology plot, where the uploaded morphology is rendered as a 2D projection of the cell. To the right of it, there is a graph representation of the cell's computational segments, where different parameters, such as channel density or synapse placement, can be visualized using a color code. The bottom panel can display time-dependent variables such as voltage, current, and input spike times. The right menu features widgets to manipulate cell morphology, spatial distributions and kinetics of ion channels and synapses, as well as the placement and parameters of virtual recording and stimulating electrodes.

DendroTweaks builds a high-level modular model representation relying on commonly used neuronal data formats. It accepts neuronal morphologies in SWC format and ion channel models as MOD files. Biophysical properties of the cell are stored in JSON format, while stimulation protocols are defined through a combination of JSON and CSV files. Model configurations can be exported to this modular format at any stage and reloaded later for further use. Additionally, models developed with *DendroTweaks* can be automatically converted into plain simulator code, enabling their smooth integration into larger network simulations.

The following sections will illustrate a potential workflow and provide a detailed description of the GUI elements and their functionality. We will begin by exploring and refining dendritic morphology and choosing the spatial discretization of the model. Next, we will explore the kinetics of ion channels and propose an automatic algorithm for standardizing such models. We will also

Figure 1

Graphical user interface (GUI).

A screenshot of the web-based GUI accessed via the Chrome browser. The interface is organized into three main components: the left menu, the main workspace, and the right menu. It supports both light and dark themes.

demonstrate how different distributions of these channels across the dendritic tree can be used to reproduce dendritic phenomena, such as sodium and calcium spikes. Then, we will demonstrate how synapses can be added to our models and explore how the kinetics and placement of synapses can affect input integration within dendrites. Next, we will use an automatic morphology reduction algorithm in order to obtain a simpler model while retaining the activity close to the original one. Finally, we will present a set of experimental protocols that have been integrated into *DendroTweaks* to facilitate model validation.

2.2 Exploring dendritic morphology

Developing a detailed biophysical neuronal model typically begins with the experimental reconstruction of a neuron's morphology. Our toolbox accepts morphology reconstructions in the widely used SWC file format, which represents a cell as a collection of connected 3D points (nodes), where each node is defined by its XYZ spatial coordinates and a radius. A rich database of these files is readily available at *https://neuromorpho.org* [Ascoli et al., 2007]. Moreover, online conversion from all reconstruction formats to SWC and standardization of existing SWC files is possible [Mehta et al., 2023]. *DendroTweaks* employs tree graphs as a universal representation of neuronal morphology across different levels of abstraction: geometry, topology, and spatial discretization (**Figure 2 A**). At the geometric level, nodes represent individual points in the reconstructed morphology. At the topological level, these nodes are grouped into sections, i.e., the parts between bifurcation points. Each section is further divided into segments, which are used in numerical simulations. These representations are linked together: each section contains pointers to its constituent points and segments, and vice versa. The toolbox features a custom module for working with such structures, enabling efficient manipulation of morphologies through operations such as insertion, removal, and translation of nodes and subtrees. Another level of abstraction is domains, i.e., collections of sections. A domain is a region of a neuron distinguished by its anatomical or functional properties. For example, a typical pyramidal cell has several domains: soma, axon, basal dendrites, and an apical dendrite. The apical dendrite can be further subdivided into trunk, oblique, and tuft dendrites.

Upon uploading a neuronal morphology in the GUI, it is rendered as a 2D (XY) projection of the cell in the main workspace on the top left Cell panel. This panel includes a slider for rotating the model around the Y-axis. For illustration, we use a realistic morphology of a Layer 2/3 (L2/3) pyramidal neuron of the mouse primary visual cortex [Park et al., 2019] (**Figure 2 B**). Users can navigate through the cell by selecting a section to visually inspect its parameters. This can be done by simply clicking on a section on the interactive plot or via a dropdown widget to select a specific section by its name. Navigation buttons are also available to select a parent, sibling, or child section. The parameters of the currently selected section are visualized in the right menu under the Morphology/Section tab with two plots (**Figure 2 C**). The upper plot displays the geometry of the section, i.e., the diameter as a function of the section's length. The lower plot shows a morphological or biophysical parameter selected by the user.

Computer simulations always involve approximating a continuous system as one that is discrete in space and time, which is also applied in neuronal modeling [Carnevale and Hines, 2006]. Discretization in time is regulated by the simulation time step, dt. Spatial discretization is achieved through segmentation. Each segment can be considered as an equivalent RC circuit representing a part of the membrane, with an associated set of differential equations to calculate voltage dynamics at a given point in space and time (**Figure 2 C**, middle). In our example section, the centers of each segment are shown as three circles (nseg = 3), equally distributed along the section's length according to the formula $(2i - 1)/2 \times \text{nseg}$, where i is an integer in the range [1, nseg]. The bottom panel (**Figure 2 C**, bottom) features a bar plot showing the value for a chosen parameter (i.e., a NEURON range variable) for each segment. We also introduce a Graph view with a tree graph representing all the segments of the given cell (**Figure 2 D**). This view serves three main purposes: (1) to visualize the distribution of different parameters along the dendritic tree using color code, (2) to select specific segments and update their parameters, and (3) to calculate

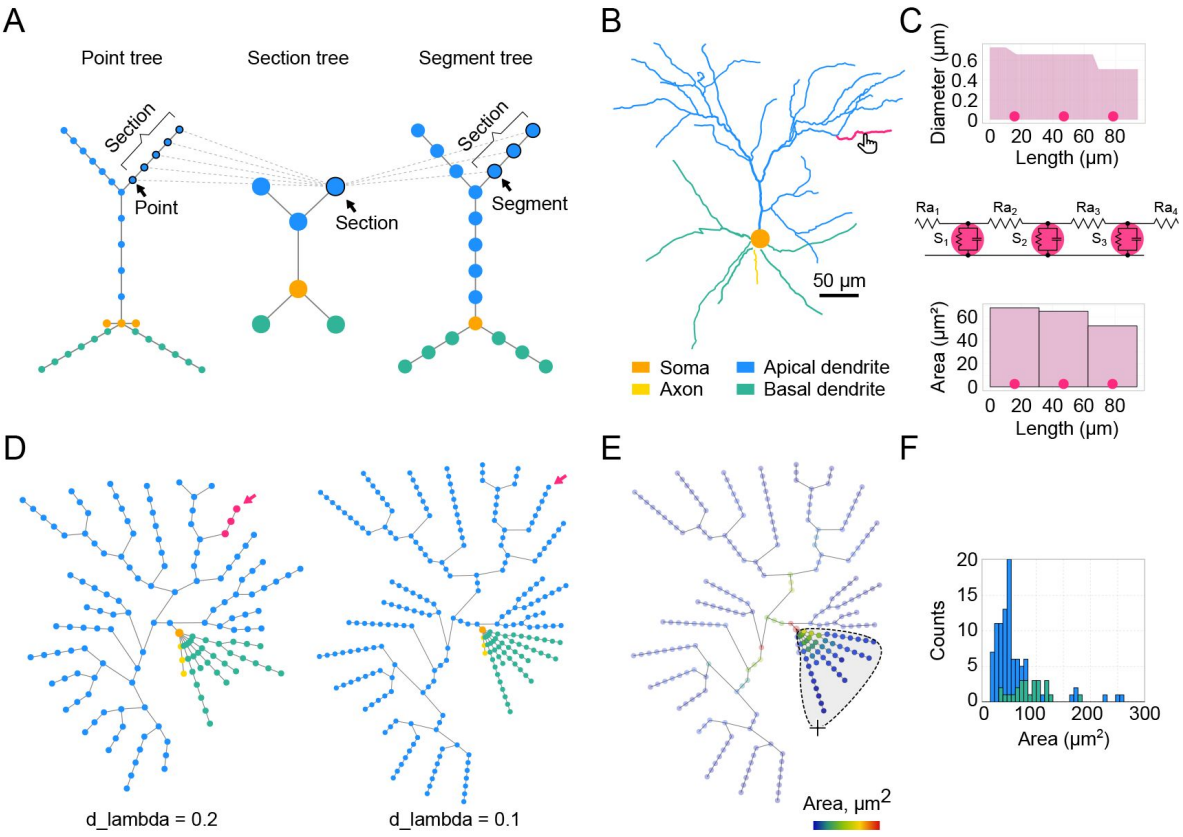

Figure 2

Dendritic morphology and segmentation.

(**A**) A schematic illustration of a neuronal morphology as three interconnected tree graphs. Colors indicate which morphological domains the nodes belong to. (**B**) Example of an L2/3 pyramidal neuron morphology from an SWC file. The selected section is highlighted in magenta. (**C**) Detailed representation of the selected section. Top: Diameter of the selected section as a function of the section's length, with circles marking segment centers. Middle: Equivalent circuit of the selected section shown as an RC circuit, assuming a passive membrane. Bottom: Bar plot showing values of a user-selected parameter (surface area, μm^2) for each segment. (**D**) Segmentation network graph representing the same cell as in (B) with d_lambda parameters of 0.2 (left) and 0.1 (right); nodes represent segments, colored as in (B). (**E**) Visualization of the selected morphological parameter on the segmentation graph using a color code. The lasso mouse tool is shown, which allows the selection of specific segments. Statistical morphometric analysis can be performed for the selected part of the cell. (**F**) Histogram of segment areas for basal (green) and apical (blue) segments.

statistics for a selected group of segments. By default, the graph plot uses color code to show the morphological domain to which a segment belongs (same colors as in the Cell plot). Rendering other parameters on the graph is discussed in the following sections. Since all parameters are ultimately set at the segment level, the segment graph offers the most detailed and accurate representation of the model. The granularity of the graph depends on the number of segments. In addition to defining the number of segments (nseg) for each individual section, it is possible to select the d_lambda parameter in the left menu that automatically assigns the number of segments for each section based on the fraction of the electrical length constant computed at the frequency of 100 Hz [Hines and Carnevale, 2001]. In the present example for the same neuron, we demonstrate a graph obtained with a d_lambda value 0.2 (**Figure 2 D**, left) and with the default value 0.1 (**Figure 2 C**, right). Note that segmentation using the d_lambda parameter also takes into account passive cable properties cm and Ra. Finally, it is worth mentioning that the spatial discretization defined here focuses exclusively on electrical properties, whereas chemical processes operate on fundamentally different spatial and temporal scales [Zador and Koch, 1994] and may require separate considerations when modeling phenomena like synaptic plasticity.

DendroTweaks supports basic morphometric analysis. Statistical analysis applies to an arbitrary subset of segments selected by the user on the interactive plot (**Figure 2 E**). This analysis includes calculating the number of sections, segments, and bifurcations, along with average diameter, length, and area, total surface area, and total length. The number of root and leaf dendrites is displayed independently of the selection. An auxiliary histogram plot (**Figure 2 F**) further aids in visualizing the distribution of any calculated parameter.

2.3 Standardization and tuning of ion channel models

In this section, we will start exploring the biophysical properties of our model at the individual ion channel level. Ion channels are crucial in shaping dendritic and somatic voltage dynamics, which are essential for neuronal communication and information processing. Since Hodgkin and Huxley's seminal work [Hodgkin and Huxley, 1952], mathematical models have been vital in understanding channel kinetics [Petousakis et al., 2023a]. The most widely used ion channel models today are written in the NMODL domain-specific language developed for the NEURON simulator [Hines and Carnevale, 2000]. *DendroTweaks* features a comprehensive NMODL-to-Python converter with a custom parser written in PyParsing (*https://pyparsing-docs.readthedocs.io*) and automatic Python code generation using Jinja templates (**Figure 3 A**). Upon importing a MOD file using the GUI, the kinetics of the corresponding channel can be visualized in the Biophys/Parameters tab of the right menu (**Figure 3 B**). The top plot shows the voltage-dependent steady-state values of the channel's gating variables, while the bottom plot shows the corresponding voltage-dependent time constants. When a channel is selected, interactive widgets for each parameter of the channel model appear in the right menu, allowing for the dynamic update of the plots.

Manual interaction with model parameters via the GUI is informative and provides good intuition about how each parameter affects voltage dynamics. However, it does not overcome the limitations of existing models. While valid and proven useful for exploring neuronal biophysics, many existing channel models exhibit inconsistencies and deviations from theoretical formulations, limiting their interpretability and reusability. Common issues include ambiguous variable names, inconsistent equations, hardcoded parameters, lack of units, incompatibility with the latest NEURON versions (such as problems with VERBATIM statements and dt), and potential overfitting to experimental data. These issues result in a steep learning curve for inexperienced modelers and significant hurdles for experienced ones, often resulting in a lack of proper exploratory analysis, laborious manual tuning of models, and simulation errors.

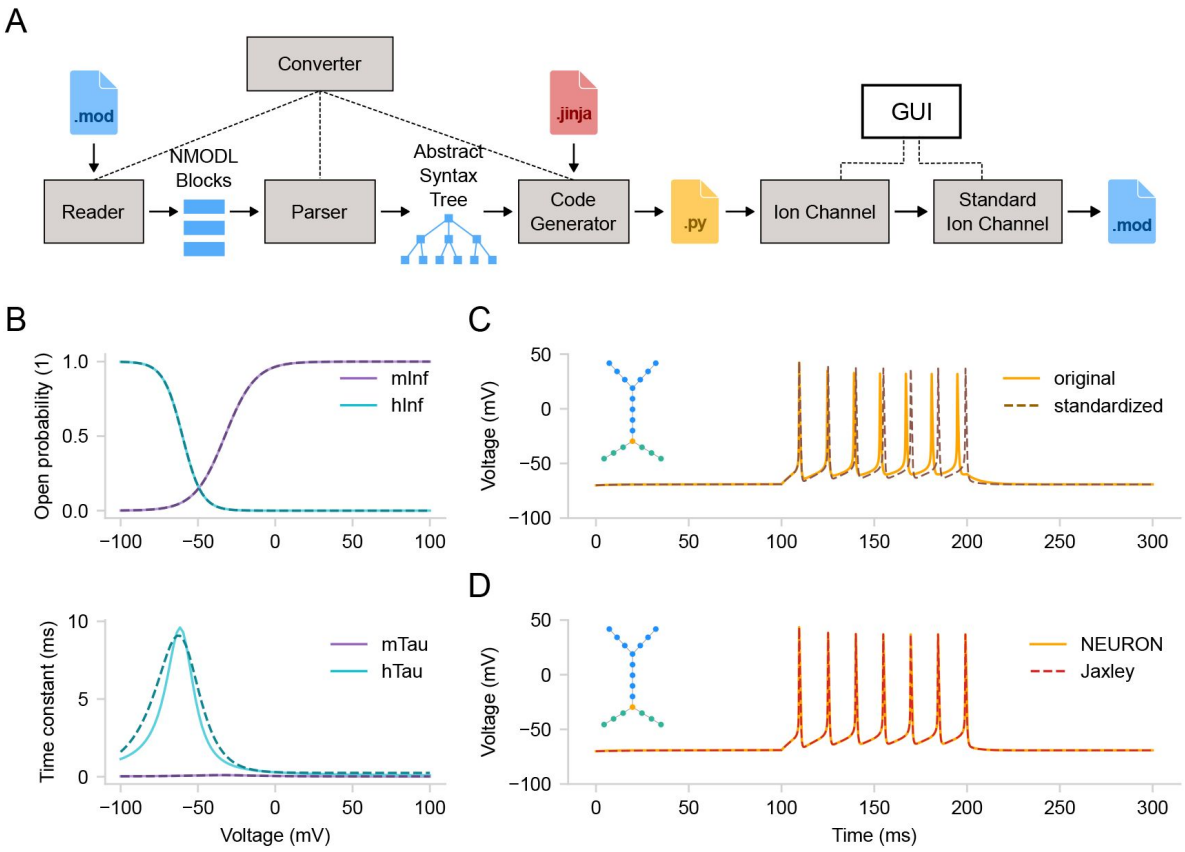

Figure 3

Standardization of ion channel models.

(**A**) Schematic of parsing and standardizing ion channel models from MOD files. The converter automatically extracts the information from a MOD file via parsing and generates a Python file containing an IonChannel class. An instance of this class can be used to visualize channel kinetics. Standardization algorithm produces a StandardIonChannel class instance. The standardized channel model can then be exported to a new MOD file. (**B**) Kinetics of a voltage-gated sodium channel. Activation (purple) and inactivation (teal) curves for the steady-state value (top) and the time constant (bottom). The solid lines represent the original model, while the dashed lines depict the model with standardized equations fitted to the original curves. (**C**) Corresponding somatic voltage traces from the original model (solid) and one with both sodium and potassium channels standardized (dashed). Inset - the segment graph of the model. (**D**) Voltage traces for the same model (with standardized channels) constructed in DendroTweaks and simulated either in NEURON (solid) or in Jaxley (dashed).

Several efforts have been made to ensure more efficient and consistent channel models. In 2010, Gleeson et al. [Gleeson et al., 2010] proposed standardizing models through NeuroML, an XML-based neuronal model description language, and manually converted models of voltage- and ligand-gated conductances using the ChannelML module. In 2017, Podlaski et al. [Podlaski et al., 2017] created a framework for the automated large-scale classification of ion channels, leading to the ICGenealogy web database (*https://icg.neurotheory.ox.ac.uk*), which categorizes new and existing models and experimental recordings. In 2019, Kumbhar et al. [Kumbhar et al., 2020] introduced a framework that parses existing MOD files to generate optimized code, significantly improving simulation speeds. Recently, a comprehensive database of the voltage-gated potassium channel (Kv) family has been made available through Channelpedia (*https://channelpedia.epfl.ch*) and extended to include other channel types in the Channelome project [Ranjan et al., 2011, 2024]. However, none of these approaches offered an automatic, visual-guided standardization of existing MOD files through the standardization of model equations. Here, we utilize a standardization approach for MOD files using a set of equations grounded in transition state theory (see Methods for details).

To demonstrate the standardization procedure, we parsed and inserted voltage-gated sodium and potassium channels and a leak channel into the cell's membrane. We then ran a simulation for 300 ms with a step current injection (amplitude 0.15 nA, delay 100 ms, duration 100 ms, temperature 37°C). Note that the parameters of the step-and-hold stimulation protocol (i.e., amplitude, delay, duration) can be adjusted using widgets in the Recordings and Stimuli/IClamps tab. For standardization, we selected the sodium channel, which has activation (m) and inactivation (h) state variables (**Figure 3 B**), as well as the potassium channel, which has an activation (n) state variable (not shown). The standardization algorithm produces plots of the original (solid) and fitted (dashed) curves for steady-state values and time constants of the state variables. Each state variable in the standardized model has five parameters: v half, sigma, k, delta, and tau 0 (see Methods for details). Besides the activation curves, users can compare cell activity before and after standardization (**Figure 3 C**). To achieve this, there is an option to "freeze" the original current and voltage traces for comparison before running the standardization algorithm.

Although NEURON is the default simulator used in *DendroTweaks* and it was employed throughout this study, the toolbox is designed to be adaptable to other simulators. As a demonstration, we applied the same modeling interface to the recently introduced Jaxley simulator [Deistler et al., 2024]. A key advantage of this process was that *DendroTweaks*'s NMODL-to-Python converter significantly reduced the need to manually re-implement existing ion channel models for Jaxley. By modifying the Jinja template, we were able to generate Jaxley-compatible classes that supported full numerical simulations, in addition to visualization of the channel's kinetic properties. *DendroTweaks* is capable of running simulations of multicompartmental models with multiple ion channels in Jaxley. To validate this, we ran simulations of the same simplified multicompartmental model, with sodium, potassium, and leak channels, in both NEURON and Jaxley (see **Figure 3, D**). These results highlight that *DendroTweaks* offers a unified modeling interface that can be integrated with simulators beyond NEURON in order to benefit from the additional functionality they provide.

2.4 Distributing ion channels

Having explored and refined the neuronal morphology and individual ion channels, a user can proceed to set up the biophysical properties of their neuronal model, including the densities and distributions of the various ion channel conductances. Dendrites of most neuron types are known to express some types of active ion channels [Johnston and Narayanan, 2008], including major ion channel families such as Nav, Cav, Kv, KCa, and HCN [Yu et al., 2005]. The distribution of each channel type is cell type-specific and is often non-uniform. For example, CaV1.x channels, also known as high-voltage activated L-type channels, are densely populated in proximal dendrites [Reuveni et al., 1993, Westenbroek et al., 1990]. In contrast, CaV3.x channels, or low-voltage activated T-type channels, increase in density with distance from the soma [Magee and Johnston,

1995, Magee et al., 1995], forming "hot-spots" for dendritic Ca^{2+} spikes in the apical dendrite [Reuveni et al., 1993, Yuste et al., 1994]. Another compelling example is the hyperpolarization-activated mixed cation current (Ih), mediated by HCN channels, which are higher in density in distal apical dendrites of CA1 [Magee, 1998, Lörincz et al., 2002] and cortical Layer 5 (L5) [Kole et al., 2006] pyramidal neurons. Even passive conductances in dendrites can show non-uniform distribution [Stuart and Spruston, 1998]. Additionally, certain kinetic properties of channels, such as half-maximal voltage, can also vary with distance from the soma [Hoffman et al., 1997, Migliore et al., 1999, Poolos et al., 2002]. While the distribution of dendritic ion channels has been the subject of significant research, there is still much to learn about their role in neuronal integrative function [Migliore and Shepherd, 2002, Johnston and Narayanan, 2008, Nusser, 2009, Shah et al., 2010]. Towards this goal, it is crucial to develop convenient tools for organizing ion channel distribution when designing software for studying neurons.

To make the process of exploring and adjusting the distributions of membrane mechanisms more user-friendly and intuitive, we utilize the graph view introduced earlier. Every membrane mechanism (distributed mechanism in NEURON) can be visualized on the graph, such that its value for a given segment is color-coded. To distribute parameters across the cell, users need to specify *where* and *how* a given parameter will be distributed. To select the segments *where* a given distribution will be applied, one needs to use segment groups. A segment group is a collection of segments that meet certain criteria (**Figure 4 A**). To create a new SegmentGroup, one must specify both a criterion and the domains where to search for matching segments. The criterion can be one of the following types: diameter, absolute distance (to the root of the tree), or relative distance within a domain. To define *how* the parameter will be distributed, one must use distribution functions. A distribution function takes a segment's distance from the soma as input and returns the parameter value at that distance (**Figure 4 B**). There are several built-in types of distributions (i.e., constant, linear, exponential, sigmoidal, etc.) which, when combined, can cover most of the existing models. Parameters of a given distribution are controlled by the widgets on the Biophys/Parameters tab. Note that, unlike domains, segment groups can overlap, allowing segments to belong to multiple groups simultaneously, and therefore can be thought of as layers. In other words, the order of groups is important: the parameters will be assigned only from the top-most group that the segment belongs to. Moreover, a segment group can encompass multiple domains or divide sections, such that different segments within the same section may belong to different groups (**Figure 4 C**).

The biophysical configuration can be exported in a JSON format that follows the structure described above. To keep the representation both compact and capable of representing complex spatial distributions, model parameters are stored using a unified, domain-based approach, where each distribution can be described with only a few coefficients (see Methods for details). As a result, the same biophysical configuration can be easily applied across multiple morphologies that have the same domains.

To illustrate the effectiveness of this interface in distributing dendritic mechanisms and investigating their impact on neuronal activity, we replicated several established models. First, we reproduced the L2/3 pyramidal neuronal model from Park et al., 2019 [Park et al., 2019] (**Figure 4 D**), to show dendritic sodium-driven backpropagation-activated action potentials (BAPs, **Figure 4 G** and **J**, left). Using this model, we were able to demonstrate that the distribution of sodium channels affects dendritic BAPs, as removing sodium channels from a specific branch prevented spike generation in that branch (**Figure 4 J**, right). Next, we replicated the CA1 pyramidal neuronal model from Poirazi et al., 2003 [Poirazi et al., 2003] to demonstrate the effect of the experimentally observed exponential distribution of HCN channels (**Figure 4 E, H**). As observed in the original study, we noted a significant depolarizing voltage sag, which was eliminated by blocking HCN channels (**Figure 4 K**). Finally, we replicated the model of an L5 pyramidal neuron from Hay et al., 2011 [Hay et al., 2011] to demonstrate dendritic Ca^{2+} spikes. We were able to evoke a dendritic Ca^{2+} plateau potential by providing

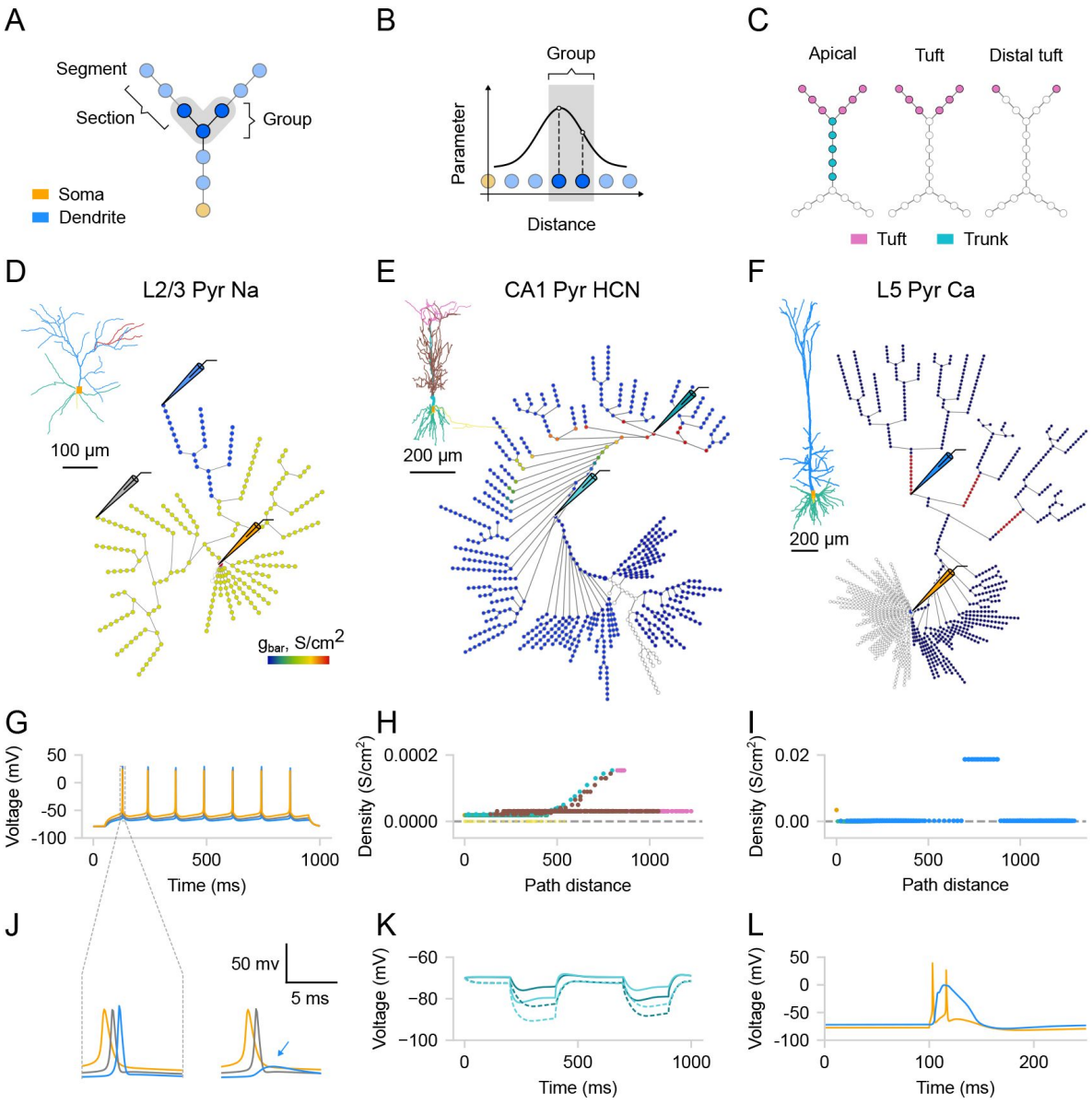

Figure 4

Distributions of ion channels.

(**A**) Schematic showing a segment group in a ball-and-stick model, satisfying specific distance criteria. (**B**) Schematic showing distribution as a function of distance from the soma; values are assigned only to segments matching the group's criteria. (**C**) Schematic showing the difference between domains and segment groups. A group can span multiple domains (left), match a single domain (middle), or partially include individual sections (right). (**D**) Example of a constant distribution for sodium channels, where maximal conductance in the selected region (dark blue) was decreased by 60%. Schematic electrodes indicate recording positions (inset: original morphology [Park et al., 2019]). (**E**) Example of an exponential distribution for the HCN channels (inset: original morphology [Poirazi et al., 2003]). (**F**) Example of a calcium "hot spot" (red) (inset: original morphology [Hay et al., 2011]). (**G**) Sodium-driven backpropagation-activated action potentials (BAPs) evoked by a 0.162 nA somatic current injection. (**H**) Distribution of maximal HCN channel conductance as a function of distance from the soma (see functional effect in K). (**I**) Distribution of maximal calcium channel conductance as a function of distance from the soma (see functional effect in L). (**J**) Expanded time scale for the two scenarios in (D), showing failure of BAP initiation (blue arrow) in the region with reduced sodium conductance. (**K**) Voltage sag produced by HCN channels. A current step (-0.2 nA, 200 ms) is injected proximally (light cyan) and, after 300 ms, distally (dark cyan) in the apical trunk. Dashed traces: blocking HCN channels, modeled as 80% reduction in channel conductance. (**L**) Dendritic calcium plateau potential triggered by synaptic input at the calcium "hot-spot" coincident with somatic current injection, leading to somatic bursting. Somatic traces are shown in orange, dendritic—in blue, cyan and gray.

coincident inputs at the soma and the Ca^{2+} "hot-spot" of the apical dendrite (**Figure 4 F, I**). This plateau potential then propagated to the soma, resulting in somatic burst firing (**Figure 4 L**). Overall, by replicating these models and respective simulations, we demonstrate how *DendroTweaks* can aid in investigating the role of ion channels in generating dendritic events through interactive parameter adjustment and visualization.

2.5 Distributing synapses

All previous examples used simple stimuli such as somatic and dendritic step current injections to demonstrate the functionalities of *DendroTweaks*. In this section, we describe how more realistic synaptic stimulation protocols can be implemented. Synaptic placement and timing play an important role in the integration of synaptic inputs. For example, neocortical pyramidal neurons respond supralinearly to spatially clustered inputs and sublinearly to randomly distributed ones [Polsky et al., 2004, Losonczy and Magee, 2006, Takahashi et al., 2012, Poirazi et al., 2003, Tran-Van-Minh et al., 2015]. Another layer of complexity to synaptic integration is added by the interplay between excitatory and inhibitory inputs [Doron et al., 2017, Du et al., 2017]. Interestingly, connections from different types of inhibitory interneurons target different dendritic domains of pyramidal neurons [Markram et al., 2004, Van Versendaal and Levelt, 2016], allowing for the selective regulation of information streams within a neuron. In light of the above, the ability to reproduce various synaptic input patterns is crucial for understanding dendritic integration.

As we do for ion channels, we use the graph view to visualize and allocate groups of synaptic inputs. While the pre-synaptic inputs are not modeled explicitly, we can create populations of "virtual" neurons projecting to our model. To create a Population in the GUI, users must select segments (**Figure 5 A**) and define population parameters using the dedicated widgets in the right menu. The creation of a population requires users to select a synapse type and specify the number of synapses to be uniformly distributed randomly within the selected segments. *DendroTweaks* offers three built-in synapse types: AMPA, NMDA, and $GABA_A$, with an additional option for a combined AMPA-NMDA synapse. Users can create as many populations as necessary. Each population can have unique kinetic parameters for the synapses (maximal conductance, g_max, equilibrium potential, e, time constants tau_rise, tau_decay) as well as parameters for incoming inputs (input rate, randomness/noise, onset, duration).

To demonstrate the power of *DendroTweaks* for exploring dendritic integration of synaptic inputs, we conducted *in silico* experiments involving different placements and activation times of synaptic inputs. First, we examined the effect of input synchronicity and NMDA synapses on generating NMDA spikes. Using the graph view, we distributed 20 excitatory AMPA-NMDA synapses within a single section of the Hay et al., 2011 passive model [Hay et al., 2011] (**Figure 5 A**). We observed that simultaneous activation of synapses or activation with a Poisson spike train produced distinct dendritic voltage responses. Additionally, by blocking NMDA conductances, we were able to eliminate NMDA spikes (**Figure 5 B**). Second, we replicated the effect of inhibition on dendritic NMDA spike generation, as shown in Doron et al., 2017 [Doron et al., 2017]. We added one inhibitory $GABA_A$ synapse in the same section and varied its activation time or location (**Figure 5 C**). Consistent with the original study, NMDA spikes could not be recovered if inhibition occurred 20 ms after excitation. Moreover, proximal inhibition had little effect on NMDA spikes, whereas distal inhibition significantly reduced them. Finally, we conducted an experiment similar to that described in Poirazi et al., 2003 [Poirazi et al., 2003], using the CA1 pyramidal neuronal model with active dendritic mechanisms. Using the graph view, we distributed 40 excitatory AMPA-NMDA synapses in two scenarios: either spread across multiple terminal branches of the apical dendrite (**Figure 5 D**) or clustered within five randomly selected branches of the apical dendrite (**Figure 5 F**). As in the original study, we found that distributed synapses failed to evoke high-frequency somatic activity (**Figure 5 E**), whereas clustering

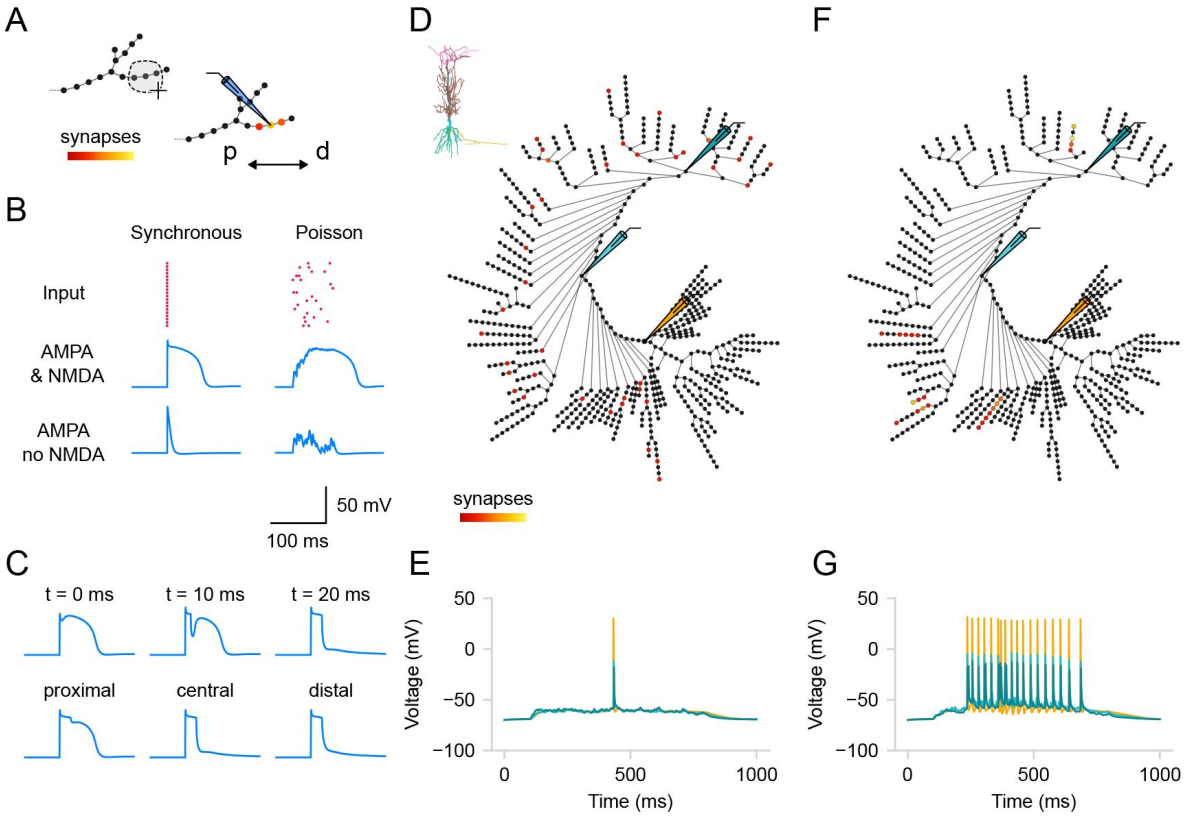

Figure 5

Kinetics and distribution of synapses.

(**A**) Schematic representation of distributing synaptic inputs. Three central segments of a distal apical branch are selected using the lasso tool, and synapses are added (p—proximal, d—distal). (**B**) Example responses evoked by activating 20 excitatory synapses placed within one branch as in (A). The regularity of inputs varies from synchronous activation to a random Poisson spike train. Note that the raster plot for input times is accessible in one of the workspace tabs. The examples demonstrate dendritic voltage responses in the presence or absence of NMDA conductances. (**C**) Experiment similar to Doron et al., 2017 [Doron et al., 2017], demonstrating the effect of inhibiting NMDA spikes. Top: One inhibitory $GABA_A$ synapse is placed in the middle of the section, and its activation time varies as 0, 10, and 20 ms after excitatory synapse activation. Bottom: The synapse location varies from the most proximal to the most distal segment of the section, with the activation time kept at 20 ms. The same stimulation protocol as in (B) with synchronous activation is used for the excitatory inputs. Scale is the same as in (B). (**D**) Distributed placement of 40 excitatory AMPA-NMDA synapses across the dendritic tree, similar to Poirazi et al., 2003 (inset: original morphology). (**E**) Somatic and dendritic voltage responses to distributed synaptic inputs (D) (25 Hz, Poisson-distributed), which nearly fail to evoke somatic action potentials. Compare with (G) for clustered inputs. (**F**) Clustered placement of the same 40 excitatory synapses from (D) within five randomly selected branches. (**G**) Somatic and dendritic voltage responses to clustered synaptic inputs (F), demonstrating robust somatic firing activity. Somatic traces are shown in orange, dendritic - in blue and cyan.

synapses made high-frequency somatic activity possible (**Figure 5 G**). These experiments illustrate how *DendroTweaks* can facilitate the investigation of synaptic input integration and its effects at the dendritic and somatic levels.

2.6 Reducing morphology

The explorations described in the previous sections aimed to enhance the user's understanding of which dendritic properties are essential for specific neuronal input-output transformations. With this knowledge, one can then proceed to simplify models using a built-in morphology reduction algorithm and export them for further use in faster and more interpretable neuronal networks.

Here, we follow the analytical impedance-based approach proposed by Amsalem et al., 2020 (neuron_reduce) [Amsalem et al., 2020]. This method maps a detailed dendritic tree to an equivalent cylinder with the same passive properties (i.e., specific membrane resistivity, capacitance, and axial resistivity). The transfer impedance from the distal, sealed end to the soma in the simplified model matches the transfer impedance from the most distal dendritic tip to the soma in the detailed model. Additionally, the input impedance at the proximal end matches that of the respective detailed dendrite when decoupled from the soma. In the original implementation, the entire subtree of each stem dendrite (e.g., the entire apical subtree) is mapped to a single corresponding cylinder. This, however, can impose some limitations on accurately capturing the complex branching patterns and electrotonic properties of the dendritic tree, potentially affecting the precision of simulations of synaptic integration and signal propagation.

In *DendroTweaks*, we extended the functionality of neuron_reduce to allow for a continuum of morphology reduction levels, bridging detailed and "ball-and-stick"-like models. Users can select any section of the cell and reduce its subtree, which allows for any intermediate level of detail to be achieved. As an example, we start with the Hay et al., 2011 [Hay et al., 2011] model used in the original study (**Figure 6 A**). We reduce its morphology using the original algorithm to the "ball-and-stick" level (**Figure 6 C**) and to an intermediate level where some apical oblique and tuft dendrites are preserved (**Figure 6 B**). Notably, the response of the partially reduced model (**Figure 6 E**) is closer to the original model's response (**Figure 6 D**) in terms of the number of spikes compared to the response of the fully reduced model (**Figure 6 F**). The reduction algorithm operates on passive morphologies and assigns active channel distributions *post hoc* by averaging the values of all original segments that map to a given segment of the reduced model. To enable the export of reduced models in *DendroTweaks*'s modular format, as well as in plain simulator code, we employed fitting the parameters of a distribution function to the resulting distribution of values. This approach provided a compact representation, requiring only a few parameters to be stored in order to reproduce a distribution. To demonstrate how these exported models can be integrated into larger simulations, we implemented a "toy" network model outside the toolbox using the reduced neurons (available in the examples directory of the package).

By integrating an enhanced version of neuron_reduce into *DendroTweaks*, we ensure easier post-reduction fine-tuning of model parameters. The graph view allows users to visualize the resulting distributions of channels and synapses after they have been mapped onto the reduced morphology. The simplified model can be re-validated to ensure it faithfully reproduces experimental observations. In the next section, we will discuss several built-in validation protocols that can be used for both original and simplified models.

2.7 Validating biophysical properties

Thus far, we have presented a comprehensive set of tools available in *DendroTweaks* for developing and exploring the parameters of multicompartmental single-cell biophysical models. In addition to these functionalities, *DendroTweaks* also offers some built-in validation protocols that allow users to ensure the resulting models align with experimental observations. This approach is semi-automated, requiring users to manually implement a stimulation protocol by setting the

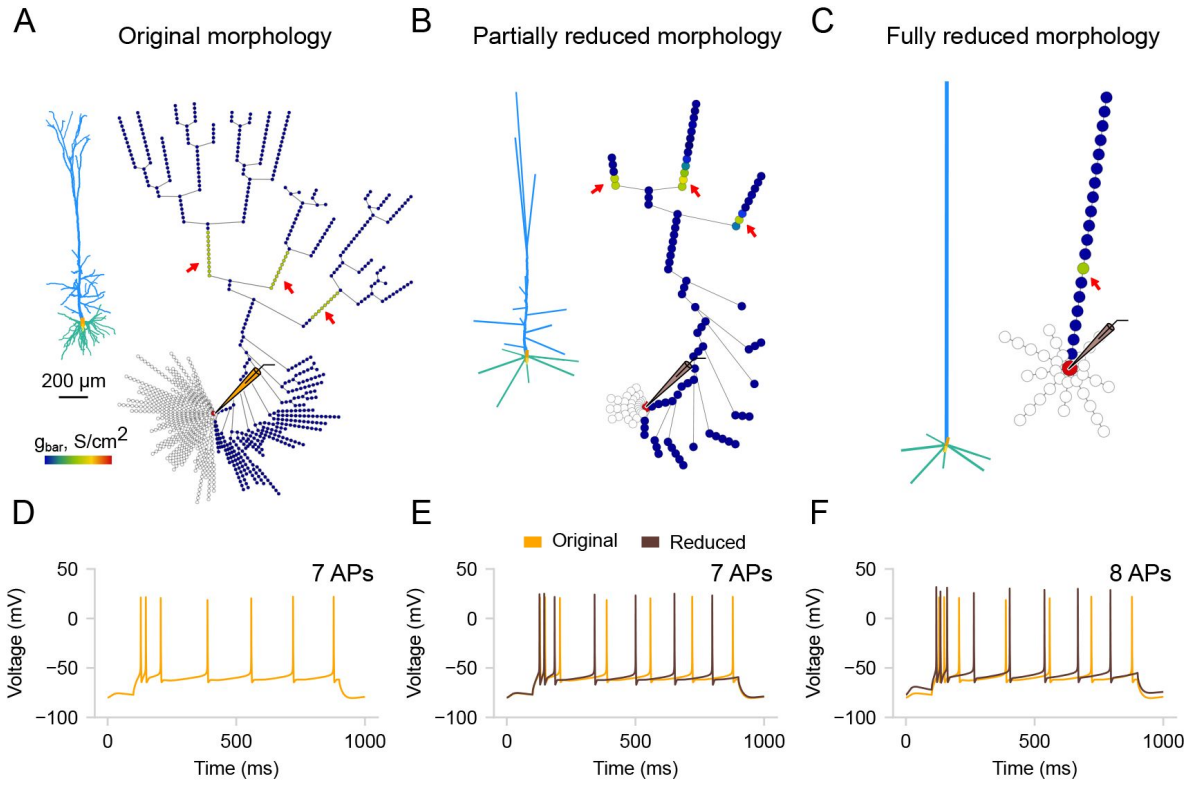

Figure 6

Morphology reduction.

(**A**) Original morphology of L5 pyramidal neuron [Hay et al., 2011] and its segmentation graph showing the distribution of calcium channels. The red arrows indicate the ”hot-spots” with increased channel density. (**B**) Partially reduced morphology using the extended version of neuron_reduce. The extended version allows for the reduction of any selected branch, allowing to retain more apical branches, in contrast to (C). (**C**) Fully reduced morphology. All stem dendrites (children of the soma) are reduced to a single equivalent cylinder. (**D-F**) Voltage response of the three models to somatic current injection of 0.5 nA. Note the difference in the number of dendritic ”hot-spots” and somatic APs between the three variations of the model. The partially reduced model more accurately reproduces the channel distribution and voltage response of the original model compared to the fully reduced one.

stimulation parameters. For example, to validate somatic action potentials, a user must apply a positive step current injection at the soma to produce somatic firing. On the Recordings and Stimuli/Analysis tab of the right menu, selecting the Somatic spikes command will automatically detect action potentials and measure their properties, such as firing rate, amplitude, and half-width. While this process is not fully automated, it allows for the use of custom stimulation protocols instead of relying on predefined stimuli parameters, thereby offering more flexibility.

We demonstrate validation of passive and active properties using built-in protocols applied to the Hay et al., 2011 model [Hay et al., 2011]. First, we measured the input resistance (74 MΩ) and the membrane time constant (34 ms) (**Figure 7 A**) by applying a step current injection (-0.05 nA) at the soma while temporally blocking the HCN channels ($\bar{g}_h = 0$) [Stuart and Spruston, 1998, Golding et al., 2005]. We then restored the HCN channel conductance to its original values and measured voltage attenuation for either somatic (-0.5 nA, **Figure 7 B**) or dendritic (-0.05 nA, **Figure 7 C**) step current injection. Next, we stimulated the soma with a positive step current (0.793 nA) and detected somatic action potentials (**Figure 7 D**). From the same trace, we measured the peaks, amplitudes, and half-widths of individual action potentials (**Figure 7 E**). We then constructed a somatic f-I curve (**Figure 7 F**) by applying current steps of increasing amplitude (0.1 nA step). After validating the somatic activity, we evaluated dendritic integration nonlinearity (**Figure 7 G**, left) by comparing measured individual waveforms (**Figure 7 G**, right) to the expected linear summation of postsynaptic potentials (PSPs). Note that protocols for both the somatic f-I curve and dendritic integration curve require multiple simulation runs with varying stimulus intensity, which are done automatically after the user specifies the initial stimulation parameters. Finally, since the model has HCN channels, we measured the voltage sag ratio (0.14) and the steady-state input resistance (40 MΩ) by applying a negative step current injection (-0.05 nA) at the soma (**Figure 7 H**). A 300 ms period before the simulations (not shown) was used to stabilize the resting membrane potential. For a detailed description of the protocols, see **Table 1** in Methods.

3 Discussion

3.1 Conceptual and implementational accessibility

We developed *DendroTweaks* to make detailed biophysical models with active dendrites more accessible at the conceptual and implementational levels. At the conceptual level, our main motivation was to illuminate how morpho-electric properties of dendrites shape neuronal activity and computations. We deliberately focused on single-cell models to provide comprehensive functionality for tuning subcellular properties, including dendritic morphology, ion channel kinetics and distributions, as well as synaptic inputs. With *DendroTweaks*' interactive interface, users can better understand how these subcellular properties influence dendritic activity and neuronal output in their models. We equip neuronal models with widgets and interactive plots, making every parameter visually accessible and interactively tunable. Importantly, for simple models, plots respond to user actions in real time, ensuring a smooth model exploration and tuning process. Interactive plots illustrating neuronal morphology significantly simplify the process of navigating through various sections and segments of the model. We enhance the interactivity of ion channel models by providing visualization of channel kinetics based on MOD files. Furthermore, by representing a cell as a graph with computational segments as nodes, we simplify and visually enhance the process of distributing ion channels and synapses throughout the cell. Finally, we extend the neuron_reduce approach for morphology reduction to encompass all potential reduction levels from full morphology to "ball-and-stick"-like models and incorporate this method into our graphical interface, thereby taking advantage of inherent visualization and validation capabilities.

Validation protocol	Readout	Minimal set of membrane parameters	Recordings	Stimuli
Input resistance	R_{in}, MΩ	cm, Ra and Leak	Any single segment	Negative step current injection
Membrane time constant	τ, ms	cm, Ra and Leak	Any single segment	Negative step current injection
Rheobase current	I_{rh}, nA	At least Na and Kdr	Soma	Positive step current injection
Somatic spikes	AP times, amplitude, half-width, rate, ISI	At least Na and Kdr	Soma	Positive step current injection above the rheobase value
Somatic f/I curve	$\text{Rate}(I_{\text{ext}}), Hz$	At least Na and Kdr	Soma	Positive step current injection (amplitude automatically increased)
Dendritic nonlinearity	EPSP, mV	Dendritic ion channels	Dendritic segment	Excitatory synapse (weight automatically increased)
Voltage attenuation	$\Delta V_0/\Delta V_i$, unitless	cm, Ra and Leak	At least two segments	Negative step current injection
Sag ratio	$\frac{V_{\text{end}} - V_{\text{min}}}{V_{\text{start}} - V_{\text{min}}}$	HCN channels (h current)	Any single segment	Negative step current injection

Table 1

Validation protocols

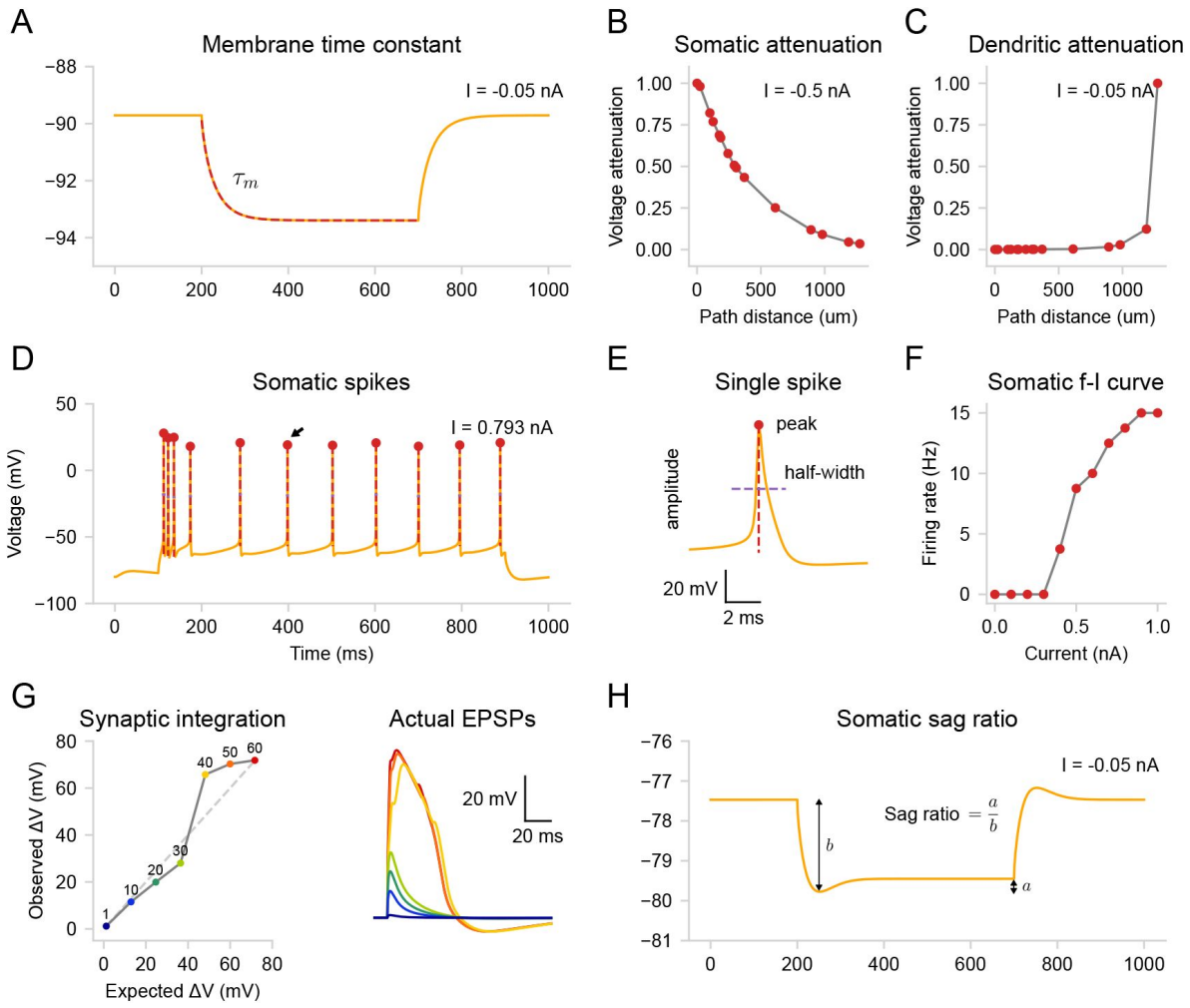

Figure 7

Validation protocols.

Built-in validation protocols applied to the Hay et al., 2011 model [Hay et al., 2011]; See also **Table 1**). (**A**) Membrane time constant (34 ms) measured by applying a step current injection (-0.05 nA) at the soma, while blocking the HCN channels. (**B**) Voltage attenuation for somatic (-0.5 nA, left) and dendritic (-0.05 nA, right) step current injection at all bifurcation points along the path from a selected tip segment. (**C**) Detected somatic action potentials from stimulation with a positive step current (0.793 nA). (**D**) Single action potential indicated with an arrow in (C), with measured peak, amplitude, and half-width values. (**E**) Somatic frequency-current (f-I) curve constructed by applying current steps of increasing amplitude (0.1 nA step) at the soma. (**F**) Nonlinear integration of synaptic inputs in a tuft dendrite. Left: Expected vs. actual EPSP amplitude for 1 to 60 synchronously activated AMPA-NMDA synapses. Right: Actual EPSP waveforms. (**G**) Voltage sag ratio at the soma measured by applying a negative step current injection (-0.05 nA). Note that unlike in (A), the hyperpolarization-activated current through the HCN channels is present here.

At the implementational level, our goal was to provide a high-level modeling framework that abstracts away technical implementation details and produces structured, transparent, and interoperable model representations. We provide users with a unified modeling interface that allows them to construct high-level simulator-independent model representations while automatically generating simulator-specific model instances. The underlying model structure is designed to be modular, clearly separating morphology, biophysical configurations, and stimulation protocols. To comprehensively capture complex dendritic properties, we introduced a custom format for biophysical parameters, capable of capturing non-uniform ion channel distributions across different dendritic domains. To enhance data management, we introduced versatile tools for parsing, generating, and standardizing commonly used neuronal data formats. In particular, we developed rich functionality for processing ion channel models written in the NMODL language, making them more comprehensible and reusable. To ensure seamless integration into broader modeling workflows, we implemented a range of export options, allowing users to reuse individual model components or entire models across different modeling software. Through this combination of advanced model construction capabilities, comprehensive data handling, and broad interoperability, *DendroTweaks* complements existing tools and fills an important niche within the neuronal modeling ecosystem. An integrated modeling workflow might include multiple steps and platforms. For example, users can first set up a model in *DendroTweaks* by removing morphological artifacts and standardizing ion channel models. They can then establish meaningful initial parameters and use external tools for automated parameter optimization. Subsequently, users can re-import the model for visual exploration, validation, or morphology reduction. Finally, they can export these refined models and use specialized software to scale them up into larger network simulations. In the following paragraphs, we compare *DendroTweaks* with various modeling software to specify its place in the neuronal modeling landscape.

3.2 Comparison to existing modeling software

Single-cell modeling encompasses a diverse range of practices, from refining morphological data and optimizing biophysical properties to visualizing and analyzing neuronal dynamics. Over the years, numerous tools have been developed to aid in the creation, visualization, and optimization of neuronal models. Primary simulation environments like NEURON [Hines and Carnevale, 2001], NEST [Gewaltig and Diesmann, 2007], and Brian 2 [Stimberg et al., 2019] have been complemented by a variety of auxiliary software tools designed to enhance interaction with model parameters.

Several tools have been developed to assist in visualizing and editing morphological reconstructions of real neurons, such as neuTube [Feng et al., 2015], REMOD [Bozelos et al., 2016], HBP MORPHOLOGY VIEWER [Bakker et al., 2017], and HUGO [Aliaga Maraver et al., 2018]. *DendroTweaks* includes a simulator-independent tree graph construction module, enabling efficient morphology manipulation through operations such as inserting, removing, and translating nodes or subtrees. Its graphical interface offers the functionality for inspecting and refining morphological parameters to identify reconstruction artifacts and ”bugs” through visual exploration and statistical analysis. Nevertheless, the toolbox lacks advanced tools, such as 3D mesh editing and neuronal growth modeling capabilities. For more extensive morphology-focused needs, users are directed to specialized software like the TREES toolbox [Cuntz et al., 2011], Neuronize [Brito et al., 2013], NeuroEditor[Velasco et al., 2024], and NETMORTH [Koene et al., 2009].

With a growing number of models available online from repositories like *https://modeldb.science* [McDougal et al., 2017], *https://neuromorpho.org* [Ascoli et al., 2007], and *https://celltypes.brain-map.org* [Gouwens et al., 2018, 2019], standardization is crucial for ensuring reproducibility and reusability in neuronal modeling. Existing standards like SONATA [Dai et al., 2020], NeuroML [Gleeson et al., 2010], and NineML [The INCF Multiscale Modeling Taskforce and Gorchetchnikov, 2010] provide frameworks for model description. However, manual

standardization can be laborious and prone to errors. Importantly, *DendroTweaks* is not presented as a new standard but as a tool that conforms to a given standard to automatically standardize a model. Future releases might include an automatic export to an expanded range of formats, such as ChannelML [Gleeson et al., 2010]. Finally, to our knowledge, *DendroTweaks* is the only tool that can simultaneously parse existing ion channel model files, allow for visual exploration and fine-tuning of their kinetics and distributions via a GUI, and offer automatic standardization.

Another critical aspect of neuronal modeling is optimization of biophysical parameters. Evolutionary [Van Geit et al., 2016], Bayesian [Gonçalves et al., 2020], and gradient-based methods [Jones and Kording, 2024, Deistler et al., 2024] have been proposed to offer data-driven model parameter optimization. While the recent advent of such automated optimizers opens new possibilities for large-scale modeling, we argue that manual exploration remains a valuable and complementary approach. Human oversight provides both initial parameters before optimization and essential sanity checks after, making intuitive understanding a must when validating model behavior. *DendroTweaks* perfectly complements black-box optimization by providing an intuitive environment for hands-on parameter tuning and model refinement. Incorporating automatic optimization algorithms into *DendroTweaks* alongside interactive visualizations presents a promising future direction, combining the strengths of both approaches.

A range of validation frameworks has been developed to ensure that optimized parameter sets remain biophysically plausible. These approaches typically perform systematic searches across the parameter space and explicitly quantify heterogeneity, i.e., populations of models that reproduce experimental observations while capturing biological variability, and degeneracy, i.e., the ability of different parameter combinations to yield similar outputs [Migliore et al., 2018, Roy and Narayanan, 2021, 2023, Reva et al., 2023]. *DendroTweaks* approaches heterogeneity and degeneracy differently. Rather than generating model populations, it enables users to interactively navigate the parameter space, visualizing neuronal activity and applying built-in validation protocols to assess whether individual model configurations fall within biologically realistic regimes. While population-based frameworks excel at systematic large-scale exploration, *DendroTweaks* complements them by supporting hypothesis-driven interrogation of how specific parameters influence neuronal function.

Visualization and the development of intuitive graphical user interfaces have been crucial in neuronal modeling. One of the most successful examples is NetPyNE [Dura-Bernal et al., 2019], which offers a graphical interface for data-driven multiscale network modeling in NEURON. Meanwhile, *DendroTweaks* focuses on the subcellular level, providing a more explicit single-cell model interface. It facilitates an interactive approach to visualize and modify morphological parameters, ion channel kinetics and distributions, as well as to observe activity in different compartments. This capability is particularly important for models with active dendrites. We envision as a good practice exporting fine-tuned single-cell models from *DendroTweaks* and incorporating them into complex networks in NetPyNE.

Indeed, the ultimate goal of creating single-cell models is often to integrate them into a network. In this context, simplifying models becomes another crucial aspect of neuronal modeling. Morphology reduction is a common simplification technique with a long history. Pioneering works [Stratford et al., 1989, Bush and Sejnowski, 1993] aimed to conserve axial resistance, reducing multicompartmental models to 8-9 compartments. Later approaches focused on preserving voltage attenuation [Destexhe, 2001] or surface area [Hendrickson et al., 2011, Marasco et al., 2013]. The most recent methods such as neuron_reduce [Amsalem et al., 2020] and NEAT [Wybo et al., 2021] provide analytical solutions to the reduction problem by preserving the impedances of the original model. In *DendroTweaks*, we adopted the neuron_reduce approach and extended it to support a continuum of reduction levels, allowing users to gradually reduce dendritic subtrees. This added flexibility is important because dendritic trees rely on semi-independent subunit integration, which enhances localized processing and computational

complexity. Spatial organization of inputs plays a crucial role in their integration, with clustered inputs being more likely to drive somatic firing in pyramidal neurons [Poirazi et al., 2003, Polsky et al., 2004, Losonczy and Magee, 2006]. When multiple synaptic inputs from otherwise isolated branches are remapped to a reduced cylinder, clustering can occur, substantially altering the integration process. Therefore, the intermediate levels of reduction we added to neuron_reduce could potentially allow for more accurate implementation of independent input integration within dendritic subunits. A compelling example of when this might be needed is recent research [Otor et al., 2022], which showed that early bifurcating L5 pyramidal neurons exhibit pronounced functional compartmentalization of the apical dendrite, correlating with behavioral variables.

3.3 Limitations and further directions

DendroTweaks addresses a broad spectrum of single-cell modeling needs. However, it is important to acknowledge its limitations and outline promising future directions. The key feature of *DendroTweaks*, its real-time interactivity, is constrained by the performance of the underlying simulator. When running hyper-detailed simulations with numerous segments over extended periods of time, the simulation can become slow, reducing the interface's responsiveness and real-time update capabilities. To enhance performance, future versions of *DendroTweaks* should consider integrating faster simulation methods or developing a custom single-cell simulator. A promising approach is to run simulations using the optimized CoreNEURON [Kumbhar et al., 2019] and Dendritic Hierarchical Scheduling (DHS) algorithms [Zhang et al., 2023], which have been shown to greatly speed up NEURON simulations, or leverage Jaxley's built-in functionality for JIT compilation and GPU usage [Deistler et al., 2024].

While the current implementation of *DendroTweaks* is based on the NEURON simulator, the approach is essentially simulator-agnostic. As we have demonstrated for Jaxley, its functionality can be further extended to include other simulators. While our proof of concept shows that a multicompartmental model with multiple ion channels can be simulated using Jaxley as an alternative to NEURON, fully preserving the programming interface, this implementation is not optimized to leverage the full range of Jaxley capabilities and still lacks some of the features that will be added in future releases. Another promising future direction is to introduce an automated conversion of simplified biophysical models to integrate-and-fire few-compartmental models implemented in Brian 2 using Dendrify [Pagkalos et al., 2023]. This would allow for the extension of the proposed workflow by automatically simplifying not only the morphology but also the biophysics of a neuron, spanning any level of conceptual granularity.

While *DendroTweaks* models rely on specific parameters for ion channel kinetics and spatial distributions, experimental measurements of these properties reflect substantial variability across cells. Averaging kinetic data or conductance gradients across multiple cells can create the misleading impression of fixed parameter values, whereas in reality, each neuron may exhibit a distinct profile. Degeneracy further complicates modeling, as multiple, widely different combinations of kinetic and distribution parameters can produce nearly identical neuronal responses, making it difficult to infer unique underlying mechanisms. While DendroTweaks does not resolve these issues inherent to most neuronal models, it allows users to better understand them by interactively adjusting parameters and immediately observing their effects on model outputs.

Another limitation of the current *DendroTweaks* implementation is the range of ion channel models that can be standardized. As of now, only voltage-gated channels using the Hodgkin-Huxley (HH) formalism can be standardized. Even though Markov chain state-based kinetic models might offer a more accurate representation of ion channel kinetics [Lampert and Korngreen, 2014], they are not supported by *DendroTweaks*. Nevertheless, it is important to note that most models use the HH formalism, as Markov models can be more complex and slower to run. Additionally, our parsing and automated standardization algorithm relies on specific

heuristics and cannot handle significant deviations in MOD file code patterns. For example, the parser assumes that variable names for the time constant will include ”tau” (case insensitive). Therefore, for some files, the algorithm may require some minor preprocessing and manual tuning by the user.

In its current implementation, *DendroTweaks* offers relatively limited capabilities for representing synaptic inputs. Synaptic parameters are defined at the population level, and setting per-synapse properties is not yet supported. Synapse placement within a user-specified region follows a uniform random distribution. In addition, activation properties are specified abstractly in terms of rates, noise levels, and durations, without support for user-defined input sequences. Extending the toolbox to include per-synapse control, synaptic scaling [Magee and Cook, 2000], plasticity mechanisms, and more flexible connectivity definitions would substantially broaden the range of research questions that can be addressed using *DendroTweaks*.

The extended functionality of the morphology reduction algorithm in DendroTweaks provides a continuum of reduction levels that better preserve properties of the original model. However, even intermediate reduction levels inevitably alter integrative properties and eventually lead to the loss of dendritic computations. Nevertheless, it is worth noting that assessing how exactly morphology reduction affects integrative properties is one of the key use cases for *DendroTweaks*, thanks to its exploratory capabilities. Importantly, we don’t view reduction as an ultimate goal for any model but rather encourage approaches where the same system can be studied at different levels of granularity, as elegantly shown in [Billeh et al., 2020]. Beyond optimizing simulation speed, morphology reduction serves as a valuable technique to explore the importance of dendritic compartmentalization. The question of whether the computational unit of a neuron is a single spine, a branch, or an entire domain remains open [Häusser and Mel, 2003, Francioni and Harnett, 2022, Stuyt et al., 2022]. Using morphology reduction at multiple levels in *in silico* experiments can help address this question by revealing how dendritic morphology shapes compartmentalization, which in turn determines the neuron’s input-output transformation properties.

Finally, there is further potential for interoperability of DendroTweaks with other neuronal modeling software. *DendroTweaks* is designed to read and write the most popular formats for representing neuronal morphology (SWC) as well as ion channel models (MOD). It is possible to automatically generate plain NEURON (Python) code to export a refined model and use it outside the toolbox. Having the same option for other simulators, such as Jaxley, would also be beneficial. To enhance compatibility with existing models and the reusability of standardized models, support for more file formats and standards needs to be incorporated. The SONATA data format [Dai et al., 2020], designed to efficiently represent network models and simulation data, plays an important role in this regard. Integration with this format is currently under development, and support for exporting models in SONATA is planned for future releases. Another notable example is NeuroML [Gleeson et al., 2010], an XML-based neuronal model description language that provides a standardized notation for both morphological and biophysical parameters. By supporting such widely adopted standards, *DendroTweaks* can be more seamlessly integrated into a larger ecosystem of interoperable, open-source software [Sinha et al., 2024].

3.4 Conclusion

We believe that *DendroTweaks* will be appealing to a wide range of researchers. For those new to computational modeling, it provides an intuitive understanding of how model parameters influence neuronal dynamics, making it also a valuable educational tool. It lowers conceptual and technical barriers to biophysical neuronal modeling for experimentalists and specialists from adjacent fields. Given the growing interest in dendritic computations within artificial intelligence and neuromorphic computing, *DendroTweaks* offers clear visualizations of dendritic integration, benefiting those less familiar with biophysical modeling. For experienced modelers, *DendroTweaks* offers a comprehensive workflow, from single-channel analysis to validating

dendritic properties, with tools for visual inspection of cell topology, geometry, channel kinetics, and their distributions, as well as neuronal activity under variable stimuli, facilitating visual debugging. It also includes standardization and morphology reduction tools to improve model tractability and reusability for network simulations. *DendroTweaks* evolves with community feedback, adding new features over time. More than just a tool, *DendroTweaks* is a versatile framework that can integrate other visualizations and algorithms, ensuring its lasting relevance in the research community.

4 Methods

4.1 User interface

The user interface of *DendroTweaks* is implemented using Python 3, following the Model-View-Presenter (MVP) architectural pattern. The Model class defines a biophysical neuronal model and also serves as the core of a standalone Python package, which exposes the toolbox's main functionality for programmatic use. For the View class, we use the Python Bokeh library [Bokeh Development Team, 2014], which facilitates data visualization by generating the necessary JavaScript code to build the web-based interface. The Presenter is a class that acts as an intermediary, processing user commands, updating the Model accordingly, and ensuring that the View reflects the current state of the Model. This separation of concerns ensures a clean and maintainable codebase, allowing for efficient data handling and user interaction. The source code is openly available on GitHub for both the standalone Python package (*https://github.com/Poirazi-Lab/DendroTweaks*) and the web application (*https://github.com/Poirazi-Lab/DendroTweaksApp*). Detailed documentation and tutorials are available through ReadTheDocs (*https://dendrotweaks.readthedocs.io/en/latest*).

4.2 Biophysical models

To demonstrate the capabilities of the toolbox, we employed three well-established biophysical neuronal models with detailed morphology. The first model is an L2/3 pyramidal neuronal model with morphology reconstruction from Park et al., 2019, and biophysical mechanisms originally developed by Mainen et al. [Mainen and Sejnowski, 1996], further refined by Smith et al. [Smith et al., 2013], and recently utilized in Park et al. [Park et al., 2019] and Petousakis et al. [Petousakis et al., 2023b]. The membrane potential was initialized at V_{init} = −79 mV, and simulations were conducted at 37°C. The equilibrium potentials were set as follows: E_{leak} = −79 mV, E_{na} = 60 mV, E_{k} = −80 mV, and E_{ca} = 140 mV. The second model is a widely-used L5 pyramidal neuronal model with morphology reconstruction from Amitai et al. [Amitai et al., 1993] and biophysical mechanisms from Hay et al. [Hay et al., 2011]. For this model, the membrane potential was initialized at either V_{init} = −80 mV or V_{init} = −90 mV, with simulations performed at 37°C. The equilibrium potentials were E_{leak} = −90 mV, E_{na} = 50 mV, E_{k} = −85 mV, and E_{ca} = 132 mV. The third model is a CA1 hippocampal pyramidal neuron, based on Poirazi et al. [Poirazi et al., 2003] and [González, 2011]. Here, the membrane potential was initialized at V_{init} = −70 mV, with simulations conducted at 34°C. The equilibrium potentials were E_{leak} = −70 mV, E_{na} = 50 mV, E_{k} = −77 mV, and E_{ca} = 140 mV. All simulations were executed on a Dell G15 5515 laptop (Ryzen 7 5800H, 16GB RAM, Linux Ubuntu 20.04 LTS) with a spatial discretization factor d_λ = 0.1 and a time step dt = 0.025 ms.

4.3 Data format

DendroTweaks uses a custom modular data format tailored to single-cell modeling needs. This format builds on widely used file types. Neuronal morphology is represented in SWC files, which *DendroTweaks* can read, modify, and export using a custom graph-processing module. Ion channel models and other membrane mechanisms are described in MOD files (see the following section for details). The biophysical configuration of a cell is specified in JSON files, adapted from the Allen

Cell Types Database schema [Gouwens et al., 2018] and extended to support non-uniform parameter distributions. Each JSON file contains three main sections: 1) the mapping of domains to inserted membrane mechanisms, 2) the definition of segment groups, and 3) the mapping of segment groups to distribution functions for each model parameter. This structure enables concise and unified representation of complex distributions for any biophysical parameter, including passive and kinetic ones. Simulation, recording, and stimulation parameters are also stored in JSON files, while element-wise parameters, such as spatial locations of recordings and stimuli, are stored in CSV files. Each model is organized within a dedicated folder containing morphology, biophys, and stimuli subfolders. User-defined MOD files must be placed in the biophys/mod folder. Once this structure is established, the toolbox takes on most of the file management responsibilities from the user (e.g., specifying file paths, compiling MOD files, etc.).

4.4 Morphology processing and simulator-specific models

Building a neuronal morphology from an SWC file in *DendroTweaks* is implemented independently of any specific simulator. Each model is represented by interconnected tree graphs capturing the morphology at three abstraction levels: point tree, section tree, and segment tree. This structure allows users to build the model incrementally, starting from the point tree, for more precise control over morphology refinement. Each tree encapsulates level-specific properties and provides methods for common operations, such as sorting, inserting, and removing nodes or subtrees. Nodes maintain references to their children and parent, as well as attributes describing level-specific properties. Sections have access to their points and segments, and vice versa, linking all levels together. Additional methods are available for computing path distances, which are used when assigning parameter distributions. In parallel, *DendroTweaks* automatically generates corresponding sections and segments within the selected simulator, either NEURON [Hines, 2009] or Jaxley [Deistler et al., 2024], and maintains the references from its own sections and segments to the simulator objects. This enables seamless retrieval and modification of simulator-specific parameters while maintaining a simulator-independent scaffold.

4.5 NMODL to Python conversion

The Converter class encapsulates three components: a Reader, a Parser, and a CodeGenerator. The Reader performs basic preprocessing, such as removing comments and splitting the input into NMODL blocks, providing a structured basis for efficient parsing and handling of individual components. The Parser, implemented with the PyParsing library (*https://pyparsing-docs.readthedocs.io*), constructs an abstract syntax tree (AST) that represents the information contained in the MOD file. The CodeGenerator then uses the AST to produce a Python file from a JINJA template (*https://jinja.palletsprojects.com*). Users may also supply custom templates to generate specialized output files when required.

4.6 Standardization of ion channel models

A part of the neuronal membrane can be modeled as an equivalent RC circuit. The membrane acts as a capacitor that can store charge, while ion channels provide resistive pathways for current flow. Voltage dynamics in this RC circuit are governed by the fundamental principle of current conservation (Kirchhoff's current law), which states that the capacitive current (left) must balance the sum of all ionic, synaptic, and external currents (right).

$$C\frac{dV}{dt} = \sum_{i=1}^{n} I_{ion,i}(t) + \sum_{j=1}^{m} I_{syn,j}(t) + I_{ext}(t) \quad (1)$$

where:

V — the membrane potential in mV

C — the membrane capacitance in μF

$I_{ion,i}$ — individual ionic currents in nA

$I_{syn,j}$ — individual synaptic currents in nA

I_{ext} — external current in nA (e.g., through a patch electrode)

t — time in ms dV

The capacitive current term $C\frac{dV}{dt}$ is derived from the time derivative of the fundamental capacitance relationship $Q = CV$.

For a given ion channel, the current I is determined by the channel's conductance state and the driving force:

$$I = g \times p(x_1, \cdots, x_n) \times (V - E) \tag{2}$$

where:

g — the maximal channel conductance in S

x_i — a state variable representing channel gating (unitless)

p — the function defining the probability of the channel to be open (e.g., for an HH sodium channel $p(m, h) = m^3h$

E — the equilibrium potential in mV

$(V - E)$ — the driving force in mV

The time derivative of a state variable x is given by:

$$\frac{dx}{dt} = \frac{x^{\infty} - x}{\tau_x} \tag{3}$$

where:

x^{∞} — the steady-state value of x (unitless)

τ_x — the time constant in ms

The voltage-dependent steady-state value x^{∞} is defined as:

$$x^{\infty} = \frac{1}{1 + \exp\left(-\frac{V - V_{half}}{\sigma}\right)} \tag{4}$$

The voltage-dependent time constant τ_x is given by:

$$\tau_x = \frac{1}{\frac{d\alpha}{dt} + \frac{d\beta}{dt}} + \tau_0 \tag{5}$$

where:

$$\frac{d\alpha}{dt} = K \times \exp\left(\frac{\delta \times (V - V_{half})}{\sigma}\right) \quad (6)$$

$$\frac{d\beta}{dt} = K \times \exp\left(\frac{-(1-\delta) \times (V_{half} - V)}{\sigma}\right) \quad (7)$$

where:

V — the membrane potential in mV

V_{half} — the half-activation voltage in mV

σ — the inverse slope in mV

δ — the skew parameter of the time constant curve (unitless)

K — the maximum rate parameter in ms^{-1}

τ_0 — the rate-limiting factor (minimum time constant) in ms

For each state variable of a channel, we fit the set of 5 parameters of the system of equations (4 - 7), namely V_{half}, σ, K, δ, and τ_0, to the data in the form of activation (inactivation) curves derived from the original MOD files for membrane potentials in the range from -100 to 100 mV. The fitting process is implemented in the symfit Python library (*https://symfit.readthedocs.io*) to fit both curves for the steady state and the time constant simultaneously. The temperature is set to the original temperature before getting data to fit. We noticed that, while accurate for both curves, simultaneous fitting results in significant changes in the voltage and current dynamics. Therefore, we introduced a second additional fit for the steady state alone, sacrificing fitting accuracy for the time constant but preserving voltage and current dynamics. Finally, a new MOD file is created from a JINJA template and is immediately available to replace the original mechanism in the neuronal model.

4.7 Morphology reduction

We extended the analytical impedance-based neuron_reduce approach proposed by Amsalem et al., 2020 [Amsalem et al., 2020] and integrated it into our GUI. The original neuron_reduce algorithm maps a dendritic subtree to a single cylinder with both ends sealed, preserving:

specific membrane resistivity, R_m in $\Omega \times cm^2$

specific membrane capacitance, C_m in F/cm^2

specific axial resistivity, R_a in $\Omega \times cm$

transfer impedance from the electrotonically most distal dendritic tip to the soma, $|Z_{0,L}(\omega)|$

input resistance at the soma end (when disconnected from the soma), $|Z_{0,0}(\omega)|$

Equations (1)–(11) in the original paper describe the unique cylindrical cable (with a specific diameter, d and length, L, and the given membrane and axial properties) that preserves the values of $|Z_{0,L}(\omega)|$ and $|Z_{0,0}(\omega)|$. In the original implementation, the entire subtree of each stem dendrite (e.g., the entire apical subtree) is mapped to a single corresponding cylinder. We extended this approach to allow a user to select any section they want and map the inclusive

subtree of this section (including the section itself) to a single cylinder. When the user selects the desired section using the GUI and clicks the button 'Reduce subtree' the inclusive subtree is disconnected from the cell and parameters for its equivalent cylinder are calculated. The exclusive subtree of the section is then removed, and the section's length and diameter are updated with the new calculated values before reconnecting it to its original parent. As in the original method, the reduced model is compartmentalized into segments (typically with a spatial resolution of 0.1 λ), and channel conductances are adjusted according to the mapping between the original and the reduced segments. In order to introduce a more general workflow, where synapses can be allocated on the already reduced model, we removed the step from the original algorithm that mapped synapses to the corresponding cylinder. To enable export of reduced models in *DendroTweaks*'s modular format and plain simulator code, we implemented a fitting procedure that approximates spatial parameter distributions using analytical functions (e.g., polynomials or step-like). Candidate models were fit to parameter values as a function of distance from the soma, and the optimal model was chosen by minimizing mean squared error, with model complexity as a secondary criterion. This approach provides a compact representation, independent of the original segmentation, requiring only a few coefficients to be stored to reproduce a distribution.

4.8 Validation

We utilized a semi-automated approach for model validation. Users need to manually specify the simulation parameters for each specific validation protocol (see **Table 1**). Depending on the protocol, one or multiple simulation runs are performed, and the simulated voltage values are used for further calculations. Input resistance is calculated according to the formula: $R_{\text{in}} = \frac{V_{\text{offset}} - V_{\text{onset}}}{I_{\text{ext}}}$ .The membrane time constant (τ) is derived by fitting a double exponential equation to the decaying part of the voltage curve after the stimulus onset, taking the slowest component. Voltage attenuation is calculated for a user-specified set of segments by measuring the voltage response at different points along the dendrite. For each segment, the voltage attenuation is computed as the ratio of the voltage change at the segment (ΔV_{seg}) to the voltage change at the stimulation site (ΔV_{stim}). The distances from the soma to each segment are also recorded, and the attenuation is plotted against these distances. For detecting somatic action potentials and measuring their amplitude and half-width, we used the SciPy [Virtanen et al., 2020] Python library for peak detection in a signal. This involves identifying the peaks in the voltage trace and calculating the time difference between the points where the voltage is half of the peak amplitude. The somatic f-I curve is built by injecting a range of current amplitudes (e.g., 0 to 1 nA in steps of 0.1 nA) into the soma and recording the number of action potentials generated at each current level. The firing rate is plotted as a function of the injected current amplitude. Dendritic nonlinearities are derived by measuring the voltage response in a dendrite to increasing synaptic input weights of a single synapse. The unitary voltage response (EPSP) is determined, and the actual voltage responses for increasing synaptic weights are compared to the expected linear sum of unitary responses. The sag ratio is calculated according to the formula: Sag ratio $= \frac{a}{b}$, where $a = V_{\text{offset}} - V_{\text{min}}$ and $b = V_{\text{onset}} - V_{\text{min}}$. This ratio is derived from the voltage response to a hyperpolarizing current injection in the presence of HCN channels (h current), with V_{onset} being the initial voltage before the injection, V_{min} the minimum voltage reached, and V_{offset} the voltage at the end of the injection.

Data availability

No new datasets were generated in the ms.

Acknowledgements

We would like to thank members of the Poirazi lab and SmartNets ITN for their valuable feedback on the manuscript. This work was supported by NIH (1R01MH124867), the European Union, Horizon 2020 Programme (H2020-FETOPEN-2018-2019-2020-01, NEUREKA (GA-863245), H2020 MSCA ITN Project SmartNets (GA-860949) and iNavigate (GA-873178), and the Stavros Niarchos Foundation (SNF) and the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the 5th Call of "Science and Society" Action Always strive for excellence – Theodoros Papazoglou" grant DendroLeap (Project Number: 28056).

Additional information

Funding

National Institutes of Health (1R01MH124867)

The European Union Horizon 2020 Programme (GA-863245)

The European Union Horizon 2020 Programme (GA-860949)

The European Union Horizon 2020 Programme (GA-873178)

The Stavros Niarchos Foundation (SNF) and the Hellenic Foundation for Research and Innovation (H.F.R.I.) (28056)

References

Aliaga Maraver J. J., Mata S., Benavides-Piccione R., DeFelipe J., Pastor L. (2018) **A Method for the Symbolic Representation of Neurons** *Frontiers in Neuroanatomy* **12**:106 https://doi.org/10.3389/fnana.2018.00106 | Google Scholar

Amitai Y., Friedman A., Connors B. W., Gutnick M. J. (1993) **Regenerative Activity in Apical Dendrites of Pyramidal Cells in Neocortex** *Cerebral Cortex* **3**:26–38 https://doi.org/10.1093/cercor/3.1.26 | Google Scholar

Amsalem O., Eyal G., Rogozinski N., Gevaert M., Kumbhar P., Schürmann F., Segev I. (2020) **An efficient analytical reduction of detailed nonlinear neuron models** *Nature Communications* **11**:288 https://doi.org/10.1038/s41467-019-13932-6 | Google Scholar

Ascoli G. A., Donohue D. E., Halavi M. (2007) **NeuroMorpho.Org: A Central Resource for Neuronal Morphologies** *The Journal of Neuroscience* **27**:9247–9251 https://doi.org/10.1523/JNEUROSCI.2055-07.2007 | Google Scholar

Bakker R., García-Amado M., Evangelio M., Clascá F., Tiesinga P., et al. (2017) **P271 workflow, data format and tools to register neuron morphologies to a reference brain atlas** In: *26th Annual Computational Neuroscience Meeting (CNS*2017): Part 3* :60 https://doi.org/10.1186/s12868-017-0372-1 | Google Scholar

Billeh Y. N., Cai B., Gratiy S. L., Dai K., Iyer R., Gouwens N. W., Abbasi-Asl R., Jia X., Siegle J. H., Olsen S. R., Koch C., Mihalas S., Arkhipov A. (2020) **Systematic Integration of Structural and Functional Data into Multi-scale Models of Mouse Primary Visual Cortex** *Neuron* **106**:388–403 https://doi.org/10.1016/j.neuron.2020.01.040 | Google Scholar

Bokeh Development Team (2014) **Bokeh: Python library for interactive visualization** http://www.bokeh.pydata.org

Bozelos P., Stefanou S. S., Bouloukakis G., Melachrinos C., Poirazi P. (2016) **REMOD: A Tool for Analyzing and Remodeling the Dendritic Architecture of Neural Cells** *Frontiers in Neuroanatomy* **9** https://doi.org/10.3389/fnana.2015.00156 | Google Scholar

Brito J. P., Mata S., Bayona S., Pastor L., DeFelipe J., Benavides-Piccione R. (2013) **Neuronize: a tool for building realistic neuronal cell morphologies** *Frontiers in Neuroanatomy* **7** https://doi.org/10.3389/fnana.2013.00015 | Google Scholar

Bush P. C., Sejnowski T. J. (1993) **Reduced compartmental models of neocortical pyramidal cells** *Journal of Neuroscience Methods* **46**:159–166 https://doi.org/10.1016/0165-0270(93)90151-G | Google Scholar

Carnevale N. T., Hines M. L. (2006) **The NEURON Book** Cambridge University Press https://doi.org/10.1017/CBO9780511541612 | Google Scholar

Cuntz H., Forstner F., Borst A., Häusser M. (2011) **The TREES Toolbox—Probing the Basis of Axonal and Dendritic Branching** *Neuroinformatics* **9**:91–96 https://doi.org/10.1007/s12021-010-9093-7 | Google Scholar

Dai K., Hernando J., Billeh Y. N., Gratiy S. L., Planas J., Davison A. P., Dura-Bernal S., Gleeson P., Devresse A., Dichter B. K., Gevaert M., King J. G., Van Geit W. A. H., Povolotsky A. V., Muller E., Courcol J.-D., Arkhipov A. (2020) **The SONATA data format for efficient description of large-scale network models** *PLOS Computational Biology* **16**:e1007696 https://doi.org/10.1371/journal.pcbi.1007696 | Google Scholar

Deistler M., Kadhim K. L., Pals M., Beck J., Huang Z., Gloeckler M., Lappalainen J. K., Schröder C., Berens P., Gonçalves P. J., Macke J. H. (2024) **Jaxley: Differentiable simulation enables large-scale training of detailed biophysical models of neural dynamics** http://biorxiv.org/lookup/doi/10.1101/2024.08.21.608979

Destexhe A. (2001) **Simplified models of neocortical pyramidal cells preserving somatodendritic voltage attenuation** *Neurocomputing* **38-40**:167–173 https://doi.org/10.1016/S0925-2312(01)00428-3 | Google Scholar

Doron M., Chindemi G., Muller E., Markram H., Segev I. (2017) **Timed Synaptic Inhibition Shapes NMDA Spikes, Influencing Local Dendritic Processing and Global I/O Properties of Cortical Neurons** *Cell Reports* **21**:1550–1561 https://doi.org/10.1016/j.celrep.2017.10.035 | Google Scholar

Du K., Wu Y.-W., Lindroos R., Liu Y., Rózsa B., Katona G., Ding J. B., Kotaleski J. H. (2017) **Cell-type–specific inhibition of the dendritic plateau potential in striatal spiny projection neurons** *Proceedings of the National Academy of Sciences* **114** https://doi.org/10.1073/pnas.1704893114 | Google Scholar

Dura-Bernal S., Suter B. A., Gleeson P., Cantarelli M., Quintana A., Rodriguez F., Kedziora D. J., Chadderdon G. L., Kerr C. C., Neymotin S. A., McDougal R. A., Hines M., Shepherd G. M., Lytton

W. W. (2019) **NetPyNE, a tool for data-driven multiscale modeling of brain circuits** *eLife* **8**:e44494 https://doi.org/10.7554/eLife.44494 | Google Scholar

Feng L., Zhao T., Kim J. (2015) **neuTube 1.0: A New Design for Efficient Neuron Reconstruction Software Based on the SWC Format** *eneuro* **2**:ENEURO.0049–14.2014 https://doi.org/10.1523/ENEURO.0049-14.2014 | Google Scholar

Francioni V., Harnett M. T. (2022) **Rethinking Single Neuron Electrical Compartmentalization: Den-dritic Contributions to Network Computation In Vivo** *Neuroscience* **489**:185–199 https://doi.org/10.1016/j.neuroscience.2021.05.038 | Google Scholar

Gewaltig M.-O., Diesmann M. (2007) **Nest (neural simulation tool)** *Scholarpedia* **2**:1430 Google Scholar

Gleeson P., Crook S., Cannon R. C., Hines M. L., Billings G. O., Farinella M., Morse T. M., Davison A. P., Ray S., Bhalla U. S., Barnes S. R., Dimitrova Y. D., Silver R. A. (2010) **NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail** *PLoS Computational Biology* **6**:e1000815 https://doi.org/10.1371/journal.pcbi.1000815 | Google Scholar

Golding N. L., Spruston N. (1998) **Dendritic Sodium Spikes Are Variable Triggers of Axonal Action Potentials in Hippocampal CA1 Pyramidal Neurons** *Neuron* **21**:1189–1200 https://doi.org/10.1016/S0896-6273(00)80635-2 | Google Scholar

Golding N. L., Mickus T. J., Katz Y., Kath W. L., Spruston N. (2005) **Factors mediating powerful voltage attenuation along CA1 pyramidal neuron dendrites** *The Journal of Physiology* **568**:69–82 https://doi.org/10.1113/jphysiol.2005.086793 | Google Scholar

González J. (2011) **Distinguishing linear vs. non-linear integration in CA1 radial oblique dendrites: it's about time** *Frontiers in Computational Neuroscience* **5** https://doi.org/10.3389/fncom.2011.00044 | Google Scholar

Gonçalves P. J., Lueckmann J.-M., Deistler M., Nonnenmacher M., Öcal K., Bassetto G., Chintaluri C., Podlaski W. F., Haddad S. A., Vogels T. P., Greenberg D. S., Macke J. H. (2020) **Training deep neural density estimators to identify mechanistic models of neural dynamics** *eLife* **9**:e56261 https://doi.org/10.7554/eLife.56261 | Google Scholar

Gouwens N. W., Berg J., Feng D., Sorensen S. A., Zeng H., Hawrylycz M. J., Koch C., Arkhipov A. (2018) **Systematic generation of biophysically detailed models for diverse cortical neuron types** *Nature Communications* **9**:710 https://doi.org/10.1038/s41467-017-02718-3 | Google Scholar

Gouwens N. W., Sorensen S. A., Berg J., Lee C., Jarsky T., Ting J., Sunkin S. M., Feng D., Anastassiou C. A., Barkan E., Bickley K., Blesie N., Braun T., Brouner K., Budzillo A., Caldejon S., Casper T., Castelli D., Chong P., Crichton K., Cuhaciyan C., Daigle T. L., Dalley R., Dee N., Desta T., Ding S.-L., Dingman S., Doperalski A., Dotson N., Egdorf T., Fisher M., De Frates R. A., Garren E., Garwood M., Gary A., Gaudreault N., Godfrey K., Gorham M., Gu H., Habel C., Hadley K., Harrington J., Harris J. A., Henry A., Hill D., Josephsen S., Kebede S., Kim L., Kroll M., Lee B., Lemon T., Link K. E., Liu X., Long B., Mann R., Mc-Graw M., Mihalas S., Mukora A., Murphy G. J., Ng L., Ngo K., Nguyen T. N., Nicovich P. R., Oldre A., Park D., Parry S., Perkins J., Potekhina L., Reid D., Robertson M., Sandman D., Schroedter M., Slaughterbeck C., Soler-Llavina G., Sulc J., Szafer A., Tasic B., Taskin N., Teeter C., Thatra N., Tung H., Wakeman W., Williams G., Young R., Zhou Z., Farrell C., Peng H., Hawrylycz M. J., Lein E., Ng L., Arkhipov A., Bernard A., Phillips J. W., Zeng H., Koch C. (2019) **Classification of electrophysiological and morphological neuron**

types in the mouse visual cortex *Nature Neuroscience* **22**:1182–1195 https://doi.org/10.1038/s41593-019-0417-0 | Google Scholar

Hay E., Hill S., Schürmann F., Markram H., Segev I. (2011) **Models of Neocortical Layer 5b Pyramidal Cells Capturing a Wide Range of Dendritic and Perisomatic Active Properties** *PLoS Computational Biology* **7**:e1002107 https://doi.org/10.1371/journal.pcbi.1002107 | Google Scholar

Hendrickson E. B., Edgerton J. R., Jaeger D. (2011) **The capabilities and limitations of conductance-based compartmental neuron models with reduced branched or unbranched morphologies and active dendrites** *Journal of Computational Neuroscience* **30**:301–321 https://doi.org/10.1007/s10827-010-0258-z | Google Scholar

Hines M. (2009) **NEURON and Python** *Frontiers in Neuroinformatics* **3** https://doi.org/10.3389/neuro.11.001.2009 | Google Scholar

Hines M. L., Carnevale N. T. (2000) **Expanding NEURON's Repertoire of Mechanisms with NMODL** *Neural Computation* **12**:995–1007 https://doi.org/10.1162/089976600300015475 | Google Scholar

Hines M. L., Carnevale N. T. (2001) **Neuron: A Tool for Neuroscientists** *The Neuroscientist* **7**:123–135 https://doi.org/10.1177/107385840100700207 | Google Scholar

Hodgkin A. L., Huxley A. F. (1952) **A quantitative description of membrane current and its application to conduction and excitation in nerve** *The Journal of Physiology* **117**:500–544 https://doi.org/10.1113/jphysiol.1952.sp004764 | Google Scholar

Hoffman D. A., Magee J. C., Colbert C. M., Johnston D. (1997) **K+ channel regulation of signal prop-agation in dendrites of hippocampal pyramidal neurons** *Nature* **387**:869–875 https://doi.org/10.1038/43119 | Google Scholar

Häusser M., Mel B. (2003) **Dendrites: bug or feature?** *Current Opinion in Neurobiology* **13**:372–383 https://doi.org/10.1016/S0959-4388(03)00075-8 | Google Scholar

Johnston D., Narayanan R. (2008) **Active dendrites: colorful wings of the mysterious butterflies** *Trends in Neurosciences* **31**:309–316 https://doi.org/10.1016/j.tins.2008.03.004 | Google Scholar

Jones I. S., Kording K. P. (2024) **Efficient optimization of ODE neuron models using gradient descent** https://arxiv.org/abs/2407.04025

Koene R. A., Tijms B., Van Hees P., Postma F., De Ridder A., Ramakers G. J. A., Van Pelt J., Van Ooyen A. (2009) **NETMORPH: A Framework for the Stochastic Generation of Large Scale Neuronal Networks With Realistic Neuron Morphologies** *Neuroinformatics* **7**:195–210 https://doi.org/10.1007/s12021-009-9052-3 | Google Scholar

Kole M. H. P., Hallermann S., Stuart G. J. (2006) **Single I h Channels in Pyramidal Neuron Dendrites: Properties, Distribution, and Impact on Action Potential Output** *The Journal of Neuroscience* **26**:1677–1687 https://doi.org/10.1523/JNEUROSCI.3664-05.2006 | Google Scholar

Kumbhar P., Hines M., Fouriaux J., Ovcharenko A., King J., Delalondre F., Schürmann F. (2019) **CoreNEURON: An Optimized Compute Engine for the NEURON Simulator** *Frontiers in Neuroinformatics* **13**:63 https://doi.org/10.3389/fninf.2019.00063 | Google Scholar

Kumbhar P., Awile O., Keegan L., Alonso J. B., King J., Hines M., Schürmann F. (2020) **An Optimizing Multi-platform Source-to-source Compiler Framework for the NEURON MODeling Language** In: *Computational Science – ICCS 2020* pp. 45–58 https://doi.org/10.1007/978-3-030-50371-0_4 | Google Scholar

Lampert A., Korngreen A. (2014) **Markov Modeling of Ion Channels** In: Blackwell K.T., editors. *Progress in Molecular Biology and Translational Science* Elsevier pp. 1–21 https://doi.org/10.1016/B978-0-12-397897-4.00009-7 | Google Scholar

Llinas R., Nicholson C. (1971) **Electrophysiological properties of dendrites and somata in alligator Purkinje cells** *Journal of Neurophysiology* **34**:532–551 https://doi.org/10.1152/jn.1971.34.4.532 | Google Scholar

Llinás R., Hess R. (1976) **Tetrodotoxin-resistant dendritic spikes in avian Purkinje cells** *Proceedings of the National Academy of Sciences* **73**:2520–2523 https://doi.org/10.1073/pnas.73.7.2520 | Google Scholar

Losonczy A., Magee J. C. (2006) **Integrative Properties of Radial Oblique Dendrites in Hippocampal CA1 Pyramidal Neurons** *Neuron* **50**:291–307 https://doi.org/10.1016/j.neuron.2006.03.016 | Google Scholar

Lörincz A., Notomi T., Tamás G., Shigemoto R., Nusser Z. (2002) **Polarized and compartment-dependent distribution of HCN1 in pyramidal cell dendrites** *Nature Neuroscience* **5**:1185–1193 https://doi.org/10.1038/nn962 | Google Scholar

Magee J. C. (1998) **Dendritic Hyperpolarization-Activated Currents Modify the Integrative Properties of Hippocampal CA1 Pyramidal Neurons** *The Journal of Neuroscience* **18**:7613–7624 https://doi.org/10.1523/JNEUROSCI.18-19-07613.1998 | Google Scholar

Magee J. C., Cook E. P. (2000) **Somatic EPSP amplitude is independent of synapse location in hippocampal pyramidal neurons** *Nature Neuroscience* **3**:895–903 https://doi.org/10.1038/78800 | Google Scholar

Magee J. C., Johnston D. (1995) **Characterization of single voltage-gated Na+ and Ca2+ channels in apical dendrites of rat CA1 pyramidal neurons** *The Journal of Physiology* **487**:67–90 https://doi.org/10.1113/jphysiol.1995.sp020862 | Google Scholar

Magee J. C., Christofi G., Miyakawa H., Christie B., Lasser-Ross N., Johnston D. (1995) **Subthreshold synaptic activation of voltage-gated Ca2+ channels mediates a localized Ca2+ in-flux into the dendrites of hippocampal pyramidal neurons** *Journal of Neurophysiology* **74**:1335–1342 https://doi.org/10.1152/jn.1995.74.3.1335 | Google Scholar

Mainen Z. F., Sejnowski T. J. (1996) **Influence of dendritic structure on firing pattern in model neocortical neurons** *Nature* **382**:363–366 https://doi.org/10.1038/382363a0 | Google Scholar

Marasco A., Limongiello A., Migliore M. (2013) **Using Strahler's analysis to reduce up to 200-fold the run time of realistic neuron models** *Scientific Reports* **3**:2934 https://doi.org/10.1038/srep02934 | Google Scholar

Markram H., Toledo-Rodriguez M., Wang Y., Gupta A., Silberberg G., Wu C. (2004) **Interneurons of the neocortical inhibitory system** *Nature Reviews Neuroscience* **5**:793–807 https://doi.org/10.1038/nrn1519 | Google Scholar

Markram H., Muller E., Ramaswamy S., Reimann M., Abdellah M., Sanchez C., Ailamaki A., Alonso-Nanclares L., Antille N., Arsever S., Kahou G., Berger T., 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 M., Ghobril J.-P., Gidon A., Graham J., Gupta A., Haenel V., Hay E., Heinis T., Hernando J., Hines M., Kanari L., Keller D., Kenyon J., Khazen G., Kim Y., King J., Kisvarday Z., Kumbhar P., Lasserre S., Le Bé J.-V., Magalhães B., Merchán-Pérez A., Meystre J., Morrice B., Muller J., Muñoz-Céspedes A., Muralidhar S., Muthurasa K., Nachbaur D., Newton T., Nolte M., Ovcharenko A., Palacios J., Pastor L., Perin R., Ranjan R., Riachi I., Rodríguez J.-R., Riquelme J., Rössert C., Sfyrakis K., Shi Y., Shillcock J., Silberberg G., Silva R., Tauheed F., Telefont M., Toledo-Rodriguez M., Tränkler T., Van Geit W., Díaz J., Walker R., Wang Y., Zaninetta S., DeFelipe J., Hill S., Segev I., Schürmann F. (2015) **Reconstruction and Simulation of Neocortical Microcircuitry** *Cell* **163**:456–492 https://doi.org/10.1016/j.cell.2015.09.029 | Google Scholar

McDougal R. A., Morse T. M., Carnevale T., Marenco L., Wang R., Migliore M., Miller P. L., Shepherd G. M., Hines M. L. (2017) **Twenty years of ModelDB and beyond: building essential modeling tools for the future of neuroscience** *Journal of Computational Neuroscience* **42**:1–10 https://doi.org/10.1007/s10827-016-0623-7 | Google Scholar

Mehta K., Ljungquist B., Ogden J., Nanda S., Ascoli R. G., Ng L., Ascoli G. A. (2023) **Online conversion of reconstructed neural morphologies into standardized SWC format** *Nature Communications* **14**:7429 https://doi.org/10.1038/s41467-023-42931-x | Google Scholar

Migliore M., Shepherd G. M. (2002) **Emerging rules for the distributions of active dendritic conductances** *Nature Reviews Neuroscience* **3**:362–370 https://doi.org/10.1038/nrn810 | Google Scholar

Migliore M., Hoffman D., Magee J., Johnston D. (1999) **Role of an A-Type K+ Conductance in the Back-Propagation of Action Potentials in the Dendrites of Hippocampal Pyramidal Neurons** *Journal of Computational Neuroscience* **7**:5–15 https://doi.org/10.1023/A:1008906225285 | Google Scholar

Migliore R., Lupascu C. A., Bologna L. L., Romani A., Courcol J.-D., Antonel S., Van Geit W. A. H., Thomson A. M., Mercer A., Lange S., Falck J., Rössert C. A., Shi Y., Hagens O., Pezzoli M., Freund T. F., Kali S., Muller E. B., Schürmann F., Markram H., Migliore M. (2018) **The physiological variability of channel density in hippocampal CA1 pyramidal cells and interneurons explored using a unified data-driven modeling workflow** *PLOS Computational Biology* **14**:e1006423 https://doi.org/10.1371/journal.pcbi.1006423 | Google Scholar

Nusser Z. (2009) **Variability in the subcellular distribution of ion channels increases neuronal diversity** *Trends in Neurosciences* **32**:267–274 https://doi.org/10.1016/j.tins.2009.01.003 | Google Scholar

Otor Y., Achvat S., Cermak N., Benisty H., Abboud M., Barak O., Schiller Y., Poleg-Polsky A., Schiller J. (2022) **Dynamic compartmental computations in tuft dendrites of layer 5 neurons during motor behavior** *Science* **376**:267–275 https://doi.org/10.1126/science.abn1421 | Google Scholar

Pagkalos M., Chavlis S., Poirazi P. (2023) **Introducing the Dendrify framework for incorporating dendrites to spiking neural networks** *Nature Communications* **14**:131 https://doi.org/10.1038/s41467-022-35747-8 | Google Scholar

Park J., Papoutsi A., Ash R. T., Marin M. A., Poirazi P., Smirnakis S. M. (2019) **Contribution of apical and basal dendrites to orientation encoding in mouse V1 L2/3 pyramidal neurons**

Nature Communications **10**:5372 https://doi.org/10.1038/s41467-019-13029-0 | Google Scholar

Petousakis K., Apostolopoulou A. A., Poirazi P. (2023a) **The impact of Hodgkin–Huxley models on dendritic research** *The Journal of Physiology* **601**:3091–3102 https://doi.org/10.1113/JP282756 | Google Scholar

Petousakis K.-E., Park J., Papoutsi A., Smirnakis S., Poirazi P. (2023b) **Modeling apical and basal tree contribution to orientation selectivity in a mouse primary visual cortex layer 2/3 pyramidal cell** *eLife* **12**:e91627 https://doi.org/10.7554/eLife.91627 | Google Scholar

Podlaski W. F., Seeholzer A., Groschner L. N., Miesenböck G., Ranjan R., Vogels T. P. (2017) **Mapping the function of neuronal ion channels in model and experiment** *eLife* **6**:e22152 https://doi.org/10.7554/eLife.22152 | Google Scholar

Poirazi P., Brannon T., Mel B. W. (2003) **Pyramidal Neuron as Two-Layer Neural Network** *Neuron* **37**:989–999 https://doi.org/10.1016/S0896-6273(03)00149-1 | Google Scholar

Polsky A., Mel B. W., Schiller J. (2004) **Computational subunits in thin dendrites of pyramidal cells** *Nature Neuroscience* **7**:621–627 https://doi.org/10.1038/nn1253 | Google Scholar

Poolos N. P., Migliore M., Johnston D. (2002) **Pharmacological upregulation of h-channels reduces the excitability of pyramidal neuron dendrites** *Nature Neuroscience* **5**:767–774 https://doi.org/10.1038/nn891 | Google Scholar

Rall W. (1959) **Branching dendritic trees and motoneuron membrane resistivity** *Experimental Neurology* **1**:491–527 https://doi.org/10.1016/0014-4886(59)90046-9 | Google Scholar

Ranjan R., Khazen G., Gambazzi L., Ramaswamy S., Hill S. L., Schürmann F., Markram H. (2011) **Channelpedia: An Integrative and Interactive Database for Ion Channels** *Frontiers in Neuroinformatics* **5** https://doi.org/10.3389/fninf.2011.00036 | Google Scholar

Ranjan R., Logette E., Dorp S. V., Kalaimaken H. A., Herzog M., Joffraud M. S., Scantamburlo E., Johnston K. G., Journe A., Markram H. (2024) **Channelome: A comprehensive resource for voltage-gated ion channel kinetics** *Biophysical Journal* **123**:527a https://doi.org/10.1016/j.bpj.2023.11.3186 | Google Scholar

Reuveni I., Friedman A., Amitai Y., Gutnick M. (1993) **Stepwise repolarization from Ca2+ plateaus in neocortical pyramidal cells: evidence for nonhomogeneous distribution of HVA Ca2+ channels in dendrites** *The Journal of Neuroscience* **13**:4609–4621 https://doi.org/10.1523/JNEUROSCI.13-11-04609.1993 | Google Scholar

Reva M., Rössert C., Arnaudon A., Damart T., Mandge D., Tuncel A., Ramaswamy S., Markram H., Van Geit W. (2023) **A universal workflow for creation, validation, and generalization of detailed neuronal models** *Patterns* **4**:100855 https://doi.org/10.1016/j.patter.2023.100855 | Google Scholar

Roy A., Narayanan R. (2021) **Spatial information transfer in hippocampal place cells depends on trial-to-trial variability, symmetry of place-field firing, and biophysical heterogeneities** *Neural Networks* **142**:636–660 https://doi.org/10.1016/j.neunet.2021.07.026 | Google Scholar

Roy R., Narayanan R. (2023) **Ion-channel degeneracy and heterogeneities in the emergence of complex spike bursts in CA3 pyramidal neurons** *The Journal of Physiology* **601**:3297–

3328 https://doi.org/10.1113/JP283539 | Google Scholar

Schiller J., Schiller Y. (2001) **NMDA receptor-mediated dendritic spikes and coincident signal amplification** *Current Opinion in Neurobiology* **11**:343–348 https://doi.org/10.1016/S0959-4388(00)00217-8 | Google Scholar

Shah M. M., Hammond R. S., Hoffman D. A. (2010) **Dendritic ion channel trafficking and plasticity** *Trends in Neurosciences* **33**:307–316 https://doi.org/10.1016/j.tins.2010.03.002 | Google Scholar

Sinha A., Gleeson P., Marin B., Dura-Bernal S., Panagiotou S., Crook S., Cantarelli M., Cannon R. C., Davison A. P., Gurnani H., Silver R. A. (2024) **The NeuroML ecosystem for standardized multi-scale modeling in neuroscience** *eLife* **13**:RP95135 https://doi.org/10.7554/eLife.95135.1 | Google Scholar

Smith S. L., Smith I. T., Branco T., Häusser M. (2013) **Dendritic spikes enhance stimulus selectivity in cortical neurons in vivo** *Nature* **503**:115–120 https://doi.org/10.1038/nature12600 | Google Scholar

Spencer W. A., Kandel E. R. (1961) **Electrophysiology of hippocampal neurons: IV. Fast prepotentials** *Journal of Neurophysiology* **24**:272–285 https://doi.org/10.1152/jn.1961.24.3.272 | Google Scholar

Stimberg M., Brette R., Goodman D. F. (2019) **Brian 2, an intuitive and efficient neural simulator** *eLife* **8**:e47314 https://doi.org/10.7554/eLife.47314 | Google Scholar

Stratford K., Mason A., Larkman A., Major G., Jack J. (1989) **The modeling of pyramidal neurons in the visual cortex** In: Durbin R., Miall C., Mitchison G., editors. *The Computing Neuron* London: Addison-Wesley pp. 296–321 Google Scholar

Stuart G., Spruston N. (1998) **Determinants of Voltage Attenuation in Neocortical Pyramidal Neuron Dendrites** *The Journal of Neuroscience* **18**:3501–3510 https://doi.org/10.1523/JNEUROSCI.18-10-03501.1998 | Google Scholar

Stuyt G., Godenzini L., Palmer L. M. (2022) **Local and Global Dynamics of Dendritic Activity in the Pyramidal Neuron** *Neuroscience* **489**:176–184 https://doi.org/10.1016/j.neuroscience.2021.07.008 | Google Scholar

Takahashi N., Kitamura K., Matsuo N., Mayford M., Kano M., Matsuki N., Ikegaya Y. (2012) **Locally Synchronized Synaptic Inputs** *Science* **335**:353–356 https://doi.org/10.1126/science.1210362 | Google Scholar

The INCF Multiscale Modeling Taskforce, Gorchetchnikov A. (2010) **NineML – a description language for spiking neuron network modeling: the user layer** *BMC Neuroscience* **11**:P71 https://doi.org/10.1186/1471-2202-11-S1-P71 | Google Scholar

Tran-Van-Minh A., Cazé R. D., Abrahamsson T., Cathala L., Gutkin B. S., DiGregorio D. A. (2015) **Contribution of sublinear and supralinear dendritic integration to neuronal computations** *Frontiers in Cellular Neuroscience* **9** https://doi.org/10.3389/fncel.2015.00067 | Google Scholar

Van Geit W., Gevaert M., Chindemi G., Rössert C., Courcol J.-D., Muller E. B., Schürmann F., Segev I., Markram H. (2016) **BluePyOpt: Leveraging Open Source Software and Cloud**

Infrastructure to Optimise Model Parameters in Neuroscience *Frontiers in Neuroinformatics* **10** https://doi.org/10.3389/fninf.2016.00017 | Google Scholar

Van Versendaal D., Levelt C. N. (2016) **Inhibitory interneurons in visual cortical plasticity** *Cellular and Molecular Life Sciences* **73**:3677–3691 https://doi.org/10.1007/s00018-016-2264-4 | Google Scholar

Velasco I., Garcia-Cantero J. J., Brito J. P., Bayona S., Pastor L., Mata S. (2024) **NeuroEditor: a tool to edit and visualize neuronal morphologies** *Frontiers in Neuroanatomy* **18**:1342762 https://doi.org/10.3389/fnana.2024.1342762 | Google Scholar

Virtanen P., Gommers R., Oliphant T. E., et al. (2020) **SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python** *Nature Methods* **17**:261–272 https://doi.org/10.1038/s41592-019-0686-2 | Google Scholar

Westenbroek R. E., Ahlijanian M. K., Catterall W. A. (1990) **Clustering of L-type Ca2+ channels at the base of major dendrites in hippocampal pyramidal neurons** *Nature* **347**:281–284 https://doi.org/10.1038/347281a0 | Google Scholar

Wybo W. A., Jordan J., Ellenberger B., Marti Mengual U., Nevian T., Senn W. (2021) **Data-driven reduction of dendritic morphologies with preserved dendro-somatic responses** *eLife* **10**:e60936 https://doi.org/10.7554/eLife.60936 | Google Scholar

Yu F. H., Yarov-Yarovoy V., Gutman G. A., Catterall W. A. (2005) **Overview of Molecular Relationships in the Voltage-Gated Ion Channel Superfamily** *Pharmacological Reviews* **57**:387–395 https://doi.org/10.1124/pr.57.4.13 | Google Scholar

Yuste R., Gutnick M. J., Saar D., Delaney K. R., Tank D. W. (1994) **Ca2+ accumulations in dendrites of neocortical pyramidal neurons: An apical band and evidence for two functional compartments** *Neuron* **13**:23–43 https://doi.org/10.1016/0896-6273(94)90457-X | Google Scholar

Zador A., Koch C. (1994) **Linearized models of calcium dynamics: formal equivalence to the cable equation** *Journal of Neuroscience* **14**:4705–4715 https://doi.org/10.1523/JNEUROSCI.14-08-04705.1994 | Google Scholar

Zhang Y., He G., Ma L., Liu X., Hjorth J. J. J., Kozlov A., He Y., Zhang S., Kotaleski J. H., Tian Y., Grillner S., Du K., Huang T. (2023) **A GPU-based computational framework that bridges neuron simulation and artificial intelligence** *Nature Communications* **14**:5798 https://doi.org/10.1038/s41467-023-41553-7 | Google Scholar

Author information

Roman Makarov

Institute of Molecular Biology and Biotechnology (IMBB), Foundation for Research and Technology-Hellas (FORTH), Heraklion, Greece, Department of Biology, University of Crete, Heraklion, Greece
ORCID iD: 0000-0002-4174-3826

Spyridon Chavlis

Institute of Molecular Biology and Biotechnology (IMBB), Foundation for Research and Technology-Hellas (FORTH), Heraklion, Greece

ORCID iD: 0000-0002-1046-1201

Panayiota Poirazi
Institute of Molecular Biology and Biotechnology (IMBB), Foundation for Research and Technology-Hellas (FORTH), Heraklion, Greece
ORCID iD: 0000-0001-6152-595X

For correspondence: poirazi@imbb.forth.gr

Editors

Reviewing Editor
Casey Schneider-Mizell
Allen Institute for Brain Science, Seattle, United States of America

Senior Editor
Albert Cardona
University of Cambridge, Cambridge, United Kingdom

Reviewer #1 (Public review):

Summary:

Dendrotweaks provides to its users a solid tool to implement, visualize, tune, validate, understand, and reduce single-neuron models that incorporate complex dendritic arbors with differential distribution of biophysical mechanisms. The visualization of dendritic segments and biophysical mechanisms therein provide users an intuitive way to understand and appreciate dendritic physiology.

https://doi.org/10.7554/eLife.103324.2.sa2

Reviewer #2 (Public review):

The paper by Makarov et al. describes the software tool called DendroTweaks, intended for examination of multi-compartmental biophysically detailed neuron models. It offers extensive capabilities for working with very complex distributed biophysical neuronal models and should be a useful addition to the growing ecosystem of tools for neuronal modeling.

Strengths

• This Python-based tool allows for visualization of a neuronal model's compartments.

• The tool works with morphology reconstructions in the widely used .swc and .asc formats.

• It can support many neuronal models using the NMODL language, which is widely used for neuronal modeling.

• It permits one to plot the properties of linear and non-linear conductances in every compartment of a neuronal model, facilitating examination of model's details.

• DendroTweaks supports manipulation of the model parameters and morphological details, which is important for exploration of the relations of the model composition and parameters with its electrophysiological activity.

• The paper is very well written - everything is clear, and the capabilities of the tool are described and illustrated with great attention to details.

Weaknesses

• Not a really big weakness, but it would be really helpful if the authors showed how the performance of their tool scales. This can be done for an increasing number of compartments - how long does it take to carry out typical procedures in DendroTweaks, on a given hardware, for a cell model with 100 compartments, 200, 300, and so on? This information will be quite useful to understand the applicability of the software.

Let me also add here a few suggestions (not weaknesses, but something that can be useful, and if the authors can easily add some of these for publication, that would strongly increase the value of the paper).

• It would be very helpful to add functionality to read major formats in the field, such as NeuroML and SONATA.

• Visualization is available as a static 2D projection of the cell's morphology. It would be nice to implement 3D interactive visualization.

• It is nice that DendroTweaks can modify the models, such as revising the radii of the morphological segments or ionic conductances. It would be really useful then to have the functionality for writing the resulting models into files for subsequent reuse.

• If I didn't miss something, it seems that DendroTweaks supports allocation of groups of synapses, where all synapses in a group receive the same type of Poisson spike train. It would be very useful to provide more flexibility. One option is to leverage the SONATA format, which has ample functionality for specifying such diverse inputs.

• "Each session can be saved as a .json file and reuploaded when needed" - do these files contain the whole history of the session or the exact snapshot of what is visualized when the file is saved? If the latter, which variables are saved, and which are not? Please clarify.

Comments on revisions:

In this revised version of the paper, the authors addressed all my comments. While many of the suggestions were addressed by textual changes in the manuscript or an explanation in the response to the reviewers (rather than adding substantial new functionality to the tool), DendroTweaks in its current updated state does represent an advanced and useful tool. Further extensions can be added as the development of the tool continues, in interaction with the community.

https://doi.org/10.7554/eLife.103324.2.sa1

Author response:

The following is the authors' response to the original reviews.

> ***Reviewer #1 (Public review):***
>
> *Summary:*
>
> *Dendrotweaks provides its users with a solid tool to implement, visualize, tune, validate, understand, and reduce single-neuron models that incorporate complex dendritic arbors with differential distribution of biophysical mechanisms. The visualization of dendritic*

> *segments and biophysical mechanisms therein provide users with an intuitive way to understand and appreciate dendritic physiology.*
>
> *Strengths:*
>
> *(1) The visualization tools are simplified, elegant, and intuitive.*
>
> *(2) The ability to build single-neuron models using simple and intuitive interfaces.*
>
> *(3) The ability to validate models with different measurements.*
>
> *(4) The ability to systematically and progressively reduce morphologically-realistic neuronal models.*
>
> *Weaknesses:*
>
> *(1) Inability to account for neuron-to-neuron variability in structural, biophysical, and physiological properties in the model-building and validation processes.*

We agree with the reviewer that it is important to account for neuron-to-neuron variability. The core approach of DendroTweaks, and its strongest aspect, is the interactive exploration of how morpho-electric parameters affect neuronal activity. In light of this, variability can be achieved through the interactive updating of the model parameters with widgets. In a sense, by adjusting a widget (e.g., channel distribution or kinetics), a user ends up with a new instance of a cell in the parameter space and receives almost real-time feedback on how this change affected neuronal activity. This approach is much simpler than implementing complex optimization protocols for different parameter sets, which would detract from the interactivity aspect of the GUI. In its revised version, DendroTweaks also accounts for neuron-to-neuron morphological variability, as channel distributions are now based on morphological domains (rather than the previous segment-specific approach). This makes it possible to apply the same biophysical configuration across various morphologies. Overall, both biophysical and morphological variability can be explored within DendroTweaks.

> *(2) Inability to account for the many-to-many mapping between ion channels and physiological outcomes. Reliance on hand-tuning provides a single biased model that does not respect pronounced neuron-to-neuron variability observed in electrophysiological measurements.*

We acknowledge the challenge of accounting for degeneracy in the relation between ion channels and physiological outcomes and the importance of capturing neuron-to-neuron variability. One possible way to address this, as we mention in the Discussion, is to integrate automated parameter optimization algorithms alongside the existing interactive hand-tuning with widgets. In its revised version, DendroTweaks can integrate with Jaxley (Deistler et al., 2024) in addition to NEURON. The models created in DendroTweaks can now be run with Jaxley (although not all types of models, see the limitations in the Discussion), and their parameters can be optimized via automated and fast gradient-based parameter optimization, including optimization of heterogeneous channel distributions. In particular, a key advantage of integrating Jaxley with DendroTweaks was its NMODL-to-Python converter, which significantly reduced the need to manually re-implement existing ion channel models for Jaxley (see here: https://dendrotweaks.readthedocs.io/en/latest/tutorials/convert_to_jaxley.html).

(1) Michael Deistler, Kyra L. Kadhim, Matthijs Pals, Jonas Beck, Ziwei Huang, Manuel Gloeckler, Janne K. Lappalainen, Cornelius Schröder, Philipp Berens, Pedro J. Gonçalves, Jakob H. Macke Differentiable simulation enables large-scale training of detailed biophysical models of neural dynamics bioRxiv 2024.08.21.608979; doi:https://doi.org/10.1101/2024.08.21.608979

> *Lack of a demonstration on how to connect reduced models into a network within the toolbox.*

Building a network of reduced models is an exciting direction, yet beyond the scope of this manuscript, whose primary goal is to introduce DendroTweaks and highlight its capabilities. DendroTweaks is designed for single-cell modeling, aiming to cover its various aspects in great detail. Of course, we expect refined single-cell models, both detailed and simplified, to be further integrated into networks. But this does not need to occur within DendroTweaks. We believe this network-building step is best handled by dedicated network simulation platforms. To facilitate the network-building process, we extended the exporting capabilities of DendroTweaks. To enable the export of reduced models in DendroTweaks's modular format, as well as in plain simulator code, we implemented a method to fit the resulting parameter distributions to analytical functions (e.g., polynomials). This approach provided a compact representation, requiring a few coefficients to be stored in order to reproduce a distribution, independently of the original segmentation. The reduced morphologies can be exported as SWC files, standardized ion channel models as MOD files, and channel distributions as JSON files. Moreover, plain NEURON code (Python) to instantiate a cell class can be automatically generated for any model, including the reduced ones. Finally, to demonstrate how these exported models can be integrated into larger simulations, we implemented a "toy" network model in a Jupyter notebook included as an example in the GitHub repository. We believe that these changes greatly facilitate the integration of DendroTweaks-produced models into networks while also allowing users to run these networks on their favorite platforms.

> *(4) Lack of a set of tutorials, which is common across many "Tools and Resources" papers, that would be helpful in users getting acquainted with the toolbox.*

This is an important point that we believe has been addressed fully in the revised version of the tool and manuscript. As previously mentioned, the lack of documentation was due to the software's early stage. We have now added comprehensive documentation, which is available at https://dendrotweaks.readthedocs.io. This extensive material includes API references, 12 tutorials, 4 interactive Jupyter notebooks, and a series of video tutorials, and it is regularly updated with new content. Moreover, the toolbox's GUI with example models is available through our online platform at https://dendrotweaks.dendrites.gr.

> ***Reviewer #2 (Public review):***
>
> *The paper by Makarov et al. describes the software tool called DendroTweaks, intended for the examination of multi-compartmental biophysically detailed neuron models. It offers extensive capabilities for working with very complex distributed biophysical neuronal models and should be a useful addition to the growing ecosystem of tools for neuronal modeling.*
>
> *Strengths*
>
> *(1) This Python-based tool allows for visualization of a neuronal model's compartments.*
>
> *(2) The tool works with morphology reconstructions in the widely used .swc and .asc formats.*
>
> *(3) It can support many neuronal models using the NMODL language, which is widely used for neuronal modeling.*
>
> *(4) It permits one to plot the properties of linear and non-linear conductances in every compartment of a neuronal model, facilitating examination of the model's details.*

> *(5) DendroTweaks supports manipulation of the model parameters and morphological details, which is important for the exploration of the relations of the model composition and parameters with its electrophysiological activity.*
>
> *(6) The paper is very well written - everything is clear, and the capabilities of the tool are described and illustrated with great attention to detail.*
>
> *Weaknesses*
>
> *(1) Not a really big weakness, but it would be really helpful if the authors showed how the performance of their tool scales. This can be done for an increasing number of compartments - how long does it take to carry out typical procedures in DendroTweaks, on a given hardware, for a cell model with 100 compartments, 200, 300, and so on? This information will be quite useful to understand the applicability of the software.*

DendroTweaks functions as a layer on top of a simulator. As a result, its performance scales in the same way as for a given simulator. The GUI currently displays the time taken to run a simulation (e.g., in NEURON) at the bottom of the Simulation tab in the left menu. While Bokeh-related processing and rendering also consume time, this is not as straightforward to measure. It is worth noting, however, that this time is short and approximately equivalent to rendering the corresponding plots elsewhere (e.g., in a Jupyter notebook), and thus adds negligible overhead to the total simulation time.

> *(2) Let me also add here a few suggestions (not weaknesses, but something that can be useful, and if the authors can easily add some of these for publication, that would strongly increase the value of the paper).*
>
> *(3) It would be very helpful to add functionality to read major formats in the field, such as NeuroML and SONATA.*

We agree with the reviewer that support for major formats will substantially improve the toolbox, ensuring the reproducibility and reusability of the models. While integration with these formats has not been fully implemented, we have taken several steps to ensure elegant and reproducible model representation. Specifically, we have increased the modularity of model components and developed a custom compact data format tailored to single-cell modeling needs. We used a JSON representation inspired by the Allen Cell Types Database schema, modified to account for non-constant distributions of the model parameters. We have transitioned from a representation of parameter distributions dependent on specific segmentation graphs and sections to a more generalized domain-based distribution approach. In this revised methodology, segment groups are no longer explicitly defined by segment identifiers, but rather by specification of anatomical domains and conditional expressions (e.g., “select all segments in the apical domain with the maximum diameter < 0.8 μm”). Additionally, we have implemented the export of experimental protocols into CSV and JSON files, where the JSON files contain information about the stimuli (e.g., synaptic conductance, time constants), and the CSV files store locations of recording sites and stimuli. These features contribute toward a higher-level, structured representation of models, which we view as an important step toward eventual compatibility with standard formats such as NeuroML and SONATA. We have also initiated a two-way integration between DendroTweaks and SONATA. We developed a converter from DendroTweaks to SONATA that automatically generates SONATA files to reproduce models created in DendroTweaks. Additionally, support for the DendroTweaks JSON representation of biophysical properties will be added to the SONATA data format ecosystem, enabling models with complex dendritic distributions of channels. This integration is still in progress and will be included in the next version of DendroTweaks. While full integration with these formats is a goal for future releases, we

believe the current enhancements to modularity and exportability represent a significant step forward, providing immediate value to the community.

> *(4) Visualization is available as a static 2D projection of the cell's morphology. It would be nice to implement 3D interactive visualization.*

We offer an option to rotate a cell around the Y axis using a slider under the plot. This is a workaround, as implementing a true 3D visualization in Bokeh would require custom Bokeh elements, along with external JavaScript libraries. It's worth noting that there are already specialized tools available for 3D morphology visualization. In light of this, while a 3D approach is technically feasible, we advocate for a different method. The core idea of DendroTweaks' morphology exploration is that each section is "clickable", allowing its geometric properties to be examined in a 2D "Section" view. Furthermore, we believe the "Graph" view presents the overall cell topology and distribution of channels and synapses more clearly.

> *(5) It is nice that DendroTweaks can modify the models, such as revising the radii of the morphological segments or ionic conductances. It would be really useful then to have the functionality for writing the resulting models into files for subsequent reuse.*

This functionality is fully available in local installations. Users can export JSON files with channel distributions and SWC files after morphology reduction through the GUI. Please note that for resource management purposes, file import/export is disabled on the public online demo. However, it can be enabled upon local installation by modifying the configuration file (app/default_config.json). In addition, it is now possible to generate plain NEURON (Python) code to reproduce a given model outside the toolbox (e.g., for network simulations). Moreover, it is now possible to export the simulation protocols as CSV files for locations of stimuli and recordings and JSON files for stimuli parameters.

> *(6) If I didn't miss something, it seems that DendroTweaks supports the allocation of groups of synapses, where all synapses in a group receive the same type of Poisson spike train. It would be very useful to provide more flexibility. One option is to leverage the SONATA format, which has ample functionality for specifying such diverse inputs.*

Currently, each population of "virtual" neurons that form synapses on the detailed cell shares the same set of parameters for both biophysical properties of synapses (e.g., reversal potential, time constants) and presynaptic "population" activity (e.g., rate, onset). The parameter that controls an incoming Poisson spike train is the rate, which is indeed shared across all synapses in a population. Unfortunately, the current implementation lacks the capability to simulate complex synaptic inputs with heterogeneous parameters across individual synapses or those following non-uniform statistical distributions (the present implementation is limited to random uniform distributions). We have added this information in the Discussion (3. Discussion - 3.2 Limitations and future directions - ¶.5) to make users aware of the limitations. As it requires a substantial amount of additional work, we plan to address such limitations in future versions of the toolbox.

> *(7) "Each session can be saved as a .json file and reuploaded when needed" - do these files contain the whole history of the session or the exact snapshot of what is visualized when the file is saved? If the latter, which variables are saved, and which are not? Please clarify.*

In the previous implementation, these files captured the exact snapshot of the model's latest state. In the new version, we adopted a modular approach where the biophysical configuration (e.g., channel distributions) and stimulation protocols are exported to separate

files. This allows the user to easily load and switch the stimulation protocols for a given model. In addition, the distribution of parameters (e.g., channel conductances) is now based on the morphological domains and is agnostic of the exact morphology (i.e., sections and segments), which allows the same JSON files with biophysical configurations to be reused across multiple similar morphologies. This also allows for easy file exchange between the GUI and the standalone version.

> ***Joint recommendations to Authors:***
>
> *The reviewers agreed that the paper is well written and that DendroTweaks offers a useful collection of tools to explore models of single-cell biophysics. However, the tooling as provided with this submission has critical limitations in the capabilities, accessibility, and documentation that significantly limit the utility of DendroTweaks. While we recognize that it is under active development and features may have changed already, we can only evaluate the code and documentation available to us here.*

We thank the reviewers for their positive evaluation of the manuscript and express our sincere appreciation for their feedback. We acknowledge the limitations they have pointed out and have addressed most of these concerns in our revised version.

> *In particular, we would emphasize:*
>
> *(1) While the features may be rich, the documentation for either a user of the graphical interface or the library is extremely sparse. A collection of specific tutorials walking a GUI user through simple and complex model examples would be vital for genuine uptake. As one category of the intended user is likely to be new to computational modeling, it would be particularly good if this documentation could also highlight known issues that can arise from the naive use of computational techniques. Similarly, the library aspect needs to be documented in a more standard manner, with docstrings, an API function list, and more didactic tutorials for standard use cases.*

DendroTweaks now features comprehensive documentation. The standalone Python library code is well-documented with thorough docstrings. The overall code modularity and readability have improved. The documentation is created using the widely adopted Sphinx generator, making it accessible for external contributors, and it is available via ReadTheDocs https://dendrotweaks.readthedocs.io/en/latest/index.html. The documentation provides a comprehensive set of tutorials (6 basic, 6 advanced) covering all key concepts and workflows offered by the toolbox. Interactive Jupyter notebooks are included in the documentation, along with the quick start guide. All example models also have corresponding notebooks that allow users to build the model from scratch.

The toolbox has its own online platform, where a quick-start guide for the GUI is available https://dendrotweaks.dendrites.gr/guide.html. We have created video tutorials for the GUI covering the basic use cases. Additionally, we have added tips and instructions alongside widgets in the GUI, as well as a status panel that displays application status, warnings, and other information. Finally, we plan to familiarize the community with the toolbox by organizing online and in-person tutorials, as the one recently held at the CNS*2025 conference (https://cns2025florence.sched.com/event/25kVa/building-intuitive-and-efficient-biophysicalmodels-with-jaxley-and-dendrotweaks). Moreover, the toolbox was already successfully used for training young researchers during the Taiwan NeuroAI 2025 Summer School, founded by Ching-Lung Hsu. The feedback was very positive.

> *(2) The paper describes both a GUI web app and a Python library. However, the code currently mixes these two in a way that largely makes sense for the web app but makes it very difficult to use the library aspect. Refactoring the code to separate apps and*

> *libraries would be important for anyone to use the library as well as allowing others to host their own DendroTweak servers. Please see the notes from the reviewing editor below for more details.*

The code in the previous àpp/model ŧolder, responsible for the core functionality of the toolbox, has been extensively refactored and extended, and separated into a standalone library. The library is included in the Python package index (PyPI, https://pypi.org/project/dendrotweaks).

> *Notes from the Reviewing Editor Comments (Recommendations for the authors):*
>
> *(1) While one could import morphologies and use a collection of ion channel models, details of synapse groups and stimulation approaches appeared to be only configurable manually in the GUI. The ability to save and load full neuron and simulation states would be extremely useful for reproducibility and sharing data with collaborators or as an interactive data product with a publication. There is a line in the text about saving states as json files (also mentioned by Reviewer #2), but I could see no such feature in the version currently online.*

We decided to reserve the online version for demonstration and educational purposes, with more example models being added over time. However, this functionality is available upon local installation of the app (and after specifying it in the 'default_config.json' in the root directory of the app). We've adopted a modular model representation to store separately morphology, channel models, biophysical parameters, and stimulation protocols.

> *(2) Relatedly, GUI exploration of complex data is often a precursor to a more automated simulation run. An easy mechanism to go from a user configuration to scripting would be useful to allow the early strength of GUIs to feed into the power of large-scale scripting.*

Any model could be easily exported to a modular DendroTweaks representation and later imported either in the GUI or in the standalone version programmatically. This ensures a seamless transition between the two use cases.

> *(3) While the paper discusses DendroTweaks as both a GUI and a python library, the zip file of code in the submission is not in good form as a library. Back-end library code is intermingled with front-end web app code, which limits the ability to install the library from a standard python interface like PyPI. API documentation is also lacking. Functions tend to not have docstrings, and the few that do, do not follow typical patterns describing parameters and types.*

As stated above, all these issues have been resolved in the new version of the toolbox. The library code is now housed in a separate repository https://github.com/Poirazi-Lab/DendroTweaks and included in PyPI https://pypi.org/project/dendrotweaks. The classes and public methods follow Numpy-style docstrings, and the API reference is available in the documentation: https://dendrotweaks.readthedocs.io/en/latest/genindex.html.

> *(4) Library installation is very difficult. The requirements are currently a lockfile, fully specifying exact versions of all dependencies. This is exactly correct for web app deployment to maintain consistency, but is not feasible in the context of libraries where you want to have minimal impact on a user's environment. Refactoring the library from the web app is critical for making DendroTweaks usable in both forms described in the paper.*

> *The lockfile makes installation more or less impossible on computer setups other than that of the author. Needless to say, this is not acceptable for a tool, and I would encourage the authors to ask other people to attempt to install their code as they describe in the text. For example, attempting to create a conda environment from the environment.yml file on an M1 MacBook Pro failed because it could not find several requirements. I was able to get it to install within a Linux docker image with the x86 platform specified, but this is not generally viable. To make this be the tool it is described as in text, this must be resolved. A common pattern that would work well here is to have a requirements lockfile and Docker image for the web app that imports a separate, more minimally restrictive library package with that could be hosted on PyPI or, less conveniently, through conda-forge.*

The installation of the standalone library is now straightforward via pip install dendrotweaks.On the Windows platform, however, manual installation of NEURON is required as described in the official NEURON documentation https://nrn.readthedocs.io/en/8.2.6/install/install_instructions.html#windows.

> *(5) As an aside, to improve potential uptake, the authors might consider an MIT-style license rather than the GNU Public License unless they feel strongly about the GPL. Many organizations are hesitant to build on GPL software because of the wide-ranging demands it places on software derived from or using GPL code.*

We thank the editor for this suggestion. We are considering changing the licence to MPL 2.0. It will maintain copyleft restrictions only on the package files while allowing end-users to freely choose their own license for any derived work, including the models, generated data files, and code that simply imports and uses our package.

> ***Reviewer #1 (Recommendations for the authors):***
>
> *(1) Abstract: Neurons rely on the interplay between dendritic morphology and ion channels to transform synaptic inputs into a sequence of somatic spikes. Technically, this would have to be morphology, ion channels, pumps, transporters, exchangers, buffers, calcium stores, and other molecules. For instance, if the calcium buffer concentration is large, then there would be less free calcium for activating the calcium-activated potassium channels. If there are different chloride co-transporters - NKCC vs. KCC - expressed in the neuron or different parts of the neuron, that would alter the chloride reversal for all the voltage- or ligand-gated chloride channels in the neuron. So, while morphology and ion channels are two important parts of the transformation, it would be incorrect to ignore the other components that contribute to the transformation. The statement might be revised to make these two components as two critical components.*

The phrase “Two critical components” was added as it was suggested by the reviewer.

> *(2) Section 2.1 - The overall GUI looks intuitive and simple.*
>
> *(3) Section 2.2*
>
> *(a) The Graph view of morphology, especially accounting for the specific d_lambda is useful.*
>
> *(b) "Note that while microgeometry might not significantly affect the simulation at a low spatial resolution (small number of segments) due to averaging, it can introduce unexpected cell behavior at a higher level of spatial discretization."*

> *It might be good to warn the users that the compartmentalization and error analyses are with reference to the electrical lambda. If users have to account for calcium microdomains, these analyses wouldn't hold given the 2 orders of magnitude differences between the electrical and the calcium lambdas (e.g., Zador and Koch, J Neuroscience, 1994). Please sensitize users that the impact of active dendrites in regulating calcium microdomains and signaling is critical when it comes to plasticity models in morphologically realistic structures.*

We thank the reviewer for this important point. We have clarified in the text that our spatial discretization specifically refers to the electrical length constant. We acknowledge that electrical and chemical processes operate on fundamentally different spatial and temporal scales, which requires special consideration when modeling phenomena like synaptic plasticity. We have sensitized users about this distinction. However, we do not address such examples in the manuscript, thus leaving the detailed discussion of non-electrical compartmentalization beyond the scope of this work.

> *(c) I am not very sure if the "smooth" tool for diameters that is illustrated is useful. Users shouldn't consider real variability in morphology as artifacts of reconstruction. As mentioned above, while this might not be an issue with electrical compartmentalization, calcium compartmentalization will severely be affected by small changes in morphology. Any model that incorporates calcium-gated channels should appropriately compartmentalize calcium. Without this, the spread of activation of calcium-dependent conductances would be an overestimate. Even small changes in cellular shape and curvature can have large impacts when it comes to signaling in terms of protein aggregation and clustering.*

Although this functionality is still available in the toolbox, we have removed the emphasis from it in the manuscript. Nevertheless, for the purpose of addressing the reviewer's comment, we provide an example when this "smoothening" might be needed:please see Figure S1 from Tasciotti et al. 2025.

(2) Simone Tasciotti, Daniel Maxim Iascone, Spyridon Chavlis, Luke Hammond, Yardena Katz, Attila Losonczy, Franck Polleux, Panayiota Poirazi. From Morphology to Computation: How Synaptic Organization Shapes Place Fields in CA1 Pyramidal Neurons bioRxiv 2025.05.30.657022; doi: https://doi.org/10.1101/2025.05.30.657022

> *(4) Section 2.3*
>
> *(a) The graphical representation of channel gating kinetics is very useful.*
>
> *(b) Please warn the users that experimental measurements of channel gating kinetics are extremely variable. Taking the average of the sigmoids or the activation/deactivation/inactivation kinetics provides an illusion that each channel subtype in a given cell type has fixed values of V_1/2, k, delta, and tau, but it is really a range obtained from several experiments. The heterogeneity is real and reflects cell-to-cell variability in channel gating kinetics, not experimental artifacts. Please sensitize the readers that there is not a single value for these channel parameters.*

This is a fair comment, and it refers to a general problem in neuronal modeling. In DendroTweaks, we follow the approach widely used in the community that indeed doesn't account for heterogeneity. We added a paragraph in the revised manuscript's Discussion (3. Discussion - 3.3 Limitations and future directions - ¶.3) to address this issue.

> *(5) Section 2.4*

> *(a) Same as above: Please sensitize users that the gradients in channel conductances are measured as an average of measurements from several different cells. This gradient need not be present in each neuron, as there could be variability in location-dependent measurements across cells. The average following a sigmoid doesn't necessarily mean that each neuron will have the channel distributed with that specific sigmoid (or even a sigmoid!) with the specific parametric values that the average reported. This is extremely important because there is an illusion that the gradient is fixed across cells and follows a fixed functional form.*

We added this information to our Discussion in the same paragraph mentioned above.

> *(b) Please provide an example where the half-maximal voltage of a channel varies as a function of distance (such as Poolos et al., Nature Neuroscience, 2002 or Migliore et al., 1999; Colbert and Johnston, 1997). This might require a step-like function in some scenarios. An illustration would be appropriate because people tend to assume that channel gating kinetics are similar throughout the dendrite. Again, please mention that these shifts are gleaned from the average and don't really imply that each neuron must have that specific gradient, given neuron-to-neuron variability in these measurements.*

We thank the reviewer for the provided literature, which we now cite when describing parameter distributions (2. Results - 2.4 Distributing ion channels - ¶.1). Please note that DendroTweaks' programming interface and data format natively support non-linear distribution of kinetic parameters alongside the channel conductances. As for the step-like function, users can either directly apply the built-in step-like distribution function or create it by combining two constant distributions.

> *(6) Section 2.5*
>
> *(a) It might be useful to provide a mechanism for implementing the normalization of unitary conductances at the cell body, (as in Magee and Cook, 2000; Andrasfalvy et al., J Neuroscience, 2001). Specifically, users should be able to compute AMPAR conductance values at each segment which would provide a somatic EPSP value of 0.2 mV.*

This functionality is indeed useful and will be added in future releases. Currently, it has been mentioned in the list of known limitations when working with synaptic inputs (3. Discussion - 3.3 Limitations and future directions - ¶.5).

> *(b) Users could be sensitized about differences in decay time constants of GABA_A receptors that are associated with parvalbamin vs. somatostatin neurons. As these have been linked to slow and fast gamma oscillations and different somatodendritic locations along different cell types, this might be useful (e.g., 10.1016/j.neuron.2017.11.033;10.1523/jneurosci.0261-20.2020; 10.7554/eLife.95562.1; 10.3389/fncel.2023.1146278).*

We thank the reviewer for highlighting this important biological detail. DendroTweaks enables users to define model parameters specific to their cell type of interest. For practical reasons, we leave the selection of biologically relevant parameters to the users. However, we will consider adding an explicit example in our tutorials to showcase the toolbox's flexibility in this regard.

> *(7) Section 2.6*
>
> *While reducing the morphological complexity has its advantages, users of this tool should be sensitized in this section about how the reduction does not capture all the*

> *complexity of the dendritic computation. For instance, the segregation/amplification properties of Polsky et al., 2004, Larkum et al., 2009 would not be captured by a fully reduced model. An example across different levels of reductions, implementing simulations in Figure 7F (but for synapses on the same vs. different branches), would be ideal. Demonstrate segregation/amplification in the full model for the same set of synapses - coming on the same branch/different branch (linear integration of synapses on different branches and nonlinear integration of synapses on the same branch). Then, show that with different levels of reduction, this segregation/amplification vanishes in the reduced model. In addition, while impedance-based approaches account for account for electrical computation, calcium-based computation is not something that is accountable with reduced models, given the small lambda_calcium values. Given the importance of calcium-activated conductances in electrical behaviour, this becomes extremely important to account for and sensitize users to. The lack of such sensitization results in presumptuous reductions that assume that all dendritic computation is accounted for by reduced models!*

We agree with the reviewer that reduction leads to a loss in the complexity of dendritic computation. This has been stated in both the original algorithm paper (Amsalem et al., 2020) and in our manuscript (e.g., 3. Discussion - 3.2 Comparison to existing modeling software - ¶.6). In fact, to address this problem, we extended the functionality of neuron_reduce to allow for multiple levels of morphology reduction. Our motivation for integrating morphology reduction in the toolbox was to leverage the exploratory power of DendroTweaks to assess how different degrees of reduction alter cell integrative properties, determining which computations are preserved, which are lost, and at what specific reduction level these changes occur. Nevertheless, to address this comment, we've made it more explicit in the Discussion that reduction inevitably alters integrative properties and, at a certain level, leads to loss of dendritic computations.

> *(8) Section 2.7*
>
> *(a) The validation process has two implicit assumptions:*
>
> *(i) There is only one value of physiological measurements that neurons and dendrites are endowed with. The heterogeneity in these measurements even within the same cell type is ignored. The users should be allowed to validate each measurement over a range rather than a single value. Users should be sensitized about the heterogeneity of physiological measurements.*
>
> *(ii) The validation process is largely akin to hand-tuning models where a one-to-one mapping of channels to measurements is assumed. For instance, input resistance can be altered by passive properties, by Ih, and by any channel that is active under resting conditions. Firing rate and patterns can be changed by pretty much every single ion channel that expresses along the somatodendritic axis.*
>
> *An updated validation process that respects physiological heterogeneities in measurements and accounts for global dependencies would be more appropriate. Please update these to account for heterogeneities and many-to-many mappings between channels and measurements. An ideal implementation would be to incorporate randomized search procedures (across channel parameters spanning neuron-to-neuron variability in channel conductances/gating properties) to find a population of models that satisfy all physiological constraints (including neuron-to-neuron variability in each physiological measurement), rather than reliance on procedures that are akin to hand-tuning models. Such population-based approaches are now common across morphologically-realistic models for different cell types (e.g., Rathour and Narayanan, PNAS, 2014; Basak and Narayanan, J Physiology, 2018; Migliore et al., PLoS*

> *Computational Biology, 2018; Basak and Narayanan, Brain Structure and Function, 2020; Roy and Narayanan, Neural Networks, 2021; Roy and Narayanan, J Physiology, 2023; Arnaudon et al., iScience, 2023; Reva et al., Patterns, 2023; Kumari and Narayanan, J Neurophysiology, 2024) and do away with the biases introduced by hand-tuning as well as the assumption of one-to-one mapping between channels and measurements.*

We appreciate the reviewer's comment and the suggested alternatives to our validation process. We have extended the discussion on these alternative approaches (3. Discussion - 2. Comparison to existing modeling software - ¶.5). However, it is important to note that neither one-value nor one-to-one mapping assumption is imposed in our approach. It is true that validation is performed on a given model instance with fixed single-value parameters. However, users can discover heterogeneity and degeneracy in their models via interactive exploration. In the GUI, a given parameter can be changed, and the influence of this change on model output can be observed in real time. Validation can be run after each change to see whether the model output still falls within a biologically plausible regime or not. This is, of course, time-consuming and less efficient than any automated parameter optimization.

However, and importantly, this is the niche of DendroTweaks. The approach we provide here can indeed be referred to as model hand-tuning. This is intentional: we aim to complement black-box optimization by exposing the relationship between parameters and model outputs. DendroTweaks is not aimed at automated parameter optimization and is not meant to provide the user with parameter ranges automatically. The built-in validation in DendroTweaks is intended as a lightweight, fast feedback tool to guide manual tuning of dendritic model parameters so as to enhance intuitive understanding and assess the plausibility of outputs, not as a substitute for comprehensive model validation or optimization. The latter can be done using existing frameworks, designed for this purpose, as mentioned by the reviewer.

> *(b) Users could be asked to wait for RMP to reach steady state. For instance, in some of the traces in Figure 7, the current injection is provided before RMP reaches steady-state. In the presence of slow channels (HCN or calcium-activated channels), the RMP can take a while to settle down. Users might be sensitized about this. This would also bring to attention the ability of several resting channels in modulating RMP, and the need to wait for steady-state before measurements are made.*

We agree with the observation and updated the validation process accordingly. We have added functionality for simulation stabilization, allowing users to pre-run a simulation before the main simulation time. For example, model.run(duration=1000, prerun_time=300) could be used to stabilize the model for a period of 300 ms before running the main simulation for 1 s.

> *(c) Strictly speaking, it is incorrect to obtain membrane time constant by fitting a single exponential to the initial part of the sag response (Figure 7A). This may be confirmed in the model by setting HCN to zero (strictly all active channel conductances to zero), obtaining the voltage-response to a pulse current, fitting a double exponential (as Rall showed, for a finite cable or for a real neuron, a single exponential would yield incorrect values for the tau) to the voltage response, and mapping membrane time constant to the slower of the two time-constants (in the double exponential fit). This value will be very different from what is obtained in Figure 7A. Please correct this, with references to Rall's original papers and to electrophysiological papers that use this process to assess membrane properties of neurons and their dendrites (e.g., Stuart and Spruston, J Neurosci, 1998; Golding and Spruston, J Physiology, 2005).*

We updated the algorithm for calculating the membrane time constant based on the reviewer's suggestions and added the suggested references. The time constant is now obtained in a model with blocked HCN channels (setting maximal conductance to 0) via a double exponential fit, taking the slowest component.

> *(9) Section 3*
>
> *(a) May be good to emphasize the many-to-many mapping between ion channels and neuronal functions here in detail, and on how to explore this within the Dendrotweaks framework.*

We have added a paragraph in the Discussion that addresses both the problems of heterogeneity and degeneracy in biological neurons and neuronal models (3. Discussion - 3.3 Limitations and future directions - ¶.3)

> *(b) May be good to have a specific section either here or in results about how the different reduced models can actually be incorporated towards building a network.*

As mentioned earlier, building a network of reduced models is a promising new direction. However, it is beyond the scope of this manuscript, whose primary goal is to introduce DendroTweaks and highlight its capabilities. DendroTweaks is designed for single-cell modeling and provides export capabilities that allow integrating it into broader workflows, including network modeling. We have added a paragraph in the manuscript (3. Discussion - 3.1 Conceptual and implementational accessibility - ¶.2) that addresses how DendroTweaks could be used alongside other software, in particular for scaling up single-cell models to the network level.

> *(10) Section 4*
>
> *(a) Section 4.3: In the second sentence (line 568), the "first Kirchhoff's law" within parentheses immediately after Q=CV gives an illusion that Q=CV is the first Kirchhoff's law! Please state that this is with reference to the algebraic sum of currents at a node.*

We have corrected the equations and apologize for this oversight.

> *(b) Table 1: In the presence of active ion channels, input resistance, membrane time constant, and voltage attenuation are not passive properties. Input resistance is affected by any active channel that is active at rest (HCN, Kir, A-type K+ through the window current, etc). The same holds for membrane time constant and voltage attenuation as well. This could be made clear by stating if these measurements are obtained in the presence or absence of active ion channels. In real neurons, all these measurements are affected by active ion channels; so, ideally, these are also active properties, not passive! Also, please mention that in the presence of resonating channels (e.g., HCN, M-type K+), a single exponential fit won't be appropriate to obtain tau, given the presence of sag.*

We thank the reviewer for pointing out this ambiguity. What the term “Passive” means in Table 1 (e.g., for the input resistance, R_in) is that the minimal set of parameters needed to validate R_in are the passive ones (i.e., Cm, Ra, and Leak). We have changed the table listing to reflect this.

> ***Reviewer #2 (Recommendations for the authors):***
>
> *(1) Figure 2B and the caption to Figure 2F show and describe the diameter of the sections, whereas the image in Figure 2F shows the radius. Which is the correct one?*

The reason for this is that Figure 2B shows the sections' geometry as it is represented in NEURON, i.e., with diameters, while Figure 2F shows the geometry as it is represented in an SWC file (as these changes are made based on the SWC file). Nevertheless, as mentioned earlier, we decided to remove panel F from the figure in the new version, to present a more important panel on tree graph representations.

> *(2) "Each segment can be viewed as an equivalent RC circuit representing a part of the membrane". The example in Figure 2B is perhaps a relatively simple case. For more complex cases where multiple nonlinear conductances are present in each section, would it be possible to show each of these conductances explicitly? If yes, it would be nice to illustrate that.*

We would like to clarify that "can be viewed" here was intended to mean "can be considered," and we have updated the text accordingly. The schematic RC circuits were added to the corresponding figure for illustration purposes only and are not present in the GUI, as this would indeed be impractical for multiple conductances.

> *(3) Some extra citations could be added. For example, it is a little strange that BRIAN2 is mentioned, but NEST is not. It might be worth mentioning and citing it. Also, the Allen Cell Types Database is mentioned, but no citation for it is given. It could be useful to add such citations (https://doi.org/10.1038/s41593-019-0417-0, https://doi.org/10.1038/s41467-017-02718-3).*

Brian 2 is extensively used in our lab on its own and as a foundation of the Dendrify library (Pagkalos et al., 2023). As stated in the discussion, we are considering bridging reduced Hodgkin-Huxley-type models to Dendrify leaky integrate-and-fire type models. For these reasons, Brian 2 is mentioned in the discussion. However, we acknowledge that our previous overview omitted references to some key software, which have now been added to the updated manuscript. We appreciate the reviewer providing references that we had overlooked.

(3) Pagkalos, M., Chavlis, S. & Poirazi, P. Introducing the Dendrify framework for incorporating dendrites to spiking neural networks. Nat Commun 14, 131 (2023). https://doi.org/10.1038/s41467-022-35747-8

https://doi.org/10.7554/eLife.103324.2.sa0