Systematic comparison and automated validation of detailed models of hippocampal neurons

Sára Sáray[1,2*], Christian A. Rössert[3], Shailesh Appukuttan[4], Rosanna Migliore[5], Paola Vitale[5], Carmen A. Lupascu[5], Luca L. Bologna[5], Werner Van Geit[3], Armando Romani[3], Andrew P. Davison[4], Eilif Muller[3], Tamás F. Freund[1,2], Szabolcs Káli[1,2*]

[1]Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, Budapest, Hungary

[2] Institute of Experimental Medicine, Budapest, Hungary,

[3] Blue Brain Project, École Polytechnique Fédérale de Lausanne, Geneva, Switzerland,

[4]Paris-Saclay Institute of Neuroscience, Centre National de la Recherche Scientifique/ Université Paris-Saclay, Gif-sur-Yvette, France,

[5]Institute of Biophysics, National Research Council, Palermo, Italy.

[*] Corresponding author: saray.sara@koki.mta.hu (SS), kali@koki.hu (SK)

Abstract

Anatomically and biophysically detailed data-driven neuronal models can be useful tools in understanding and predicting the behavior and function of neurons. Due to the increasing availability of experimental data from anatomical and electrophysiological measurements as well as the growing number of computational and software tools that enable accurate neuronal modeling, there are now a large number of different models of many cell types available in the literature. These models were usually built to capture a few important or interesting properties of the given neuron type, and it is often unknown how they would behave outside their original context. This limits the re-use and further development of the existing models, and thus prevents the building of consensus “community models” that could capture an increasing proportion of the electrophysiological properties of the given cell type. We addressed this problem for the representative case of the CA1 pyramidal cell of the rat hippocampus by developing an open-source Python test suite, which makes it possible to automatically and systematically test the generalization properties of models by making quantitative comparisons between the models and electrophysiological data. The tests cover various aspects of somatic behavior, and signal propagation and integration in apical dendrites. To demonstrate the utility of our approach, we applied our validation tests to compare the behavior of several different hippocampal CA1 pyramidal cell models from the ModelDB database against electrophysiological data available in the literature, and concluded that all of these models perform well in some domains but badly in others. We also show how we employed the test suite to aid the development of models within the European Human Brain Project (HBP), and describe the integration of the tests into the validation framework developed

in the HBP, with the aim of facilitating more reproducible and transparent community model building.

Author summary

Anatomically and biophysically detailed neuronal models are useful tools in neuroscience because they allow the prediction of the behavior and the function of the studied cell type under circumstances that are hard to investigate experimentally. However, most detailed biophysical models have been built to capture only a few properties of the real neuron, and it is often unknown how they would behave under different circumstances, or whether they can be used to successfully answer different scientific questions. To help the modeling community develop neural models that generalize better, and make the process of model building more reproducible and transparent, we developed a test suite that enables the comparison of the behavior of models of neurons in the rat hippocampus and their evaluation against experimental data. Applying our tests to several models available in the literature, we show that each model is able to capture some of the important properties of the real neuron but performs badly in other domains. We also use the test suite in the model development workflow of the European Human Brain Project to aid the construction of better models of hippocampal neurons and networks.

Introduction

The construction and simulation of anatomically and biophysically detailed models is becoming a standard tool in neuroscience [1]. Such models, which typically employ the compartmental modeling approach and a Hodgkin-Huxley-type description of voltage-gated ion channels, are capable of providing fairly accurate models of single neurons [2–9] and (when complemented by appropriate models of synaptic interactions) even large-scale circuits [10–13]. However, building such detailed multi-compartmental models of neurons requires setting a large number of parameters (such as the densities of various ion channels in multiple neuronal compartments) that are often not directly constrained by the available experimental data. These parameters are typically tuned (either manually or using automated parameter-search methods [9,14–16]) until the simulated physiological behavior of the model matches some pre-defined set of experimental observations.

For an increasing number of cell types, the available experimental data already provide diverse constraints on the expected physiological behavior of the neuron under a variety of conditions. However, only a small subset of these constraints is typically taken into account when the models are developed, and it is often unknown, even by their developers, how these models would behave in other situations, outside their original context. This sparsity of information about the performance of detailed models might be one reason why model re-use in the community is relatively limited, and there are often a large number of different models of the same cell type available in the literature that were developed for different purposes. As an example, there are currently 129 different models related to the hippocampal CA1 pyramidal cell (PC) in the ModelDB database [17]. In addition, even when models are re-used, they are often altered to fit a different subset of the available experimental data, and they may lose their ability to capture the behaviors that were used to constrain the original model. This phenomenon (whereby introducing new features breaks previously correct behavior) is known as a "regression" in software development, and is typically avoided by regularly applying a set of tests that comprehensively verify the correct behavior of the software under various circumstances. Such comprehensive checks are not routinely performed when neural models are developed – and this may be one of the reasons why the development of consensus (community) models, which would aim to capture a wide range of experimental observations by integrating diverse efforts, has rarely been attempted in neuroscience.

A collaborative approach to modeling, and even a systematic comparison of existing models built in different laboratories, requires the development of a comprehensive validation suite, a set of automated tests that quantitatively compare various aspects of model behavior with the corresponding experimental data. Such validation suites enable all modeling groups to evaluate their existing and newly developed models according to common, standardized criteria, thus facilitating model comparison and providing an objective measure of progress in matching relevant experimental observations. Applying automated tests also allows researchers to learn more about models published by other groups (beyond the results included in the papers) with relatively little effort, thus facilitating optimal model re-use and co-operative model development. Systematic testing of models during their development also helps avoid regressions, aids the identification of problematic aspects of model behavior, and is thus expected to lead to an increased efficiency in developing good models. The technical framework for developing such test suites already exists [18], and is currently used by several groups to create a variety of tests for models of neural structure and function at different scales [19–23]. In the current study, our goal was to develop a validation suite for the physiological behavior of one of the most studied cell types of the mammalian brain, the pyramidal cell in area CA1 of the rat hippocampus.

CA1 pyramidal neurons display a large repertoire of nonlinear responses in all of their compartments (including the soma, axon, and various functionally distinct parts of the dendritic tree), which are experimentally well-characterized. In particular, there are detailed quantitative results available on the subthreshold and spiking voltage response to somatic current injections [3,24]; on the properties of the action potentials back-propagating from the soma into the dendrites [25–27], which is a basic measure of dendritic excitability; and on the characteristics of the spread [28] and non-linear integration of synaptically evoked signals in the dendrites, including the conditions necessary for the generation of dendritic spikes [29–32].

The test suite that we have developed allows the quantitative comparison of the behavior of anatomically and biophysically detailed models of CA1 pyramidal neurons with experimental data in all of these domains. In this paper, we first describe the implementation of the HippoUnit validation suite. Next, we show how we used this test suite to systematically compare existing models from six prominent publications from different laboratories. We then show an example of how the tests have been applied to aid the development of new models in the context of the European Human Brain Project (HBP). Finally, we describe the integration of our test suite into the general validation framework developed in the HBP.

Methods

Implementation of HippoUnit

HippoUnit is a Python test suite based on the SciUnit [18] framework, which is a Python package for testing scientific models, and during its implementation the NeuronUnit package [19] was taken into account as an example of how to use the SciUnit framework for testing neuronal models. In SciUnit tests usually four main classes are implemented: the test class, the model class, the capabilities class and the score class. HippoUnit is built in a way that keeps this structure. The key idea behind this structure is the decoupling of the model implementation from the test implementation by defining standardized interfaces (capabilities) between them, so that tests can easily be used with different models without being rewritten, and models can easily be adapted to fit the framework.

Each test of HippoUnit is a separate Python class that, similarly to other SciUnit packages, can run simulations on the models to generate model *predictions*, which can be compared with experimental *observations* to yield the final score, provided that the model has the required capabilities implemented to mimic the appropriate experimental protocol and produce the same type of measurable output. All measured or calculated data that contribute to the final score are saved in JSON or pickle files (or, in many cases, in both types of files). JSON files are human readable, and can be easily loaded into Python dictionaries. Data with a more complex structure are saved into pickle files. This makes it possible to easily write and read the data (for further processing or analysis) without changing its Python structure, no matter what type of object or variable it is.

Similarly to many of the existing SciUnit packages the implementations of specific models are not part of the HippoUnit package itself. Instead, HippoUnit contains a general `ModelLoader` class. This class is implemented in a way that it is able to load and deal with most types of models defined in the HOC language of the NEURON simulator (either as standalone HOC models or as HOC templates) [33]. It implements all model-related methods (capabilities) that are needed to simulate these kinds of neural models in order to generate the prediction without any further coding required from the user.

For the smooth validation of the models developed using parameter optimization within the HBP there is a child class of the `ModelLoader` available in HippoUnit that is called `ModelLoader_BPO`. This class inherits most of the functions (especially the capability functions) from the `ModelLoader` class, but it implements additional functions that are able to automatically deal with the specific way in which information is represented and stored in these optimized models. The role of these functions is to gather all the information from the metadata and configuration files of the models that are needed to set the parameters required to load the models and run the simulations on them (such as path to the model files, name of the model template or the simulation temperature (the `celsius` variable of Neuron)). This enables the validation of these models without any manual intervention needed from the user. The section lists required by the tests of HippoUnit are also created automatically using the morphology files of these models (for details see the "Classify apical sections of pyramidal cells" subsection). For neural models developed using other software and methods, the user needs to implement the capabilities through which the tests of HippoUnit perform the simulations and recordings on the model.

The capabilities are the interface between the tests and the models. The `ModelLoader` class inherits from the capabilities and must implement the methods of the capability. The test can only be run on a model if the necessary capability methods are implemented in the `ModelLoader`. All communication between the test and the model happens through the capabilities.

The methods of the score classes perform the quantitative comparison between the *prediction* and the *observation*, and return the score object containing the final score and some related data, such as the paths to the saved figure and data (JSON) files and the prediction and observation data. Although SciUnit and NeuronUnit have a number of different score types implemented, HippoUnit has its own scores, which better fit its tests and the observations belonging to them. For simplicity, we refer to the discrepancy between the target experimental data (*observation*) and the models' behavior (*prediction*) with respect to a studied feature using the term feature error. In most cases, when the basic statistics (mean and standard deviation) of the experimental features (typically measured in several different cells of the same cell type) are available, feature errors are computed as the absolute difference between the feature value of the model and the experimental mean feature value, divided by the experimental standard deviation (Z-score) [34]. The final score of a given test achieved by a given model is given by the average (or, in some cases, the sum) of the feature error scores for all the features evaluated by the test.

Implementation of the tests of HippoUnit

The Somatic Features Test

The Somatic Features Test uses the Electrophys Feature Extraction Library (eFEL) [35] to extract and evaluate the values of both subthreshold and suprathreshold (spiking) features from voltage traces that represent the response of the model to somatic current injections of different positive (depolarizing) and negative (hyperpolarizing) current amplitudes. Spiking features describe action potential shape (like AP width, AP rise/fall rate, AP amplitude, etc.) and timing (frequency, inter-spike intervals, time to first/last spike, etc.), while some passive features (such as the voltage base or the steady state voltage), and subthreshold features for negative current stimuli (voltage deflection, sag amplitude, etc.) are also examined.

In this test step currents of varying amplitudes are injected into the soma of the model and the voltage response is recorded. The simulation protocol is set according to an input configuration JSON file, which contains all the current amplitudes, the delay and the duration of the stimuli, and the stimulation and recording positions. Simulations using different current amplitudes are run in parallel if this is supported by the computing environment.

As the voltage responses of neurons to somatic current injections can strongly depend on the experimental method, and especially on the type of electrode used, target values for these features were extracted from two different datasets. One dataset was obtained from sharp electrode recordings from adult rat CA1 neurons (sharp electrode data set) [3], and the other dataset is from patch clamp recordings in rat CA1 pyramidal cells (data provided by Judit Makara (patch clamp dataset)). For both of these datasets we had access to the recorded voltage traces from multiple neurons, which made it possible to perform our own feature extraction using eFEL. This ensures that the features are interpreted and calculated the same way for both the experimental data and the models' voltage response during the simulation. Furthermore, it allows a more thorough comparison against a large number of features extracted from experimental recordings yielded using the exact same protocol, which is unlikely to be found in any paper of the available literature. However, to see how representative these datasets are of the literature as a whole we first compared some of the features extracted from these datasets to data available on Neuroelectro.org [36] and on Hippocampome.org [37]. The features we compared were the following: resting potential, voltage threshold, after-hyperpolarization (AHP) amplitudes (fast, slow), action potential width and sag ratio. Although these databases have mean and standard deviation values for these features that are calculated from measurements using different methods, protocols and from different animals, we found that most of the feature values for our two experimental datasets fall into the ranges declared as typical for CA1 PCs in the online databases. The only conspicuous exception is the fast AHP amplitude of the patch clamp dataset used in this study, which is 1.7 ± 1.5 mV, while the databases cite values between 6.8 and 11.64 mV. This deviation could possibly stem from a difference in the way that the fast AHP is measured.

We also performed a more specific review of the relevant literature to compare the most important somatic features of the patch clamp dataset to results from available patch clamp recordings. In the literature the somatic AP voltage threshold of CA1 pyramidal cells is between -46 and -53 mV [38–41]. The same feature (*AP_begin_voltage*) extracted from out patch clamp dataset falls into this range (-51.13±0.97 mV (0.15 nA current step), -50.14±1.97 mV (0.2 nA current step); -49.36±2.02 mV (0.25 nA current step)). The AP amplitude falls into a relatively broad range between 71 and 112 mV according to the literature [38,41,42]. As most of these sources calculate the amplitude between the peak and the voltage base, we take the eFEL feature *AP_amplitude_from_voltagebase* into account here. The value of it in our patch clamp dataset is similar to the experimental observations (98.36±5.82 mV (0.15 nA current step), 96.83±5.66 mV (0.2 nA current step), 95.99±5.22 mV (0.25 nA current step)). The AP width measured at half amplitude ranges from 0.8 to 1.29 ms in the literature [38,40–42]. The value of this feature is near the upper end of this range in our patch clamp dataset (1.23±0.096 ms (0.15 nA step current); 1.25±0.11 ms (0.2 nA step current); 1.32±0.086 ms (0.25 nA step current)). Regarding the features extracted from response to hyperpolarizing currents the sag ratio can be compared. The values from literature are: 0.84±0.02 [42] and 0.83±0.01 [43], while the values extracted from our patch clamp dataset are quite similar (0.79±0.023 (-0.05 nA step current); 0.81±0.03 (-0.1 nA step current); 0.81±0.027 (-0.15 nA step current); 0.81±0.03 (-0.2 nA step current); 0.80±0.03 (-0.25 nA step current). We conclude that the patch clamp dataset is in good agreement with experimental observations available in the literature, and will be used as a representative example in this study.

The *observation* data are loaded from a JSON file of a given format which contains the names of the features to be evaluated, the current amplitude for which the given feature is evaluated and the corresponding experimental mean and standard deviation values. Setting the `specify_data_set` parameter it can be ensured that the test results against different experimental data sets are saved into different folders.

For certain features eFEL returns a vector as a result; in these cases, the feature value used by HippoUnit is the average of the elements of the vector. These are typically spiking features for which eFEL extracts a value corresponding to each spike fired. For features that use the ‘AP_begin_time’ or ‘AP_begin_voltage’ feature values for further calculations, we exclude the first element of the vector output before averaging because we discovered that these features are often incorrectly detected for the first action potential of a train.

The score class of this test returns as the final score the average of *Z-scores* for the evaluated eFEL features achieved by the model. Those features that could not be evaluated (e.g., spiking features from voltage responses without any spikes) are listed in a log file to inform the user, and the number of successfully evaluated features out of the number of features attempted to be evaluated is also reported.

The Depolarization Block Test

This test aims to determine whether the model enters depolarization block in response to a prolonged, high intensity somatic current stimulus. For CA1 pyramidal cells, the test relies on experimental data from Bianchi et al. [24]. According to these data, CA1 PCs respond to somatic current injections of increasing intensity with an increasing number of action potentials until a certain threshold current intensity is reached. For current intensities higher than the threshold, the cell does not fire over the whole period of the stimulus; instead, firing stops after some action potentials, and the membrane potential is sustained at some constant depolarized level for the rest of the stimulus. This phenomenon is termed depolarization block [24].

This test uses the same capability class as the Somatic Features Test for injecting current and recording the somatic membrane potential (see the description above). Using this capability, the model is stimulated with 1000 ms long square current pulses increasing in amplitude from 0 to 1.6 nA in 0.05 nA steps, analogous to the experimental protocol. The stimuli of different amplitudes are run in parallel. Somatic spikes are detected and counted using eFEL [35].

From the somatic voltage responses of the model, the following features are evaluated. I_{th} is the threshold current to reach depolarization block; experimentally, this is both the amplitude of the current injection at which the cell exhibits the maximum number of spikes, and the highest stimulus amplitude that does not elicit depolarization block. In the test two separate features are evaluated for the model and compared to the experimental I_{th}: the current intensity for which the model fires the maximum number of action potentials (*I_maxNumAP*), and the current intensity one step before the model enters depolarization block (*I_below_depol_block*). If these two feature values are not equal, a penalty is added to the score. The model is defined to exhibit depolarization block if *I_maxNumAP* is not the highest amplitude tested, and if there exists a current intensity higher than *I_maxNumAP*, for which the model does not fire action potentials during the last 100 ms of its voltage response.

In the experiment the V_{eq} feature is extracted from the voltage response of the pyramidal cells to the current injection one step above I_{th} *(or I_max_num_AP* in the test). Both in the experiment and in this test this is calculated as the mean voltage over the last 100 ms of the voltage trace. However, in the test, before calculating this value it is examined whether there are any action potentials during this period. The presence of spikes here means that the model did not enter depolarization block prior to this period. In these cases the test iterates further on the voltage traces corresponding to larger current steps to find if there is any where the model actually entered depolarization block; if an appropriate trace is found, the value of V_{eq} is extracted there. This trace is the response to the current intensity one step above *I_below_depol_block.*

If the model does not enter depolarization block, a penalty is applied, and the final score gets the value of 100. Otherwise, the final score achieved by the model on this test is the average of the error scores (Z-scores) for the features described above, plus an additional penalty if *I_maxNumAP* and *I_below_depol_block* differ. This penalty is 200 times the difference between the two current amplitude values (in pA – which in this case is 10 times the number of examined steps between them).

The Back-propagating AP Test

This test evaluates the strength of action potential back-propagation in the apical trunk at locations of different distances from the soma. The observation data for this test were yielded by the digitization of Figure 1B of [26], using the DigitizeIt software [44]. The values were then averaged over distances of 50, 150, 250, 350 ± 20 μm from the soma to get the mean and standard deviation of the features. The features tested here are the amplitudes of the first and last action potentials of a 15 Hz spike train, measured at the 4 different dendritic locations.

The test automatically finds current amplitudes for which the soma fires, on average, between 10-20 Hz and chooses the amplitude that leads to firing nearest to 15 Hz. For this task, the following algorithm was implemented. Increasing current step stimuli of 0.0 - 1.0 nA amplitude with a step size of 0.1 nA are applied to the model and the number of spikes is counted for each resulting voltage trace. If spontaneous spiking occurs (i.e., if there are spikes even when no current is injected) or if the spiking rate does not reach 10 Hz even for the highest amplitude, the test quits with an error message. Otherwise the amplitudes for which the soma fires between 10 and 20 Hz are appended to a list and (if the list is not empty) the one providing the spiking rate nearest to 15 Hz is chosen. If the list is empty because the spiking rate is smaller than 10 Hz for a step amplitude but higher than 20 Hz for the next step, a binary search method is used to find an appropriate amplitude in this range.

This test uses a trunk section list (or generates one if the `find_section_lists` variable of the `ModelLoader` is set to True – see the section ‘Classifying the apical sections of pyramidal cells’ below) to automatically find the dendritic locations for the measurements. The desired distances of the locations from the soma and the distance tolerance are read from the input configuration file, and must agree with the distances and the tolerance over which the experimental data were averaged. All the trunk dendritic segments whose distance from the soma falls into one of the distance ranges are selected. The locations and also their distances are then returned in separate dictionaries.

Then the soma is stimulated with a current injection of the previously chosen amplitude and the voltage response of the soma and the selected dendritic locations are recorded and returned.

The test implements its own function to extract the amplitudes of back-propagating action potentials, but the method is based on eFEL features. This is needed because eFEL’s spike detection is based on a given threshold value for spike initiation, which may not be reached by the back-propagating signal at more distant regions. First the maximum depolarization of the first and the last action potentials are calculated. This is the maximum value of the voltage trace in a time interval around the somatic action potential, based on the start time of the spike (using the AP_begin_time feature of eFEL) and the inter-spike interval to the next spike recorded at the soma. Then the amplitudes are calculated as the difference between this maximum value and the voltage at the begin time of the spike (on the soma) minus 1 ms (which is early enough not to include the rising phase of the spike, and late enough in the case of the last action potential not to include the afterhyperpolarization of the previous spike).

To calculate the feature error scores the amplitude values are first averaged over the distance ranges to be compared to the experimental data and get the feature Z-scores. The final score here is the average of the Z-scores achieved for the features of first and last action potential amplitudes at different dendritic distances. In the result it is also stated whether the model is more like a strongly or a weakly propagating cell in the experiment, where they found examples of both types [26].

The PSP Attenuation Test

The PSP Attenuation test evaluates how much the post-synaptic potential attenuates as it propagates from different dendritic locations to the soma in CA1 pyramidal cell models. The *observation* data for this test were yielded by the digitization of Figure 1E and Figure 2B of Magee and Cook, 2000 [28] using the DigitizeIt software [44]. The somatic and dendritic depolarization values were then averaged over distances of 100, 200, 300 ± 50 μm from the soma and the soma/dendrite attenuation was calculated to get the mean and standard deviation of the attenuation features at the three different input distances.

In this test the apical trunk receives excitatory post-synaptic current (EPSC)-shaped current stimuli at locations of different distances from the soma. The maximum depolarization caused by the input is extracted at the soma and divided by the maximum depolarization at the location of the stimulus to get the soma/dendrite attenuation values that are then averaged in distance ranges of 100, 200, 300 ± 50 μm and compared to the experimental data. The distances and tolerance are defined in the configuration file and must agree with how the *observation* data were generated.

The test uses a trunk section list, which needs to be specified in the NEURON HOC model (or the test generates one if the `find_section_lists` variable of the `ModelLoader` is set to True – see the section 'Classify apical sections of pyramidal cells' below) to find the dendritic locations to be stimulated. Randomly selected dendritic locations are used because the distance ranges that are evaluated cover almost the whole length of the trunk of a pyramidal cell. The probability of selecting a given dendritic segment is set to be proportional to its length. The number of dendritic segments examined can be chosen by the user by setting the `num_of_dend_locations` argument of the test. The random seed (also an argument of the test) must be kept constant to make the selection reproducible. If a given segment is selected multiple times (or it is closer than 50 µm or further than 350 µm), a new random number is generated. If the number of locations to be selected is more than the number of trunk segments available in the model, all the segments are selected.

The *Exp2Syn* synaptic model of NEURON with a previously calculated weight is used to stimulate the dendrite. The desired EPSC amplitude and time constants are given in the input configuration file according to the experimental protocol. To get the proper synaptic weight, first the stimulus is run with weight = 0. The last 10% of the trace is averaged to get the resting membrane potential (Vm). Then the synaptic weight required to induce EPSCs with the experimentally determined amplitude is calculated according to Equation 1:

(1) weight = - EPSC_amp / Vm

where EPSC_amp is read from the `config` dictionary, and the synaptic reversal potential is assumed to be 0 mV.

To get the somatic and dendritic maximum depolarization from the voltage traces, the baseline trace (weight = 0) is subtracted from the trace recorded in the presence of the input. To get the attenuation ratio the maximum value of the somatic depolarization is divided by the maximum value of the dendritic depolarization.

To calculate the feature error scores the soma/dendrite attenuation values are first averaged over the distance ranges to be compared to the experimental data to get the feature Z-scores. The final score is the average of the feature error scores calculated at the different dendritic locations.

The Oblique Integration Test

This test evaluates the signal integration properties of radial oblique dendrites, determined by providing an increasing number of synchronous (0.1 ms between inputs) or asynchronous (2 ms between inputs) clustered synaptic inputs. The experimental mean and standard error (SE) of the features examined are available in the paper of Losonczy and Magee [32] and are read from a JSON file into the *observation* dictionary of the test. The SE values are then converted to standard deviation values. The following features are tested: voltage threshold for dendritic spike initiation (defined as the expected somatic depolarization at which a step-like increase in peak dV/dt occurs); proximal threshold (defined the same way as above, but including only those results in the statistics where the proximal part of the examined dendrite was stimulated); distal threshold; degree of nonlinearity at threshold; suprathreshold degree of nonlinearity; peak derivative of somatic voltage at threshold; peak amplitude of somatic EPSP; time to peak of somatic EPSP; degree of nonlinearity in the case of asynchronous inputs.

The test automatically selects a list of oblique dendrites that meet the criteria of the experimental protocol, based on a section list containing the oblique dendritic sections (this can either be provided by the HOC model, or generated automatically if the `find_section_lists` variable of the `ModelLoader` is set to True – see the section ‘Classify apical sections of pyramidal cells’ below). For each selected oblique dendrite a proximal and a distal location is examined. The criteria for the selection of dendrites, which were also applied in the experiments, are the following. The selected oblique dendrites should be terminal dendrites (they have no child sections) and they should be at most 120 µm from the soma. This latter criterion can be changed by the user by changing the value of the `ModelLoader`'s *`max_dist_from_soma`* variable, and it can also increase automatically if needed. In particular, if no appropriate oblique is found up to the upper bound provided, the distance is increased iteratively by 15 µm, but not further than 190 µm.

Then an increasing number of synaptic inputs are activated at the selected dendritic locations separately, while recording the local and somatic voltage response. HippoUnit provides a default synapse model to be used in the `ObliqueIntegrationTest`. If the *`AMPA_name`*, and *`NMDA_name`* variables are not set by the user, the default synapse is used. In this case the AMPA component of the synapse is given by the built-in `Exp2Syn` synapse of NEURON, while the NMDA component is defined in an NMODL (.mod) file which is part of the HippoUnit package. This NMDA receptor model uses a Jahr-Stevens voltage dependence [45] and rise and decay time constants of 3.3 and 102.38 ms, respectively. The time constant values used here are temperature- (Q10-) corrected values from [41]. Q10 values for the rise and decay time constants were 2.2 [46] and 1.7 [47], respectively. The model's own AMPA and NMDA receptor models can also be used in this test if their NMODL files are available and compiled among the other mechanisms of the model. In this case the `AMPA_name`, and `NMDA_name` variables need to be provided by the user. The time constants of the built-in Exp2Syn AMPA component and the AMPA/NMDA ratio can be adjusted by the user by setting the *AMPA_tau1*, *AMPA_tau2* and `AMPA_NMDA_ratio` parameter of the `ModelLoader`. The default AMPA/NMDA ratio is 2.0 from [41], and the default `AMPA_tau1` and `AMPA_tau2` are 0.1 ms and 2.0 ms, respectively [28,29].

To test the Poirazi et al. 2003 model using its own receptor models, we also had to implement a modified version of the synapse functions of the `ModelLoader` that can deal with the different (pointer-based) implementation of synaptic activation in this model. For this purpose, a child class was implemented that inherits from the `ModelLoader` class. This modified version is not part of the official HippoUnit version, but is available here: https://github.com/KaliLab/HippoUnit_demo/blob/master/ModelLoader_Poirazi_2003_CA1.py.

The synaptic weights for each selected dendritic location are automatically adjusted by the test using a binary search algorithm so that the threshold for dendritic spike generation is 5 synchronous inputs – which was the average number of inputs that had to be activated by glutamate uncaging to evoke a dendritic spike in the experiments [32]. This search runs in parallel for all selected dendritic locations. The search interval of the binary search and the initial step size of the searching range can be adjusted by the user through the `c_minmax` and `c_step_start` variables of the `ModelLoader`. During the iterations of the algorithm the step size may decrease if needed; a lower threshold for the step size (`c_step_stop` variable of the `ModelLoader`) must be set to avoid infinite looping. Those dendritic locations where this first dendritic spike generates a somatic action potential, or where no dendritic spike can be evoked, are excluded from further analysis. To let the user know, this information is displayed on the output and also printed into the log file saved by the test. Most of the features above are extracted at the threshold input level (5 inputs).

The final score of this test is the average of the feature error scores achieved by the model for the different features; however, a T-test analysis is also available as a separate score type for this test.

Parallel computing

Most of the tests of HippoUnit require multiple simulations of the same model, either using stimuli of different intensities or at different locations in the cell. To run these simulations in parallel and save time, the Python `multiprocessing.Pool` module is used. The size of the pool can be set by the user. Moreover, all NEURON simulations are performed in multiprocessing pools to ensure that they run independently of each other, and to make it easy to erase the models after the process has finished. This is especially important in the case of HOC templates in order to avoid previously loaded templates running in the background and the occurrence of 'Template cannot be redefined' errors when the same model template is loaded again.

Classifying the apical sections of pyramidal cells

Some of the validation tests of HippoUnit require lists of sections belonging to the different dendritic types of the apical tree (main apical trunk, apical tuft dendrites, radial oblique dendrites). To classify the dendrites NeuroM [48] is used as a base package. NeuroM contains a script that, starting from the tuft (uppermost dendritic branches in Fig 1) endpoints, iterates down the tree to find a single common ancestor. This is considered as the apical point. The apical point is the upper end of the main apical dendrite (trunk), from where the tuft region arises. Every dendrite branching from the trunk below this point is considered an oblique dendrite.

However, there are many CA1 pyramidal cell morphologies where the trunk bifurcates close to the soma to form two or even more branches. In these cases the method described above finds this proximal bifurcation point as the apical point (see Fig 1A). To overcome this issue, we worked out and implemented a method to find multiple apical points by iterating the function provided by NeuroM. In particular, if the initial apical point is closer to the soma than a pre-defined threshold, the function is run again on subtrees of the apical tree where the root node of the subtree is the previously found apical point, to find apical points on those subtrees (see Fig 1B). When (possibly after multiple iterations) apical points that are far enough from the soma are found, NeuroM is used to iterate down from them on the parent sections, which will be the trunk sections (blue dots in Fig 1C). Iterating up, the tuft sections are found (green dots in Fig 1C), and the other descendants of the trunk sections are considered to be oblique dendrites (yellow dots in Fig 1C). Once all the sections are classified, their NeuroM coordinates are converted to NEURON section information for further use.

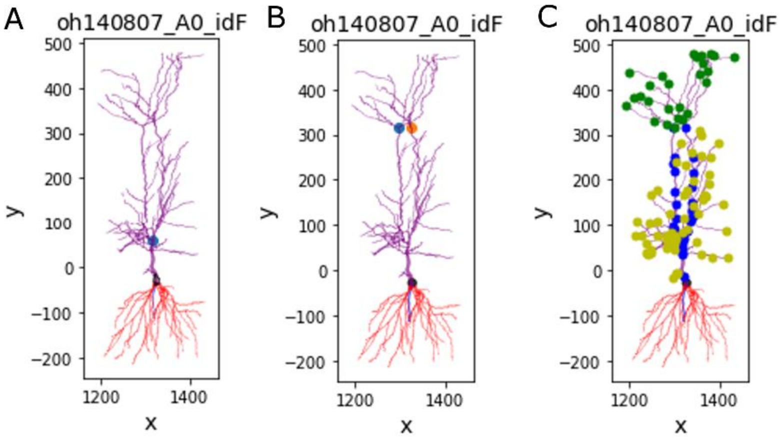

Fig 1: Classifying the apical dendrites of pyramidal cells. Morphological reconstruction made within the HBP at University College London (UCL). (A) The original method of NeuroM finds a single apical point which is actually a bifurcation of the trunk. (B) Further developing the method, multiple apical points can be found. (C)The apical dendritic sections are classified. Blue: trunk, yellow: oblique dendrites, green: tuft sections.

We note that this function can only be used for hoc models that load their morphologies from a separate morphology file (e.g., ASC, SWC) as NeuroM can only deal with morphologies provided in these standard formats. For models with NEURON morphologies implemented directly in the hoc language, the SectionLists required by a given test should be implemented within the model.

Models from literature

In this paper we demonstrate the utility of the HippoUnit validation test suite by applying its tests to validate and compare the behavior of several different detailed hippocampal CA1 pyramidal cell models available on ModelDB [17]. For this initial comparison we chose models published by several modeling groups worldwide that were originally developed for various purposes.

The Golding et al., 2001 model [26] (ModelDB accession number: 64167) was developed to show the dichotomy of the back-propagation efficacy and the amplitudes of the back-propagating action potentials at distal trunk regions in CA1 pyramidal cells and to make predictions on the possible causes of this behavior. It contains only the most important ion channels (Na, K_{DR}, K_A) needed to reproduce the generation and propagation of action potentials. Here we tested three different versions of the model: the ones corresponding to Figure 8A, Figure 8B and Figure 9B of the paper [26].

The Katz et al., 2009 model [49] (ModelDB accession number: 127351) is based on the Golding et al. 2001 model and was built to investigate the functional consequences of the distribution of strength and density of synapses on the apical dendrites that they observed experimentally, for the mode of dendritic integration.

The Migliore et al., 2011 model [50] (ModelDB accession number: 138205) was used to study schizophrenic behavior. It is based on earlier models of the same modeling group, which were used to investigate the initiation and propagation of action potentials in oblique dendrites, and have been validated against different electrophysiological data.

The Poirazi et al., 2003 model [6,51] (ModelDB accession number: 20212) was designed to clarify the issues about the integrative properties of thin apical dendrites that may arise from the different and sometimes conflicting interpretations of available experimental data. This is a quite complex model in the sense that it contains a large number of different types of ion channels, whose properties were adjusted to fit in vitro experimental data, and it also contains four types of synaptic receptors.

The Bianchi et al., 2012 model [24] (ModelDB accession number: 143719) was designed to investigate the mechanisms behind depolarization block observed experimentally in the somatic spiking behavior of CA1 pyramidal cells. It was developed by combining and modifying the Shah et al., 2008 [52] and the Poirazi et al. 2003 models [6,51]. The former of these was developed to show the significance of axonal M-type potassium channels.

The Gómez González et al., 2011 [53] model (ModelDB accession number: 144450) is based on the Poirazi et al. 2003 model and it was modified to replicate the experimental data of [32] on the nonlinear signal integration of radial oblique dendrites when the inputs arrive in a short time window. The model was adjusted to five different detailed morphologies.

Models from literature that are published on ModelDB typically implement their own simulations and plots to make it easier for users and readers to reproduce and visualize the results shown in the corresponding paper. Therefore, to be able to test the models described above using our test suite, we needed to create standalone versions of them. These standalone versions do not display any GUI, or contain any built-in simulations and run-time modifications, but otherwise their behavior should be identical to the published version of the models. We also added section lists of the radial oblique and the trunk dendritic sections to those models where this was not done yet, as some of the tests require these lists. To ensure that the standalone versions have the same properties as the original models, we checked their parameters after running their built-in simulations (in case including any run-time modifications), and made sure they match the parameters of the standalone version. We also asked the developers of the models to check the standalone versions and give us feedback; however, we received substantial feedback only from the developers of the Migliore et al. 2011 and the Bianchi et al., 2012 models, which were positive. The modified models used for running validation tests are available in this GitHub repository: https://github.com/KaliLab/HippoUnit_demo.

Results

The HippoUnit validation suite

HippoUnit (https://github.com/KaliLab/hippounit) is an open source test suite for the automatic and quantitative evaluation and validation of the behavior of neural single cell models. The tests of HippoUnit automatically perform simulations that mimic common electrophysiological protocols on neuronal models to compare their behavior with quantitative experimental data using various feature-based error functions. Current validation tests cover somatic (subthreshold and spiking) behavior as well as signal propagation and integration in the dendrites. The tests were developed using data and models for rat hippocampal CA1 pyramidal cells. However, most of the tests are directly applicable to or can be adapted for other cell types if the necessary experimental data are available; examples of this will be presented in later sections.

HippoUnit is implemented in the Python programming language, and is based on the SciUnit [18] framework for testing scientific models. The current version of HippoUnit is capable of handling single cell models implemented in the NEURON simulator. As a result, by adapting and using the example Jupyter notebooks described in S1 Appendix, the tests of HippoUnit can be run on neural models that are built in the NEURON simulator software, without any further coding required from the user. In principle, neural models developed using other software tools can also be tested by HippoUnit; however, this requires the re-implementation by the user of the interface functions that allow HippoUnit to run the necessary simulations and record their output (see the Methods section for more details).

In the current tests of HippoUnit, once all the necessary simulations have been performed and the responses of the model have been recorded, electrophysiological features are extracted from the voltage traces, and the discrepancy between the model's behavior and the experiment is computed by comparing the feature values with those extracted from the experimental data (see Methods). For simplicity, we refer to the result of this comparison as the feature error; however, we note that there are many possible sources of such discrepancy including, among others, experimental artefacts and noise, shortcomings of the models, and differences between the conditions assumed by the models and those in the actual experiments (see the Discussion for more details). The final score of a given test achieved by a given model is given by the average (or, in some cases, the sum) of the feature error scores for all the features evaluated by the test.

Besides the final score, which is the basic output of all the tests, the tests of HippoUnit typically provide a number of other useful outputs (see Methods), including figures that visualize the model's behavior through traces and plot the feature and error values compared to the experimental data. It is always strongly recommended to look at the traces and other figures to get a fuller picture of the model's response to the stimuli, which helps with the correct interpretation of validation results. Such closer inspection also makes it possible to detect possible test failures, when the extraction of certain features does not work correctly for a given model.

HippoUnit can also take advantage of the parallel execution capabilities of modern computers. When tests require multiple simulations of the same model using different settings (e.g., different stimulation intensities or different stimulus locations in the cell), these simulations are run in parallel, which can make the validation process substantially faster, depending on the available computing resources.

One convenient way of running a test on a model is to use an interactive computational notebook, such as the Jupyter Notebook [54], which enables the combination of program codes to be run (we used Python code to access the functionality of HippoUnit), the resulting outputs (e.g. figures, tables, text) and commentary or explanatory text in a single document. Therefore, we demonstrate the usage of HippoUnit through this method (See S1 Appendix and https://github.com/KaliLab/HippoUnit_demo).

Comparison of the behavior of rat hippocampal CA1 pyramidal cell models selected from the literature

We selected six different publications containing models of hippocampal CA1 pyramidal cells whose implementations for the NEURON simulator were available in the ModelDB database. Our aim was to compare the behavior of every model to the experimental target data using the tests of HippoUnit, which also allowed us to compare the models to each other, and to test their generalization performance in paradigms that they were not originally designed to capture. These models differ in their complexity regarding the number and types of ion channels that they contain, and they were built for different purposes (see the Methods section for more details on the models). A common property of these models is that their parameters were set using manual procedures with the aim of reproducing the behavior of real CA1 PCs in one or a few specific paradigms. As some of them were built by modifying and further developing previous models, these share the same morphology (see Fig. 2). On the other hand, the model of Gómez González et al. 2011 was adjusted to 5 different morphologies, which were all tested. In the case of the Golding et al. 2001 model, we tested three different versions (shown in Figures 8A, 8B and 9A of the corresponding paper [26]) that differ in the distribution of the sodium and the A-type potassium channels, and therefore the back-propagation efficacy of the action potentials. The morphologies and characteristic voltage responses of all the models used in this comparison are displayed in Fig 2.

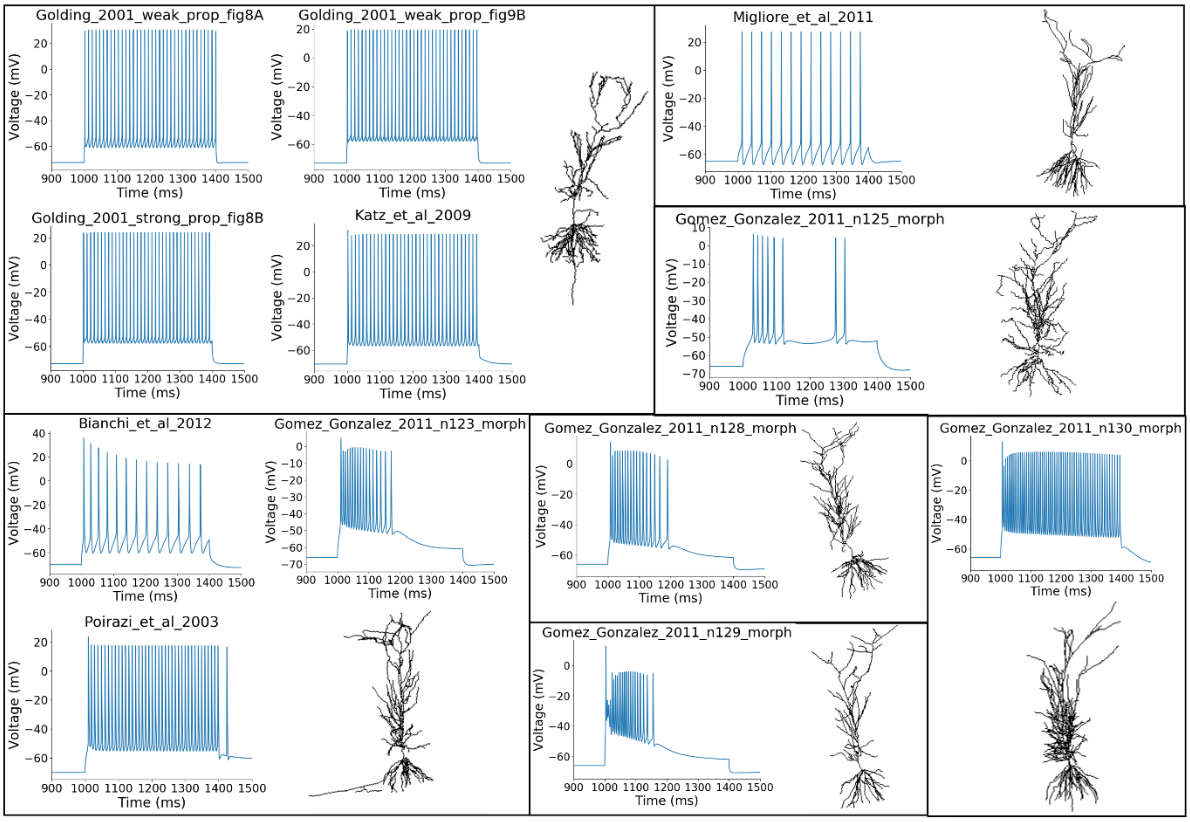

Fig 2: The morphologies of the different models tested and their voltage responses to a 400 ms step current injection of 0.6 nA amplitude.

Running the tests of HippoUnit on these models we took into account the original settings of the simulations of the models, and set the `v_init` (the initial voltage when the simulation starts), and the `celsius` (the temperature at which the simulation is done) variables accordingly. For the Bianchi et al 2012 model we used variable time step integration during all the simulations, as it was done in the original modeling study. For the other models a fixed time step were used (dt=0.025 ms).

Somatic Features Test

Using the Somatic Features Test of HippoUnit, we compared the behavior of the models to both patch clamp recordings (patch clamp dataset) and sharp electrode recordings (sharp electrode dataset). To see how representative these datasets are of the literature as a whole we first compared some of the features extracted from these datasets to data available on Neuroelectro.org [36] and on Hippocampome.org [37]. We also performed a more specific review of the relevant literature to compare the most important somatic features of our patch clamp dataset to results from published patch clamp recordings [38–43] (see Methods). We conclude that the patch clamp dataset is in good agreement with experimental observations available in the literature, and will be used as a representative example in this study.

The two datasets used in this study (sharp electrode dataset, patch clamp dataset) differ not only in the recording technique, but also in the simulation protocol. In the sharp electrode recordings, the cells received 400 ms-long depolarizing and hyperpolarizing current injections, using amplitudes of 0.2, 0.4, 0.6, 0.8 and 1.0 nA in both directions. In the patch clamp recordings, both the depolarizing and the hyperpolarizing current injections were 300 ms long and 0.05, 0.1, 0.15, 0.2, 0.25 nA in amplitude.

As each of the tested models apparently used experimental data obtained from patch clamp recordings as a reference, here we show the detailed results of the test on the models when their output was compared to the features extracted from the patch clamp data (we will return to the comparison between the two datasets near the end of this section). During these recordings the cells were stimulated with relatively low amplitude current injections. Some of the examined models (Migliore et al. 2011, Gómez González et al. 2011 n125 morphology) did not fire even for the highest amplitude tested. Some other models started to fire for higher current intensities than it was observed experimentally. In these cases the features that describe action potential shape or timing properties cannot be evaluated for the given model (for the current amplitudes affected). Therefore, besides the final score achieved by the models on this test (the average Z-score for the successfully evaluated features – see Methods for details), we also consider the proportion of the successfully evaluated features as an important measure of how closely the model matches this specific experimental dataset (Fig 4B).

Fig 3 shows how the extracted feature values of the somatic response traces of the different models fit the experimental values. It is clear that the behavior of the different models is very diverse. Each model captures some of the experimental features but shows a larger discrepancy for others.

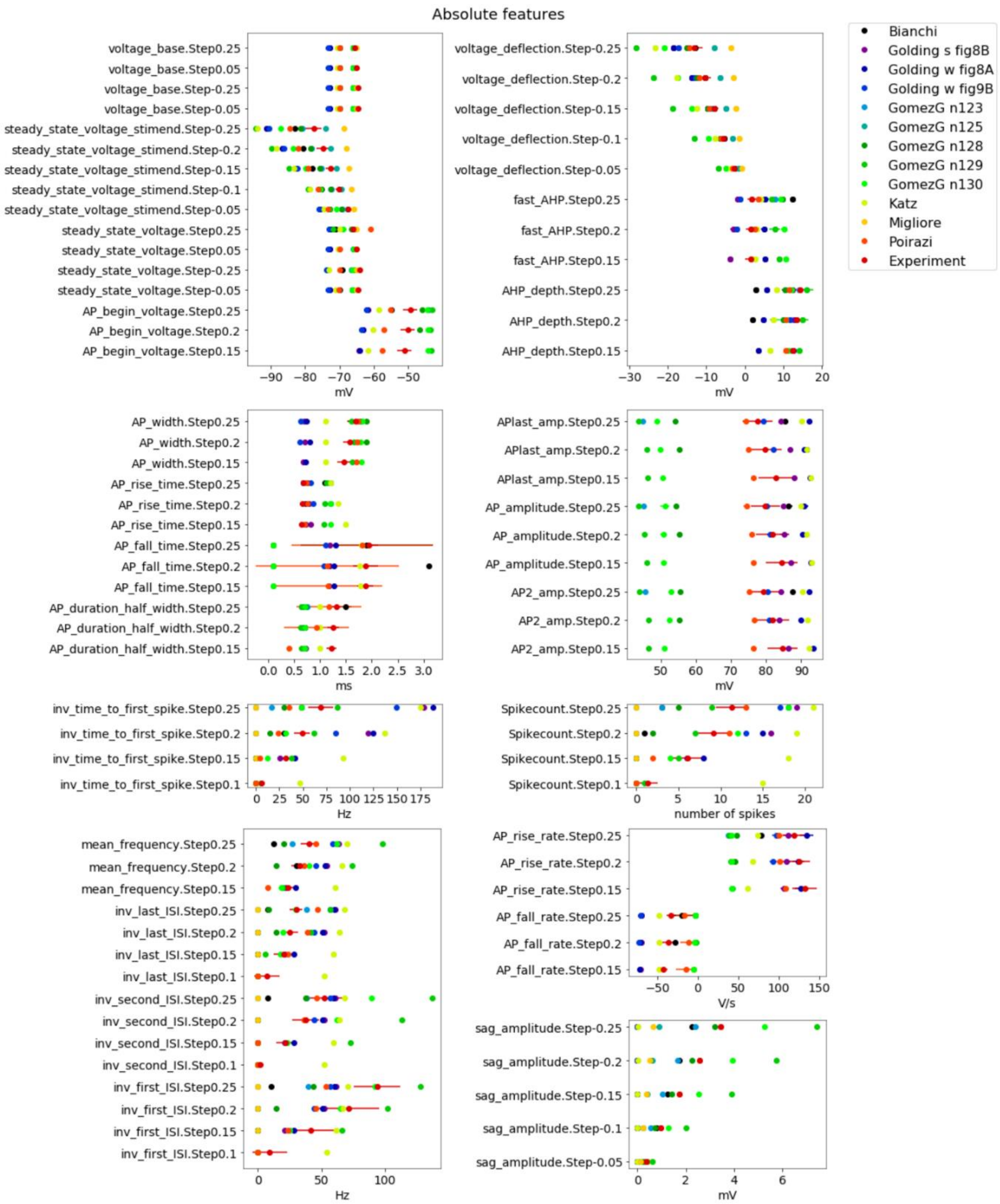

Fig 3: Feature values from the Somatic Features Test of HippoUnit applied to several published models. Absolute feature values extracted from the voltage responses of the models to somatic current injections of varying amplitude, compared to experimental values (darkest red) that were extracted from the patch clamp dataset . (Not all the evaluated features are shown here.)

The resting membrane potential (*voltage_base*) for all of the models was apparently adjusted to a more hyperpolarized value than in the experimental recordings we used for our comparison, and most of the models also return to a lower voltage value after the step stimuli (*steady_state_voltage*). An exception is the Poirazi et al. 2003 model, where the decay time constant after the stimulus is unusually high (data not shown in Fig 3, but the slow decay can be seen in the example trace in Fig 2.). The voltage threshold for action potential generation (*AP_begin_voltage*) is lower than the experimental value for most of the models (that were able to generate action potentials in response to the examined current intensities), but it is higher than the experimental value for most versions of the Gómez González et al. 2011 model. For negative current steps most of the models gets more hyperpolarized (*voltage_deflection*) (the most extreme is the Gómez González et al. 2011 model with the n129 morphology), while the Gómez González et al. 2011 model with the n125 morphology and the Migliore et al. 2011 model get less hyperpolarized than it was observed experimentally. The sag amplitudes are also quite high for the Gómez González et al. 2011 n129, and n130 models, while the Katz et al. 2009, and all versions of the Golding et al. 2001 models basically have no hyperpolarizing sag.

It is quite conspicuous how much the amplitude of the action potentials (*APlast_amp, AP_amplitude, AP2_amp*) differs in the Gómez González et al. 2011 models from the experimental values and from the other models as well. The Katz et al. 2009 and one of the versions (Fig 8A) of the Golding et al. 2001 model have slightly too high action potential amplitudes, and these models have relatively small action potential width (*AP_width*). On the other hand, the rising phase (*AP_rise_time, AP_rise_rate*) of the Katz et al. 2009 model appears to be too slow.

Looking at the inverse interspike interval (*ISI*) values, it can be seen that the experimental spike trains show adaptation in the ISIs, meaning that the first ISI is smaller (the inverse ISI is higher) than the last ISI for the same current injection amplitude. This behavior can be observed in the case of the Katz et al. 2009 model, three versions (n128, n129, n130 morphology) of the Gómez González et al. 2011 model, but cannot really be seen in the Bianchi et al. 2011, the Poirazi et al. 2003 and the three versions of the Golding et al. 2001 models. At first look it may seem contradictory that in the case of the Gómez González et al. 2011 model version n129 morphology the spike counts are quite low, while the mean frequency and the inverse ISI values are high. This is because the soma of this model does not fire over the whole period of the stimulation, but starts firing at higher frequencies, then stops firing for rest of the currently used (relative short) stimulus (see Fig 2), although it would start firing again for longer current injections (data not shown). The Katz et al. 2009 model fires quite a high number of action potentials (*Spikecount*) compared to the experimental data, at a high frequency.

In the experimental recordings there is a delay before the first action potential is generated, which becomes shorter with increasing current intensity (indicated by the *inv_time_to_first_spike* feature that becomes larger with increasing input intensity). In most of the models this behavior can be observed, albeit to different degrees. The Katz et al. 2009 model has the shortest delays (highest *inv_time_to_first_spike* values), but the effect is still visible.

To quantify the difference between the experimental dataset and the simulated output of the models, these were compared using the feature-based error function (Z-Score) described above to calculate the feature errors. Fig 4A shows the mean error scores of the model features whose absolute values are illustrated in Fig 3 (averaged over the different current step amplitudes examined). From this figure it is even more clearly visible that each model fits some experimental features well but does not capture others. For example, it is quite noticeable in Fig 4A that most of the versions of the Gómez González et al. 2011 model (greenish dots) perform well for features describing action potential timing (upper part of the figure, e.g., *ISIs*, *mean_frequency*, *spikecount*), but get higher error scores for features of action potential shape (lower part of the figure, e.g., *AP_rise_rate*, *AP_rise_time*, *AP_fall_rate*, *AP_fall_time*, *AP* *amplitudes*). Conversely, the Katz et al. 2009 model achieved better scores for AP shape features than for features describing AP timing. It is also worth noting that none of the error scores for the model of Migliore et al. 2011 was higher than 4; however, looking at Fig 4B it can be seen that less than half of the experimental features were successfully evaluated in this model, which is because it does not fire action potentials for the current injection amplitudes examined here.

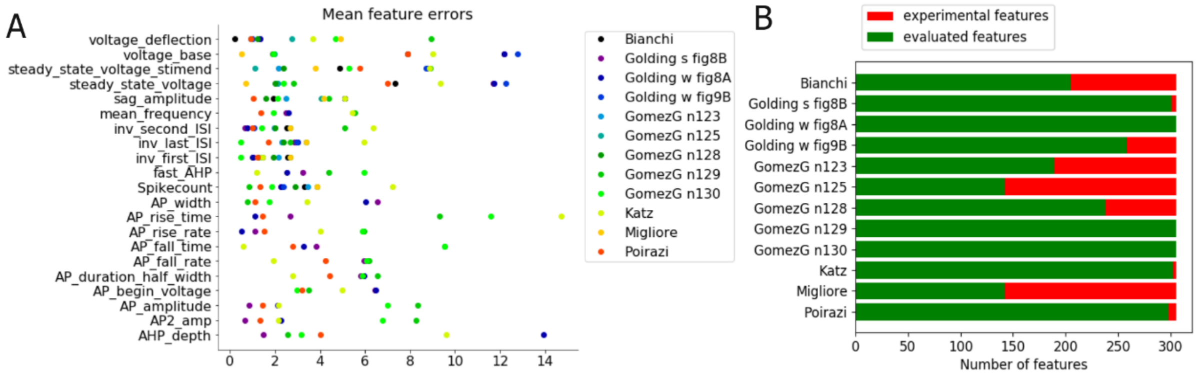

Fig 4: Evaluation of results from the Somatic Features Test of HippoUnit applied to published models. (A) Mean feature errors (in units of the experimental SD) of the different models. Feature error values are averaged over the different input step amplitudes. (B) The bars represent the number of features that were attempted to be evaluated for the models (i.e., the number of features extracted from the experimental patch clamp dataset). The number of successfully evaluated features for the various models is shown in green, and the number of features that could not be evaluated for a particular model is shown in red.

Besides enabling the comparison of different models regarding how well they match a particular dataset, the tests of HippoUnit also allow one to determine the match between a particular model and several datasets of the same type. As experimental results can be heavily influenced by recording conditions and protocols, and also depend on factors such as the strain, age, and sex of the animal, it is important to find out whether the same model can simultaneously capture the outcome of different experiments, and if not, how closely it is able to match the different datasets. As a practically relevant example, we looked at how well the various published models that we were testing captured a different experimental dataset that also contained current clamp recordings from rat CA1 PCs, but which was obtained using sharp electrodes rather than the whole-cell patch clamp technique [3]. We therefore evaluated all the models with the Somatic Features Test of HippoUnit using both datasets, and then compared the results.

When we simply compared the raw outputs of the test for each model evaluated using the two different data sets (Fig 5A) we identified two factors that substantially bias the results. First, we found that the standard deviation values for the features extracted from the two datasets are very different in magnitude; more specifically, the patch clamp recording data set had much lower standard deviation values for most of the features. This results in relatively higher feature error scores achieved by the models, as the difference of the model output from the experimental features is given in the unit of the experimental standard deviation. The other source of bias is that not all the features could be extracted from both of the data sets, and, as mentioned before, not the same current step protocol (current intensity, duration) was used during the different experiments. Consequently, the models are not compared to exactly the same set of features in the two cases. Mainly as a result of these two confounding factors, comparison of the raw scores of the models for the two data sets (Fig 5A) appears to indicate that most models fit the dataset obtained from sharp electrode recordings better, even though these models were typically built mostly based on patch clamp data.

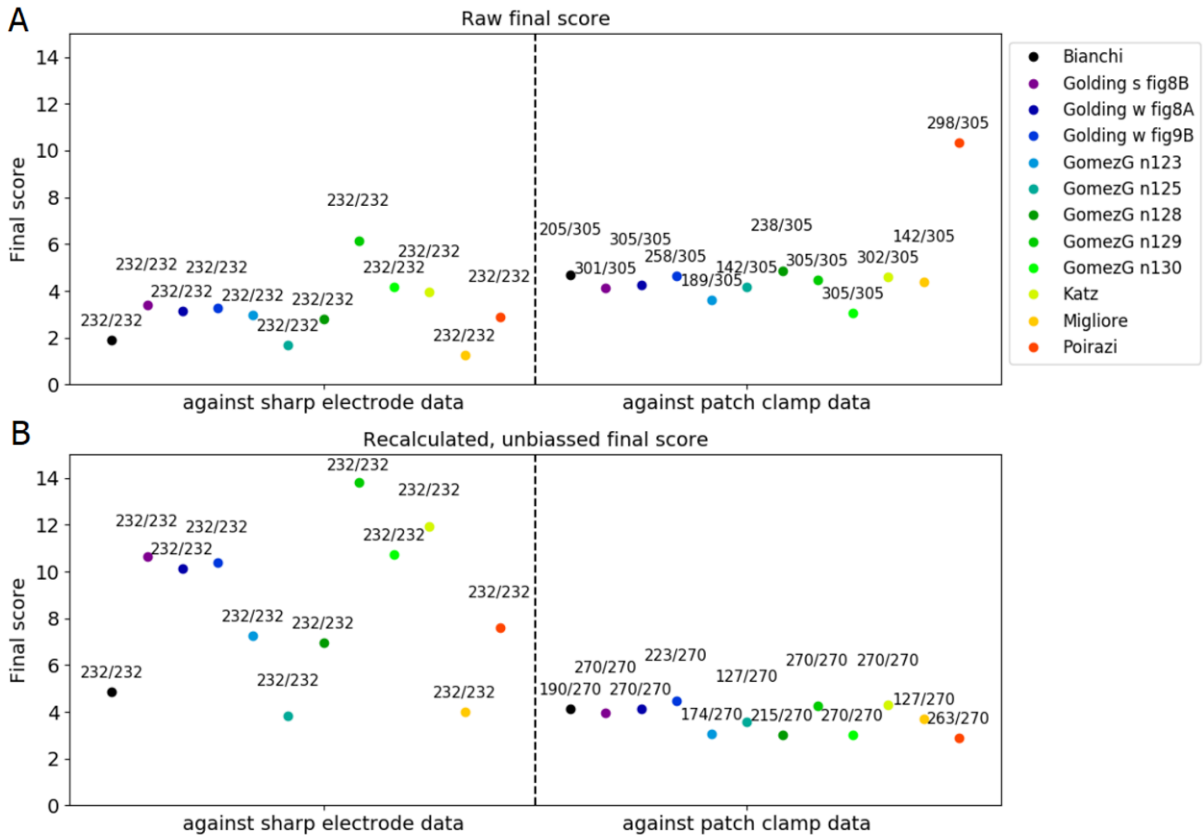

Fig 5: Comparison of the final scores achieved by the different models on the Somatic Features Test against validation data from two different datasets (sharp electrode data, patch clamp data). In the upper panel (A) the raw output of the tests is shown, while in the lower panel (B) the feature errors and the final scores have been recalculated using standardized standard deviation values. Numbers above each data point show the proportion of the successfully evaluated features compared to the number of features attempted to be evaluated (successfully extracted from the data set). Note that while in the recalculated final scores (B) only those eFEL features were taken into account that could be extracted from both datasets, they are extracted for different current step amplitudes, which accounts for the difference in the number of observation features for the two datasets.

To overcome these issues and make unbiased comparisons of the models to the two datasets, the feature error scores and the final scores were recalculated in the following way (Fig 5B). The new feature error scores for the two different data sets were calculated as the difference of the model’s feature value from the mean feature value of each dataset (as before), but divided by a common standard deviation value. This standardized SD value for each eFEL feature was the mean of the standard deviation values over the current steps in the patch clamp dataset (the results were qualitatively similar if we used the SD values from the sharp electrode dataset everywhere instead). Averaging the standard deviation values of the eFEL features over the current steps was required because the current step amplitudes were not the same in the two data sets, and we therefore needed to define SD values that were independent of the amplitude. To get rid of the second bias, only those eFEL features were used in the final score recalculation that are present in both observations (sharp electrode and patch clamp datasets) for at least one current step amplitude. (This change had the side effect of significantly decreasing the final score for the Poirazi et al. 2003 model because the feature *decay_time_constant_after_stim* was excluded here, as it could not be extracted from the sharp electrode data.) Now that the final scores are recalculated to get rid of most of the biasing factors, it becomes clear that the somatic behavior of every model fits the patch clamp data better (Fig 5B).

It is worth noting that one biasing factor still remains in the last comparison: as it has already been mentioned, not all the observation features can be evaluated for each of the models, especially when they are compared to the patch clamp data set, which uses smaller currents. To allow the assessment of the potential effect of this issue, the proportion of the successfully evaluated features relative to the number of features attempted to be evaluated (successfully extracted from the data set) for each model is also shown in Fig 5 next to each data point.

Depolarization Block Test

In the Depolarization Block Test three features are evaluated. Two of them examine the threshold current intensity to reach depolarization block. The *I_maxNumAP* feature is the current intensity at which the model fires the maximum number of action potentials, and the

I_below_depol_block feature is the current intensity one step before the model enters depolarization block. Both are compared to the experimental I_{th} feature because, in the experiment [24], the number of spikes increased monotonically with increasing current intensity up to the current amplitude where the cell entered depolarization block during the stimulus, which led to a drop in the number of action potentials. By contrast, we experienced that some models started to fire fewer spikes for higher current intensities while still firing over the whole period of the current step stimulus, i.e., without entering depolarization block. Therefore, we introduced the two separate features for the threshold current. If these two feature values are not equal, a penalty is added to the score. The third evaluated feature is V_{eq}, the equilibrium potential during the depolarization block, which is calculated as the average of the membrane potential over the last 100 ms of a current pulse with amplitude 50 pA above *I_maxNumAP* (or 50 pA above *I_below_depol_block* if its value is not equal to *I_maxNumAP*). Each model has a value for the "I_maxNumAP" feature, while those models that do not enter depolarization block are not supposed to have a value for the *I_below_depol_block* feature and the *Veq* feature.

The results from applying the Depolarization Block Test to the models from ModelDB are shown in Fig 6. According to the test, four of the models entered depolarization block. However, by looking at the actual voltage traces provided by the test, it becomes apparent that only the Bianchi et al. 2011 model behaves correctly (which was developed to show this behavior). The other three models actually managed to "cheat" the test.

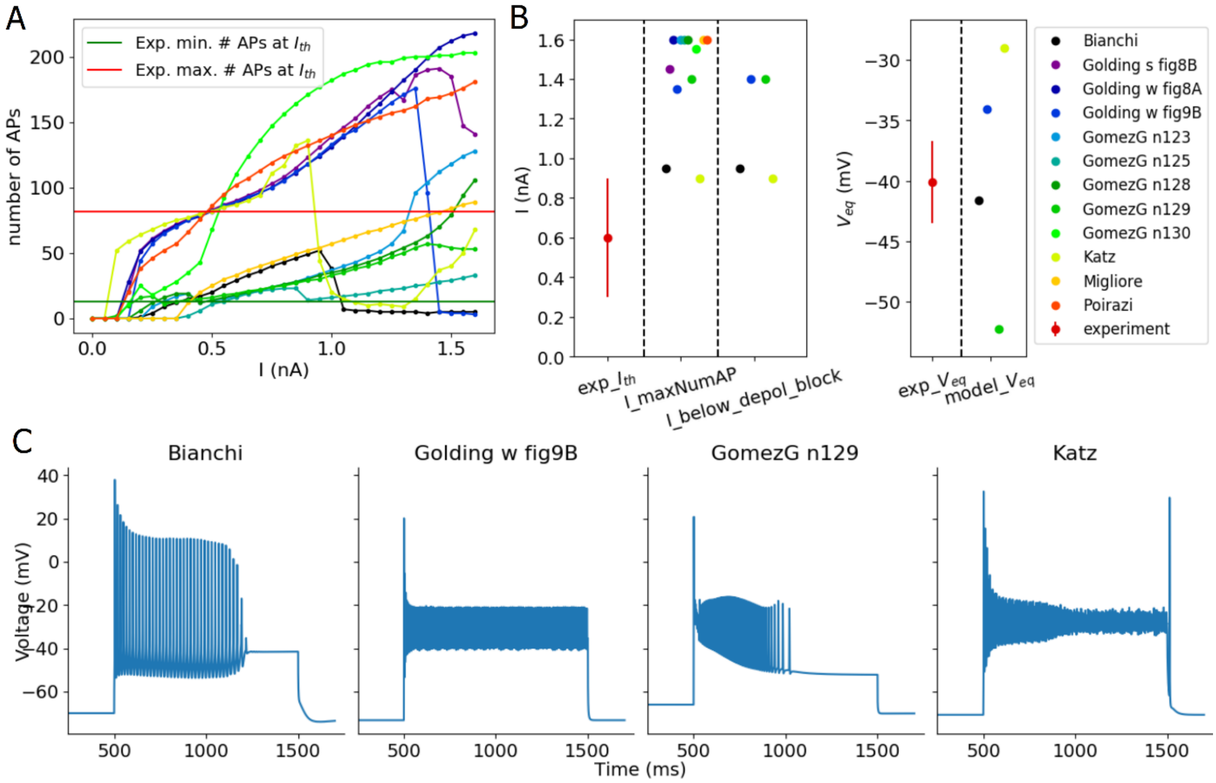

Fig 6: Results from the Depolarization Block Test of HippoUnit applied to published models. (A) Number of APs fired by the models in response to current injections of increasing intensity. (B) Depolarization block feature values extracted from the voltage responses of the models. (C) Voltage traces of different models that were recognized by the test as depolarization block.

In the case of the Katz et al. 2009 and the Golding et al. 2001 Fig 9B models, the APs get smaller and smaller with increasing stimulus amplitude until they get so small that they do not reach the threshold for action potential detection; therefore, these APs are not counted by the test and V_{eq} is also calculated. The Gómez González et al. 2011 model adjusted to the n129 morphology does not fire during the whole period of the current stimulation for a wide range of current amplitudes (see Fig 2). Increasing the intensity of the current injection it fires an increasing number of spikes, but always stops after a while before the end of the stimulus. On the other hand, there is a certain current intensity after which the model starts to fire fewer action potentials, and which is thus detected as *I_maxNumAP* by the test. Because no action potentials can be detected during the last 100 ms of the somatic response one step above the detected "threshold" current intensity, the model is declared to have entered depolarization block, and a V_{eq} value is also extracted. These cases underline the importance of critically evaluating the full output of the tests rather than blindly accepting the final scores provided.

Back-propagating Action Potential Test

This test first finds all the dendritic segments that belong to the main apical dendrite of the model and which are 50, 150, 250, 350 ± 20 μm from the soma, respectively. Then a train of action potentials of frequency around 15 Hz is triggered in the soma by injecting a step current of appropriate amplitude (as determined by the test), and the amplitudes of the first and last action potentials in the train are measured at the selected locations. In the Bianchi et al. 2012 and the Poirazi et al. 2003 models (which share the same morphology, see Fig 2) no suitable trunk locations could be found in the most proximal (50 ± 20 μm) and most distal (350 ± 20 μm) regions. This is because this morphology has quite long dendritic sections that are divided into a small number of segments. In particular, the first trunk section (apical_dendrite[0]) originates from the soma, is 102.66 μm long, and has only two segments. The center of one of them is 25.67 μm far from the soma, while the other is already 77 μm away from the soma. None of these segments belongs to the 50 ± 20 μm range, and therefore they are not selected by the test. The n123 morphology of the Gómez González et al. 2011 model has the same shape (Fig 2), but in this case the segments are different, and therefore it does not share the same problem.

At the remaining, successfully evaluated distance ranges in the apical trunk of the Bianchi et al. 2012 model, action potentials propagate very actively, barely attenuating. For the

AP1_amp and *APlast_amp* features at these distances, this model has the highest error score (Fig 7), while the Poirazi et al. 2003 model performs quite well.

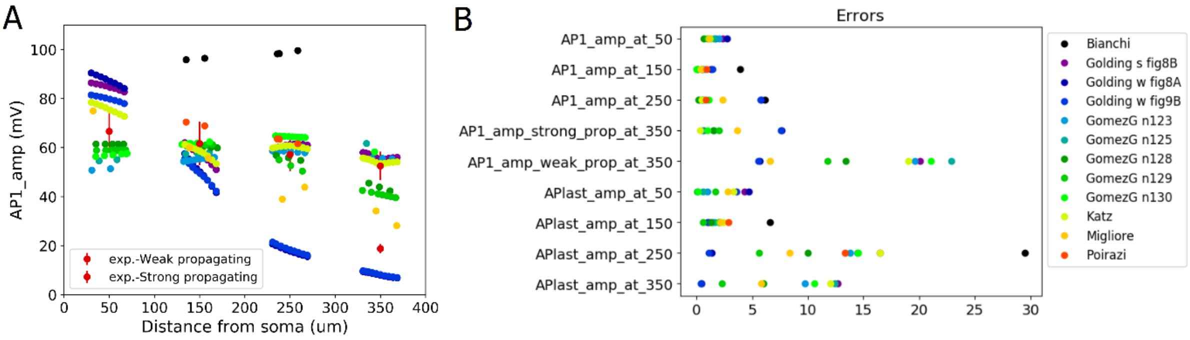

Fig 7: Results from the Back-propagating Action Potential Test of HippoUnit applied to published models. (A) The amplitudes of the first back-propagating action potentials (in a train of spikes with frequency around 15 Hz) as a function of recording location distance from the soma. (B) Feature error scores achieved by the different models on the Back-propagating AP Test. The amplitudes of the first and last back-propagating action potentials were averaged over the distance ranges of 50, 150, 250, 350 ±μm and compared to the experimental features (see Methods for more details).

The Golding et al. 2001 model was designed to investigate how the distribution of ion channels can affect the back-propagation efficacy in the trunk. The two versions of the Golding et al. 2001 model ("fig8A" and "fig9B" versions) which are supposed to be weakly propagating according to the corresponding paper [26], are also weakly propagating according to the test. However, the difference between their strongly and weakly propagating feature error scores is not too large (Fig 7), which is probably caused by the much smaller standard deviation value of the experimental data for the weakly propagating case. Although the amplitudes of the first action potentials of these two models fit the experimental data relatively well, they start to decline slightly closer to the soma than it was observed experimentally, as the amplitudes are already very small at 250 ± 20 μm (Fig 7). (In Fig 7 the data corresponding to these two versions of the model are almost completely overlapping for more distal regions.) The amplitudes for the last action potential fit the data well, except in the most proximal regions (data not shown). For all versions of the Golding et al. 2001 model, AP amplitudes are too high at the most proximal distance range. As for the strongly propagating version of the Golding et al. 2001 model (“fig8B” version), the amplitude of the first action potential is too high at the proximal locations, but further it fits the data well. The amplitude of the last action potential remains too high even at more distal locations. It is worth noting that, in the corresponding paper [26], they only examined a single action potential triggered by a 5 ms long input in their simulations, and did not examine or compare to their data the properties of the last action potential in a longer spike train. Finally, we note that in all versions of the Golding et al. 2001 model a spike train with frequency around 23 Hz was evoked and examined as it turned out to be difficult to set the frequency closer to 15 Hz.

The different versions of the Gómez González et al. 2011 model behave qualitatively similarly in this test, although there were smaller quantitative differences. In almost all versions the amplitudes of the first action potential in the dendrites are slightly too low at the most proximal locations but fit the experimental data better at further locations. The exceptions are the versions with the n128 and n129 morphologies, which have lower first action potential amplitudes at the furthest locations, but not low enough to be considered as weak propagating. The amplitudes for the last action potential are too high at the distal regions but fit better at the proximal ones. The only exception is the one with morphology n129, where the last action potential attenuates more at further locations and fits the data better.

In the case of the Katz et al. 2009 model, a spike train with frequency around 40 Hz was examined, as the firing frequency increases so suddenly with increasing current intensity in this model that no frequency closer to 15 Hz could be adjusted. In this model the last action potential propagates too strongly, while the dendritic amplitudes for the first action potential are close to the experimental values.

In the Migliore et al. 2011 model the amplitudes for the last action potential are too high, while the amplitude of the first back-propagating action potential is too low at locations in the 250 ± 20 μm and 350 ± 20 μm distance ranges.

Finally, all the models that we examined were found to be strongly propagating by the test, with the exception of those versions of the Golding et al. 2001 model that were explicitly developed to be weakly propagating.

PSP Attenuation Test

In this test the extent of the attenuation of the amplitude of an excitatory post-synaptic potential (EPSP) is examined as it propagates towards the soma from different input locations in the apical trunk. The Katz et al. 2009, the Bianchi et al. 2012, and all versions of the Golding et al. 2001 models perform quite well in this test. The various versions of the Golding et al. 2001 model are almost identical in this respect, which is not surprising as they differ only in the distribution of the sodium and A-type potassium channels. This shows that, as we would expect, these properties do not have much effect on the propagation of relatively low-amplitude signals such as unitary PSPs. Interestingly, the different versions of the Gómez González et al. 2011 model, with different morphologies, behave quite differently, which shows that this behavior can depend very much on the morphology of the dendritic tree.

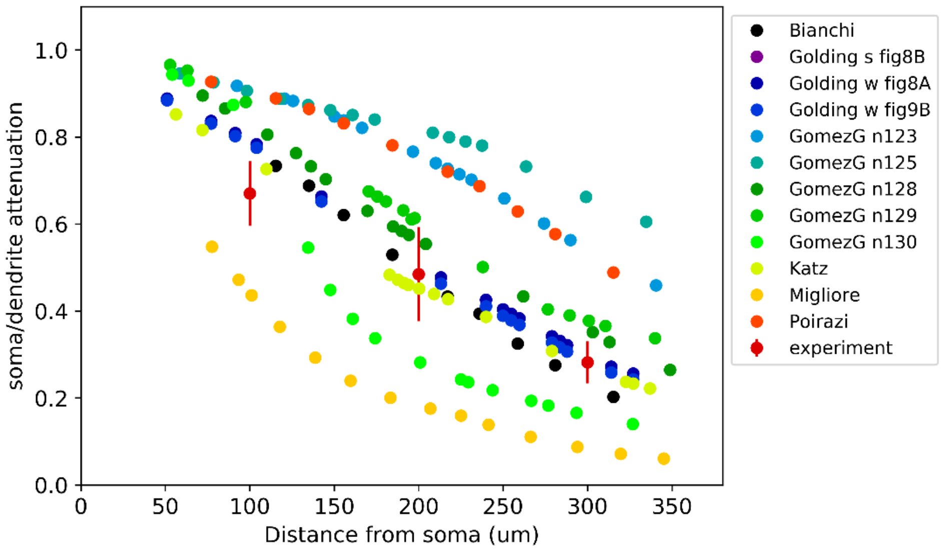

Fig 8: Results from the PSP Attenuation Test of HippoUnit applied to published models. Soma/dendrite EPSP attenuation as a function of the input distance from the soma in the different models.

Oblique Integration Test

This test probes the integration properties of the radial oblique dendrites of CA1 pyramidal cell models. The test is based on the experimental results described in [32]. In this study, the somatic voltage response was recorded while synaptic inputs in single oblique dendrites were activated in different spatio-temporal combinations using glutamate uncaging. The main finding was that a sufficiently high number of synchronously activated and spatially clustered inputs produced a supralinear response consisting of a fast (Na) and a slow (NMDA) component, while asynchronously activated inputs summed linearly or sublinearly.

This test selects all the radial oblique dendrites of the model that meet the experimental criteria: they are terminal dendrites (they have no child sections) and are at most 120 µm from the soma. Then the selected dendrites are stimulated in a proximal and in a distal region (separately) using an increasing number of clustered, synchronous or asynchronous synaptic inputs to get the voltage responses of the model, and extract the features of dendritic integration.

The synaptic inputs are not unitary inputs, i.e., their strength is not equivalent to the strength of one synapse in the real cell; instead, the strength is adjusted in a way that 5 synchronous inputs are needed to trigger a dendritic action potential. The intensity of the laser used for glutamate uncaging was set in a similar way in the experiments [32]. Most of the features were extracted at this just-suprathreshold level of input. We noticed that in some cases the strength of the synapse is not set correctly by the test; for example, it may happen that an actual dendritic spike does not reach the spike detection threshold in amplitude, or sometimes the EPSP may reach the threshold for spike detection without actual spike generation. The user has the ability to set the threshold used by eFEL for spike detection, but sometimes a single threshold may not work even for the different oblique dendrites (and proximal and distal locations in the same dendrites) of a single model. For consistency, we used the same spike detection threshold of -20 mV for all the models.

The synaptic stimulus contains an AMPA and an NMDA receptor-mediated component. As the default synapse, HippoUnit uses the Exp2Syn double exponential synapse built into NEURON for the AMPA component, and its own built-in NMDA receptor model, whose parameters were set according to experimental data from the literature (see the Methods section for more details). In those models that originally do not have any synaptic component (the Bianchi et al 2011 model and all versions of the Golding et al. 2001 model) this default synapse was used. Both the Katz et al. 2009 and the Migliore et al. 2011 models used the Exp2Syn in their simulations, so in their case the time constants of this function were set to the values used in the original publications. As these models did not contain NMDA receptors, the default NMDA receptor model and the default AMPA/NMDA ratio of HippoUnit were used. The Gómez González et al 2011 and the Poirazi et al. 2003 models have their own AMPA and NMDA receptor models and they own AMPA/NMDA ratio values to be tested with.

As shown by the averaged "measured EPSP vs expected EPSP" curves in Fig 9, all three versions of the Golding et al. 2001 model have a jump in the amplitude of the somatic response at the threshold input level, which is the result of the generation of dendritic spikes. However, even these larger average responses do not reach the supralinear region, as it would be expected according to the experimental observations [32]. The reason for this discrepancy is that a dendritic spike was generated in the simulations in only a subset of the stimulated dendrites; in the rest of the dendrites tested, the amplitude of the EPSPs went above the spike detection threshold during the adjustment of the synaptic weight without actually triggering a dendritic spike, which led to the corresponding synaptic strength being incorrectly set for that particular dendrite. Averaging over the results for locations with and without dendritic spikes led to an overall sublinear integration profile.

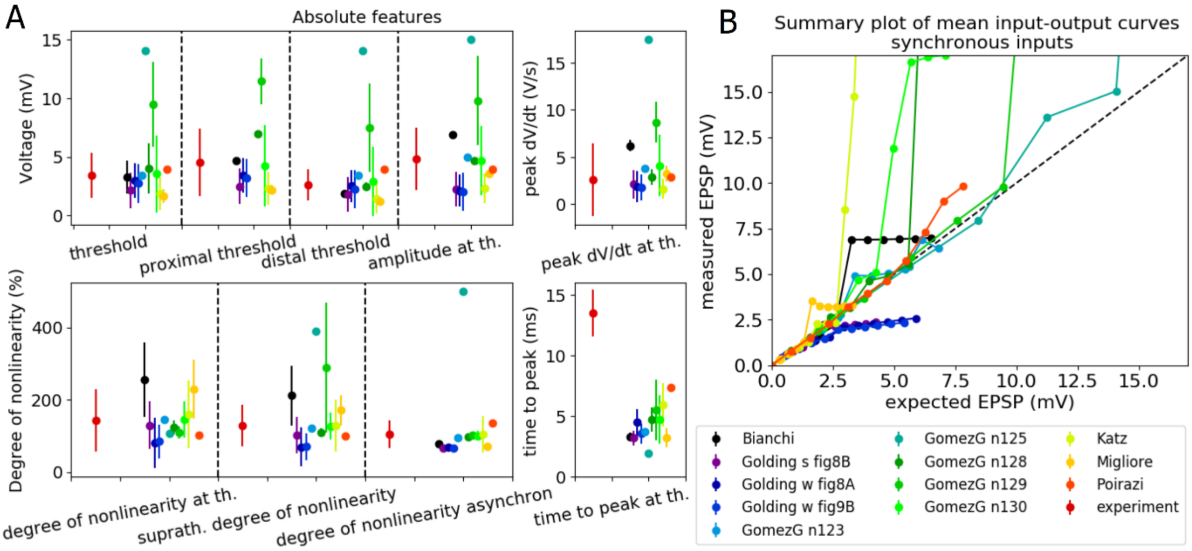

Fig 9: Results from the Oblique Integration Test of HippoUnit applied to published models. (A) Comparison of the responses of the models to experimental results (dark red) according to features of dendritic integration. (B) The averaged input – output curves of all the dendritic locations examined. EPSP amplitudes are measured at the soma.

The Migliore et al. 2011 model performs quite well on this test. In this case, seven dendrites could be tested out of the ten dendrites within the correct distance range because, in the others, the dendritic spike at the threshold input level also elicited a somatic action potential, and therefore these dendrites were excluded from further testing.

In the Katz et al. 2009 model all the selected dendritic locations could be tested, and in most of them the synaptic strength could be adjusted appropriately. For a few dendrites, some input levels higher than the threshold for dendritic spike generation also triggered somatic action potentials. This effect causes the high supralinearity in the “measured EPSP vs expected EPSP” curve in Fig 9, but has no effect on the extracted features.

In the Bianchi et al. 2012 model only one dendrite could be selected, in which very high amplitude dendritic spikes were evoked by the synaptic inputs, making the signal integration highly supralinear.

In the Poirazi et al. 2003 model also only one dendrite could be selected based on its distance from the soma; furthermore, only the distal location could be tested even in this dendrite, as at the proximal location the dendritic action potential at the threshold input level generated a somatic action potential. However, at the distal location, the synaptic strength could not be set correctly. For the synaptic strength chosen by the test, the actual threshold input level where a dendritic spike is first generated is at 4 inputs, but this dendritic AP is too small in amplitude to be detected, and the response to 5 inputs is recognized as the first dendritic spike instead. Therefore, the features that should be extracted at the threshold input level are instead extracted from the voltage response to 5 inputs. In this model this results in a reduced *supralinearity* value, as this feature is calculated one input level higher than the actual threshold. In addition, for even higher input levels dendritic bursts can be observed, which causes large *supralinearity* values in the “measured EPSP vs expected EPSP” curve in Fig 9, but this does not affect the feature values.

Models from Gómez González et al. 2011 were expected to be particularly relevant for this test, as these models were tuned to fit the same data set on which this test is based. However, we encountered an important issue when comparing our test results for these models to the results shown in the paper [53]. In particular, the paper clearly indicates which dendrites were examined, and it is stated that those are at maximum 150 µm from the soma. However, when we measured the distance of these locations from the soma by following the path along the dendrites (as it is done by the test of HippoUnit), we often found it to be larger than 150 µm. We note that when the distance was measured in 3D coordinates rather than along the dendrites, all the dendrites used by Gómez González et al. 2011 appeared to be within 150 µm of the soma, so we assume that this definition was used in the paper. As we consider the path distance to be more meaningful than Euclidean distance in this context, and this was also the criterion used in the experimental study, we consistently use path distance in HippoUnit to find the relevant dendritic segments. Nevertheless, this difference in the selection of dendrites should be kept in mind when the results of this validation for models of Gómez González et al. 2011 are evaluated.

In two versions of the Gómez González et al. 2011 model (those that were adjusted to the n123 and n125 morphologies) only one oblique dendrite matched the experimental criteria and could therefore be selected, and these are not among those that were studied by the developers of the model. In each of these cases the dendritic spike at the proximal location at the input threshold level triggered a somatic action potential, and therefore only the distal location could be tested. In the case of the n125 morphology, the dendritic spikes that appear first for just-suprathreshold input are so small in amplitude that they do not reach the spike detection threshold (-20 mV), and are thus not detected. Therefore, the automatically adjusted synaptic weight is larger than the appropriate value would be, which results in larger somatic EPSPs than expected (see Fig 9). With this synaptic weight the first dendritic spike, and therefore the jump in the "measured EPSP vs expected EPSP" curve to the supralinear region is for 4 synaptic inputs, instead of 5. This is also the case in one of the two selected dendrites of the version of this model with the n128 morphology. Similarly to the Poirazi et al. 2003 model, this results in a lower *degree of nonlinearity at threshold* feature value, than it would be if the feature were extracted at the actual threshold input level (4 inputs) instead of the one which the test attempted to adjust (5 inputs). The *suprathreshold nonlinearity* feature has a high value because at that input level (6 inputs), somatic action potentials are triggered.

In the version of the Gómez González et al. 2011 model that uses the n129 morphology, 10 oblique dendrites could be selected for testing (none of them is among those that its developers used) but only 4 could be tested because, for the rest, the dendritic spike at the threshold input level already elicits a somatic action potential. The synaptic weights required to set the threshold input level to 5 are not found correctly in most cases; the actual threshold input level is at 4 or 3. Suprathreshold nonlinearity is high, because at that input level (6 inputs) somatic action potentials are triggered for some of the examined dendritic locations.

The version of the Gómez González et al. 2011 model that uses the n130 morphology achieves the best (lowest) final score on this test. In this model many oblique dendrites could be selected and tested, including two (179, 189) that the developers used in their simulations [53]. In most cases the synaptic weights are nicely found to set the threshold input level to 5 synapses. For some dendrites there are somatic action potentials at higher input levels, but that does not affect the features.

The value of the *time to peak* feature for each model is much smaller than the experimental value (Fig 9). This is because in each of the models the maximum amplitude of the somatic EPSP is determined by the fast component, caused by the appearance of the dendritic sodium spikes, while in the experimental observation this is rather shaped by the slow NMDA component following the sodium spike.

Overall performance and model comparison

In summary, using HippoUnit, we compared the behavior of several hippocampal CA1 pyramidal cell models available on ModelDB in several distinct domains, and found that all of these models match experimental results well in some domains (typically those that they were originally built to capture) but fit the experimental observation less precisely in others. Fig 10 summarizes the final scores achieved by the different models on the various tests (lower scores indicate a better match in all cases).

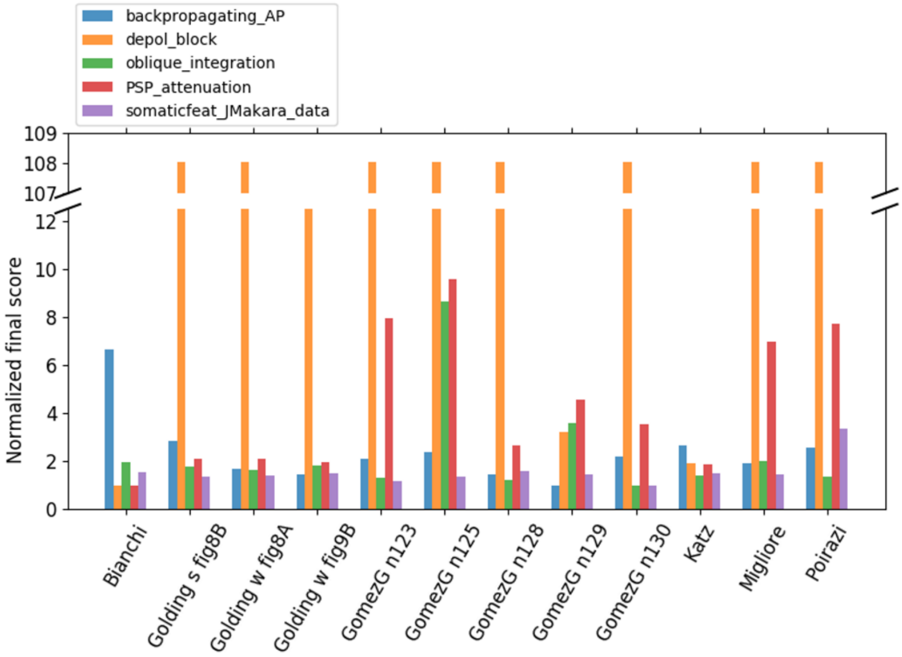

Fig 10: Normalized final scores achieved by the different published models on the various tests of HippoUnit. The final scores of each test are normalized by dividing the scores of each model by the best achieved score on the given test.

Perhaps a bit surprisingly, the different versions of the Golding et al. 2001 model perform quite well on all of the tests (except for the Depolarization Block Test), even though these are the simplest ones among the models in the sense that they contain the smallest number of different types of ion channels. On the other hand, they do not perform outstandingly well on the Back-propagating Action Potential Test, although they were developed to capture the behavior evaluated by this test.

The Bianchi et al. 2012 model is the only one that can produce real depolarization block within the range of input strengths examined by the corresponding test. The success of this model in this test is not surprising because this is the only model that was tuned to reproduce this behavior; on the other hand, the failure of the other models in this respect clearly shows that proper depolarization block requires some combination of mechanisms that are at least partially distinct from those that allow good performance in the other tests. The Bianchi et al. 2012 model achieves a relatively high final error score only on the Back-propagating Action Potential Test, as action potentials seem to propagate too actively in its dendrites, leading to high AP amplitudes even in more distal compartments.

The Gómez González et al. 2011 models were developed to capture the same experimental observations on dendritic integration that are tested by the Oblique Integration Test of HippoUnit, but, somewhat surprisingly, some of its versions achieved quite high error scores on this test, while others perform quite well. This is partly caused by the fact that HippoUnit often selects different dendritic sections for testing from those that were studied by the developers of these models (see above for details). Some of its versions also perform relatively poorly on the PSP-Attenuation Test, similar to the Migliore et al. 2011 and the Poirazi et al. 2003 models. The Katz et al. 2009 model is not outstandingly good in any of the tests, but still achieves relatively good error scores everywhere (although its apparent good performance on the Depolarization Block Test is misleading - see detailed explanation above).

The model files that were used to test the models described above, the detailed validation results (all the output files of HippoUnit), and the Jupyter Notebooks that show how to run the tests of HippoUnit on these models are available in the following Github repository: https://github.com/KaliLab/HippoUnit_demo.

Application of HippoUnit to models built using automated parameter optimization within the Human Brain Project

Besides enabling a detailed comparison of published models, HippoUnit can also be used to monitor the performance of new models at various stages of model development. Here, we illustrate this by showing how we have used HippoUnit within the HBP to systematically validate detailed multi-compartmental models of hippocampal neurons developed using multi-objective parameter optimization methods implemented by the open source Blue Brain Python Optimization Library (BluePyOpt [15]). To this end, we extended HippoUnit to allow it to handle the output of optimization performed by BluePyOpt (see Methods).

Models of CA1 pyramidal cells were optimized using target feature data extracted from the same sharp electrode dataset [3] that was also one of the datasets used by the Somatic Features Test of HippoUnit. However, while during validation all the eFEL features that could be successfully extracted from the data are considered, only a subset of these features was used in the optimization (mostly those that describe the rate and timing of the spikes; e.g., the different inter-spike interval (ISI), time to last/first spike, mean frequency features).

In addition, sharp electrode measurements were also available for several types of interneuron in the hippocampal CA1 region, and models of these interneurons were also constructed using similar automated methods [3]. Using the appropriate observation file and the stimulus file belonging to it, the Somatic Features Test of HippoUnit can also be applied to these models to evaluate their somatic spiking features. The other tests of HippoUnit are currently not applicable to interneurons, mostly due to the lack of appropriate target data.

We applied the tests of HippoUnit to the version of the models published in [3], and to a later version (v4) described in Ecker et al. (2020)[55], which was intended to further improve the dendritic behavior of the models, as this is critical for their proper functioning in the network. The two sets of models were created using the same morphology files and similar optimization methods and protocols. These new optimizations differed mainly in the allowed range for the density of the sodium channels in the dendrites. For the pyramidal cell models a new feature was also introduced in the parameter optimization that constrains the amplitudes of back-propagating action potentials in the main apical dendrite. The new interneuron models also had an exponentially decreasing (rather than constant) density of Na channels, and A-type K channels with more hyperpolarized activation in their dendrites. For more details on the models, see the original publications ([3,55]).

After running all the tests of HippoUnit on both sets of models generated by BluePyOpt, we performed a comparison of the old [3] and the new versions of the models by doing a statistical analysis of the final scores achieved by the models of the same cell type on the different tests. In Fig 11 the median, the interquartile range and the full range of the final error scores achieved by the two versions of the model set are compared. According to the results of the Wilcoxon signed-rank test the new version of the models achieved significantly better scores on the Back-propagating Action Potential test ($p = 0.0046$), on the Oblique Integration Test ($p = 0.0033$), and on the PSP Attenuation Test ($p = 0.0107$), which is the result of reduced dendritic excitability. Moreover, in most of the other cases the behavior of the models improved slightly (but not significantly) with the new version. Only in the case of the Somatic Features test applied to bAC interneurons did the new models perform slightly worse (but still quite well), and this difference was not significant ($p = 0.75$).

These results show the importance of model validation performed against experimental findings, especially those not considered when building the model, in every iteration during the process of model development. This approach can greatly facilitate the construction of models that perform well in a variety of contexts, help avoid model regression, and guide the model building process towards a more robust and general implementation.

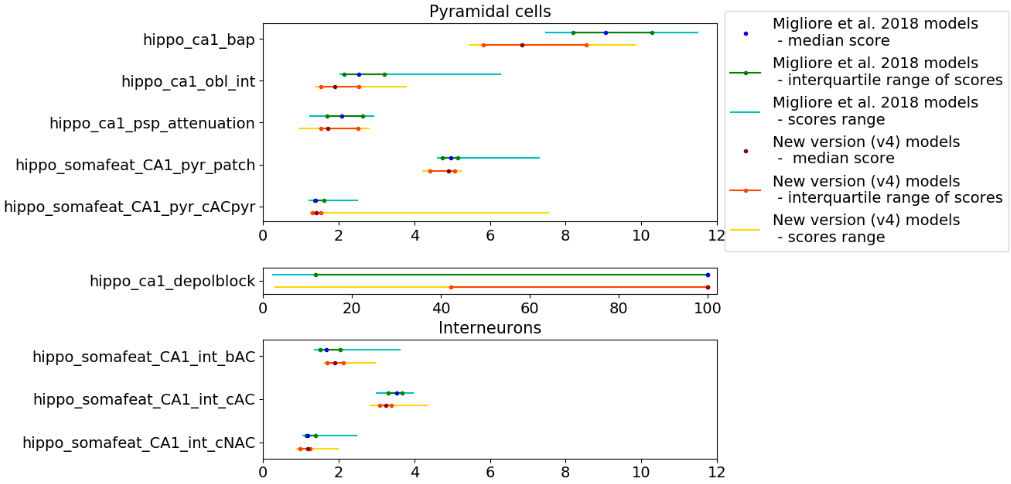

Fig 11: Statistical comparison of the final scores achieved on the different tests by the two versions of hippocampal CA1 models of the same cell types, developed by automated optimization using BluePyOpt.

Integration of HippoUnit into the Validation Framework and the Brain Simulation Platform of the Human Brain Project

The HBP is developing scientific infrastructure to facilitate advances in neuroscience, medicine, and computing [56]. One component of this research infrastructure is the Brain Simulation Platform (BSP) (https://bsp.humanbrainproject.eu), an online collaborative platform that supports the construction and simulation of neural models at various scales. As we argued above, systematic, automated validation of models is a critical prerequisite of collaborative model development. Accordingly, the BSP includes a software framework for quantitative model validation and testing that explicitly supports applying a given validation test to different models and storing the results [57]. The framework consists of a web service, and a set of test suites, which are Python modules based on the SciUnit package. As we discussed earlier, SciUnit uses the concept of capabilities, which are standardized interfaces between the models to be tested and the validation tests. By defining the capabilities to which models must adhere, individual validation tests can be implemented independently of any specific model and used to validate any compatible model despite differences in their internal structures, the language and/or the simulator used. Each test must include a specification of the required model capabilities, the location of the reference (experimental) dataset, and data analysis code to transform the recorded variables (e.g., membrane potential) into feature values that allow the simulation results to be directly and quantitatively compared to the experimental data through statistical analysis. The web services framework [57] supports the management of models, tests, and validation results. It is accessible via web apps within the HBP Collaboratory, and also through a Python client. The framework makes it possible to permanently record, examine and reproduce validation results, and enables tracking the evolution of models over time, as well as comparison against other models in the domain.

Every test of HippoUnit described in this paper has been individually registered in the Validation Framework. The JSON files containing the target experimental data for each test are stored (besides the HippoUnit_demo GitHub repository) in storage containers at the Swiss National Supercomputing Centre (CSCS), where they are publicly available. The location of the corresponding data file is associated with each registered test, so that the data are loaded automatically when the test is run on a model via the Validation Framework. As the Somatic Features Test of HippoUnit was used to compare models against five different data sets (data from sharp electrode measurements in pyramidal cells and interneurons belonging to three different electronic types, and data obtained from patch clamp recordings in pyramidal cells), these are considered to be and have been registered as five separate tests in the Validation Framework.

All the models that were tested and compared in this study (including the CA1 pyramidal cell models from the literature and the BluePyOpt optimized CA1 pyramidal cells and interneurons of the HBP) have been registered and are available in the Model Catalog of the Validation Framework with their locations in the CSCS storage linked to them. In addition to the modifications that were needed to make the models compatible with testing with HippoUnit (described in the section "Methods – Models from literature"), the versions of the models uploaded to the CSCS container also contain an `__init__.py` file. This file implements a python class that inherits all the functions of the `ModelLoader` class of HippoUnit without modification. Its role is to make the validation of these models via the Framework more straightforward by defining and setting the parameters of the `ModelLoader` class (such as the path to the HOC and NMODL files, the name of the section lists, etc.) that otherwise need to be set after instantiating the `ModelLoader` (see the HippoUnit_demo GitHub repository: https://github.com/KaliLab/HippoUnit_demo/tree/master/jupyter_notebooks).

The validation results discussed in this paper have also been registered in the Validation Framework, with all their related files (output figures and JSON files) linked to them. These can be accessed using the Model Validation app of the framework.

The Brain Simulation Platform of the HBP contains several online 'Use Cases', which are available on the platform and help the users to try and use the various established pipelines. The Use Case called 'Hippocampus Single Cell Model Validation' can be used to apply the tests of HippoUnit to models that were built using automated parameter optimization within the HBP.

The Brain Simulation Platform also hosts interactive "Live Paper" documents that refer to published papers related to the models or software tools on the Platform. Live Papers provide links that make it possible to visualize or download results and data discussed in the respective paper, and even to run the associated simulations on the Platform. We have created a Live Paper (https://humanbrainproject.github.io/hbp-bsp-live-papers/2020/saray_et_al_2020/saray_et_al_2020.html) showing the results of the study presented in this paper in more detail. This interactive document provides links to all the output figures and data files resulting from the validation of the models from literature discussed here. This provides a more detailed insight into their behavior individually. Moreover, as part of this Live Paper a HippoUnit Use Case is also available in the form of a Jupyter Notebook, which guides the user through running the validation tests of HippoUnit on the models from literature that are already registered in the Framework, and makes it possible to reproduce the results presented here.

Discussion

Role of validation in collaborative model building

For anatomically and biophysically detailed data-driven neural models to be predictive, it is important that they are able to generalize beyond their original scope. However, most detailed biophysical models to date were built to capture only a few important or interesting properties of a given neuron type. Systematic testing and comparison of the behavior of these models is still rare, and thus it is often unknown how these models would behave when used under different circumstances, and to what extent they can be used to address different scientific questions. As a result, the modeling community still keeps building new models of the same cell type for various purposes, instead of reusing and further developing the already existing ones. On the other hand, in those cases when new models are based on previously published ones, model parameters are often adjusted to fit a new set of experimental data. These adjustments typically alter the ability of the model to capture the experimental data targeted by the original model, but this remains unrecognized because of the lack of comprehensive testing. As we have shown, an illustrative example of this regression issue is the Bianchi et al. 2012 model. This model was mainly based on the Poirazi et al. 2013 model, which was developed to show specific dendritic behaviors and was even tested using data on back-propagating action potentials during its development [51]. However, the test results presented above indicate that when the somatic behavior of this model was adjusted to reproduce experimental observations on depolarization block, it lost the ability to show realistic back-propagation of action potentials into the apical dendrite. In addition, some publications on neuronal models simply state that the model has been validated against electrophysiological data, but the details of these validations (such as the methods used, the experimental data considered or even the results) are usually not shared. Finally, more systematic testing and coordinated development would enable building consensus (community) models, which would aim to capture a wide range of experimental observations.

The framework for implementing unit tests that make it possible to automatically and systematically compare the behavior of models to experimental data already exists (SciUnit [18]). Furthermore, the recently developed Validation Framework of the HBP makes it possible to collect neural models and validation tests, and supports the application of the registered tests to the registered models. Most importantly, it makes it possible to save the validation results and link them to the models in the Model Catalog, making them publicly available and traceable for the modeling community.

Our goal was to contribute to this new approach in collaborative model building by developing a validation suite to test the behavior of models of the hippocampal CA1 pyramidal neuron, which is one of the most studied cell types in the brain. Here we presented how we applied HippoUnit to test and compare the behavior of several models of this cell type available on ModelDB [17] in several distinct domains against electrophysiological data available in the literature. Through the example of the models optimized within the HBP we also showed how a test suite like HippoUnit can be a useful tool in tracing the performance of models during the process of their development.

Although we consider it essential to evaluate the generalization properties of neural models and test them in as many domains as possible, it is important to emphasize that a high error score on a given validation test using a particular experimental dataset does not mean that the model is not good enough or cannot be useful for a variety of purposes (including the ones it was originally developed for). The discrepancy between the target data and the model's behavior, as quantified by the validation tests, may be due to several different reasons. First, all experimental data contain noise and may have systematic biases associated with the experimental methods employed. Sometimes the experimental protocol is not described in sufficient detail to allow its faithful reproduction in the simulations. It may also occur that a model is based on experimental data that were obtained under conditions that are substantially different from the conditions for the measurement of the validation target dataset. Using different recording techniques, such as sharp electrode or patch clamp recordings or the different circumstances of the experiments (e.g., the strain, age, and sex of the animal, or the temperature during measurement) can heavily affect the experimental results. Furthermore, the post-processing of the recorded electrophysiological data can also alter the results. For these reasons, probably no single model should be expected to achieve an arbitrarily low score on all of the validation tests developed for a particular cell type. Keeping this in mind, it is important that the modelers decide which properties of the cell type are relevant for them, and what experimental conditions they aim to mimic. Validation results should be interpreted or taken into account accordingly, and the tests themselves may need to be adapted.

By providing the software tools and examples on how to validate the different models, we hope to encourage the modeling community to use more systematic testing during model development, in order to create neural models that generalize better, and to make the process of model building more reproducible and transparent.

Uniform model formats reduce the costs of validation

Although HippoUnit is built in a way that its tests are, in principle, model-agnostic, so that the implementation of the tests does not depend on model implementation, it still required a considerable effort to create the standalone versions of the models from literature to be tested, even though all of the selected models were developed for the NEURON simulator. This is because each model has a different file structure and internal logic that needs to be understood in order to create an equivalent standalone version. When the section lists of the main dendritic types do not exist, the user needs to create them by extensively analyzing the morphology and even doing some coding. In order to reduce the costs of systematic validation, models would need to be expressed in a format that is uniform and easy to test. As HippoUnit already has its capability functions implemented in a way that it is able to handle models developed in NEURON, the only requirement for such models is that they should contain a HOC file that describes the morphology (including the section lists for the main dendritic types of the dendritic tree) and all the biophysical parameters of the model, without any additional simulations, GUIs or run-time modifications. Currently, such a standalone version of the models is not made available routinely in publications or on-line databases, but could be added by the creators of the models with relatively little effort.

On the other hand, applying the tests of HippoUnit to models built in other languages requires the re-implementation of the capability functions that are responsible for running the simulations on the model (see Methods). In order to save the user from this effort, it would be useful to publish neuronal models in a standard and uniform format that is simulator independent and allows general use in a variety of paradigms. This would allow an easier and more transparent process of community model development and validation, as it avoids the need of reimplementation of parts of software tools (such as validation suites), and the creation of new, (potentially) non-traced software versions. This approach is already initiated for neurons and neuronal networks by the developers of NeuroML [58], NineML [59], PyNN [60], Sonata [61], and Brian [62]. Once a large set of models becomes available in these standardized formats, it will be straightforward to extend HippoUnit (and other similar test suites) to handle these models.

Extensibility of HippoUnit

As HippoUnit is based on the SciUnit package [18] it inherits SciUnits's modular structure. This means that a test is usually composed of four main classes: the test class, the model class, the capabilities class and the score class (as described in more detail in the Methods section). Thanks to this structure it is easy to extend HippoUnit with new tests by implementing them in new test classes and adding the capabilities and scores needed. The methods of the new capabilities can be implemented in the `ModelLoader` class, which is a generalized Model class for models built in the NEURON simulator, or in a newly created Model class specific to the model to be tested.

As HippoUnit is open-source and is shared on GitHub, it is possible for other developers, modelers or scientists to modify or extend the test suite working on their own forks of the repository. If they would like to directly contribute to HippoUnit, a 'pull request' can be created to the main repository.

Generalization possibilities of the tests of HippoUnit

In the current version of HippoUnit most of the validation tests can only be used to test models of hippocampal CA1 pyramidal cells, as the observation data come from electrophysiological measurements of this cell type and the tests were designed to follow the experimental protocols of the papers from which these data derive. However, with small modifications most of the tests can be used for other cell types, or with slightly different stimulation protocols, if there are experimental data available for the features or properties tested.

The Somatic Features Test can be used for any cell type and with any current step injection protocol even in its current form using the appropriate data and configuration files. These two files must be in agreement with each other; in particular, the configuration file should contain the parameters of the step current protocols (delay, duration, amplitude) used in the experiments from which the feature values in the data file derive. In this study this test was used with two different experimental protocols (sharp electrode measurements and patch clamp recordings that used different current step amplitudes and durations), and for testing four different cell types (hippocampal CA1 PC and interneurons).

In the current version of the Depolarization Block Test the properties of the stimulus (delay, duration, amplitudes) are hard-coded to reproduce the experimental protocol used in a study of CA1 PCs [24]. However, the test could be easily modified to read these parameters from a configuration file like in the case of other tests, and then the test could be applied to other cell types if data from similar experimental measurements are available.

As the Back-propagating AP Test examines the back-propagation efficacy of action potentials in the main apical dendrite (trunk), it is mainly suitable for testing pyramidal cell models; however, it can be used for PC models from other hippocampal or cortical regions, potentially using different distance ranges of the recording sites. If different distances are used, the feature names (‘AP1_amp_X’ and ‘APlast_amp_X’, where X is the recording distance) in the observation data file and the recording distances given in the stimuli file must be in agreement. Furthermore, it would also be possible to set a section list of other dendritic types instead of the trunk to be examined by the test. This way, models of other cell types (with dendritic trees qualitatively different from those of PCs) could also be tested. The frequency range of the spike train (10 – 20 Hz, preferring values closest to 15 Hz) is currently hard-coded in the function that automatically finds the appropriate current amplitude, but the implementation could be made more flexible in this case as well.

The PSP Attenuation Test is quite general. Both the distances and tolerance values that determine the stimulation locations on the dendrites and the properties of the synaptic stimuli are given using the configuration file. Here again the feature names in the observation data file (‘attenuation_soma/dend_x_um’, where x is the distance from the soma) must fit the distances of the stimulation locations in the configuration file when one uses the tests with data from a different cell type or experimental protocol. Similarly to the Back-propagating AP Test the PSP Attenuation Test also examines the main apical dendrite (trunk), but could be altered to use section lists of other dendritic types.

The Oblique Integration Test is very specific to the experimental protocol of [32]. There is no configuration file used here, but the synaptic parameters (of the `ModelLoader` class) and the number of synapses to which the model should first generate a dendritic spike (‘threshold_index’ parameter of the test class) can be adjusted by the user after instantiating the `ModelLoader` and the test classes respectively. The time intervals between the inputs (synchronous (0.1 ms), asynchronous (2.0 ms)) are currently hard-coded in the test.

HippoUnit has been used mainly to test models of rat hippocampal CA1 pyramidal cells as described above. However, having the appropriate observation data, most of its tests could easily be adapted to test models of different cell types, even in cases when the experimental protocol is slightly different from the currently implemented ones. The extent to which a test needs to be modified in order to test models of other cell types depends on how much the behavior of the new cell type differs from the behavior of CA1 pyramidal cells, and to what extent the protocol of the experiment differs from the ones we used as the bases of comparison in the current study.

Acknowledgements

We thank Judit Makara and her group in the Laboratory of Neuronal Signaling, Institute of Experimental Medicine, Hungary for the patch clamp recording data used in this study. We also thank Luca Tar, a member of our group, for her help in testing the validation tests, and in literature review, and for useful discussions.

References

1. Einevoll GT, Destexhe A, Diesmann M, Grün S, Jirsa V, de Kamps M, et al. The Scientific Case for Brain Simulations. Neuron. 2019;102: 735–744. doi:10.1016/j.neuron.2019.03.027

2. Káli S, Freund TF. Distinct properties of two major excitatory inputs to hippocampal pyramidal cells: A computational study. European Journal of Neuroscience. 2005;22: 2027–2048. doi:10.1111/j.1460-9568.2005.04406.x

3. Migliore R, Lupascu CA, Bologna LL, Romani A, Courcol JD, Antonel S, et al. The physiological variability of channel density in hippocampal CA1 pyramidal cells and interneurons explored using a unified data-driven modeling workflow. PLoS Computational Biology. 2018;14: 1–25. doi:10.1371/journal.pcbi.1006423

4. Hay E, Hill S, Schürmann F, Markram H, Segev I. Models of neocortical layer 5b pyramidal cells capturing a wide range of dendritic and perisomatic active properties. PLoS Computational Biology. 2011;7. doi:10.1371/journal.pcbi.1002107

5. Herz AVM, Gollisch T, Machens CK, Jaeger D. Modeling single-neuron dynamics and computations: A balance of detail and abstraction. Science. 2006;314: 80–85. doi:10.1126/science.1127240

6. Poirazi P, Brannon T, Mel BW. Pyramidal neuron as two-layer neural network. Neuron. 2003;37: 989–999. doi:10.1016/S0896-6273(03)00149-1

7. Bower JM. The 40-year history of modeling active dendrites in cerebellar purkinje cells: Emergence of the first single cell “community model.” Frontiers in Computational Neuroscience. 2015;9: 1–18. doi:10.3389/fncom.2015.00129

8. Traub RD, Wong RKS, Miles R, Michelson H. A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances. Journal of Neurophysiology. 1991;66: 635–650. doi:10.1152/jn.1991.66.2.635

9. Almog M, Korngreen A. A quantitative description of dendritic conductances and its application to dendritic excitation in layer 5 pyramidal neurons. Journal of Neuroscience. 2014;34: 182–196. doi:10.1523/JNEUROSCI.2896-13.2014

10. Markram H, Muller E, Ramaswamy S, Reimann MW, Abdellah M, Sanchez CA, et al. Reconstruction and Simulation of Neocortical Microcircuitry. Cell. 2015;163: 456–492. doi:10.1016/j.cell.2015.09.029

11. Traub RD, Contreras D, Cunningham MO, Murray H, LeBeau FEN, Roopun A, et al. Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts. Journal of Neurophysiology. 2005;93: 2194–2232. doi:10.1152/jn.00983.2004

12. Schneider CJ, Bezaire M, Soltesz I. Toward a full-scale computational model of the rat dentate gyrus. Frontiers in Neural Circuits. 2012;6: 1–8. doi:10.3389/fncir.2012.00083

13. Bezaire MJ, Raikov I, Burk K, Vyas D, Soltesz I. Interneuronal mechanisms of hippocampal theta oscillations in a full-scale model of the rodent CA1 circuit. eLife. 2016;5: 1–106. doi:10.7554/eLife.18566

14. Friedrich P, Vella M, Gulyás AI, Freund TF, Káli S. A flexible, interactive software tool for fitting the parameters of neuronal models. Frontiers in Neuroinformatics. 2014;8: 1–19. doi:10.3389/fninf.2014.00063

15. van Geit W, Gevaert M, Chindemi G, Rössert C, Courcol JD, Muller EB, et al. BluePyOpt: Leveraging open source software and cloud infrastructure to optimise model parameters in neuroscience. Frontiers in Neuroinformatics. 2016;10: 1–18. doi:10.3389/fninf.2016.00017

16. Vanier MC, Bower JM. A comparative survey of automated parameter-search methods for compartmental neural models. Journal of Computational Neuroscience. 1999;7: 149–171. doi:10.1023/A:1008972005316

17. McDougal RA, Morse TM, Carnevale T, Marenco L, Wang R, Migliore M, et al. Twenty years of ModelDB and beyond: building essential modeling tools for the future

of neuroscience. Journal of Computational Neuroscience. 2017;42: 1–10. doi:10.1007/s10827-016-0623-7

18. Omar C, Aldrich J, Gerkin RC. Collaborative infrastructure for test-driven scientific model validation. 36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings. 2014; 524–527. doi:10.1145/2591062.2591129

19. Gerkin R, Omar C. NeuroUnit: Validation Tests for Neuroscience Models. Frontiers in Neuroinformatics. 2013. doi:10.3389/conf.fninf.2013.09.00013

20. Appukuttan S, Garcia PE, Davison AP. MorphoUnit. Zenodo; 2020. doi:https://doi.org/10.5281/zenodo.3862936

21. Appukuttan S, Dainauskas J, Davison AP. SynapseUnit. Zenodo; 2020. doi:http://doi.org/10.5281/zenodo.3862944

22. Garcia PE, Davison AP. HippoNetworkUnit. Zenodo; 2020. doi:http://doi.org/10.5281/zenodo.3886484

23. Sharma BL, Davison AP. CerebUnit. Zenodo; 2020. doi:http://doi.org/10.5281/zenodo.3885673

24. Bianchi D, Marasco A, Limongiello A, Marchetti C, Marie H, Tirozzi B, et al. On the mechanisms underlying the depolarization block in the spiking dynamics of CA1 pyramidal neurons. Journal of Computational Neuroscience. 2012;33: 207–225. doi:10.1007/s10827-012-0383-y

25. Spruston N, Schiller Y, Stuart G, Sakmann B. Activity-Dependent Action Potential Invasion and Calcium Influx into Hippocampal CA1 Dendrites. Science. 1995;268: 297–300. doi:10.1126/science.7716524

26. Golding NL, Kath WL, Spruston N. Dichotomy of action-potential backpropagation in CA1 pyramidal neuron dendrites. Journal of Neurophysiology. 2001;86: 2998–3010. doi:10.1152/jn.2001.86.6.2998

27. Gasparini S, Losonczy A, Chen X, Johnston D, Magee JC. Associative pairing enhances action potential back-propagation in radial oblique branches of CA1 pyramidal neurons. Journal of Physiology. 2007;580: 787–800. doi:10.1113/jphysiol.2006.121343

28. Magee JC, Cook EP. Somatic EPSP amplitude is independent of synapse location in hippocampal pyramidal neurons. Nature Neuroscience. 2000;3: 895–903. doi:10.1038/78800

29. Gasparini S, Magee JC. State-dependent dendritic computation in hippocampal CA1 pyramidal neurons. Journal of Neuroscience. 2006;26: 2088–2100. doi:10.1523/JNEUROSCI.4428-05.2006

30. Ariav G, Polsky A, Schiller J. Submillisecond precision of the input-output transformation function mediated by fast sodium dendritic spikes in basal dendrites of CA1 pyramidal neurons. Journal of Neuroscience. 2003;23: 7750–7758. doi:10.1523/jneurosci.23-21-07750.2003

31. Takahashi H, Magee JC. Pathway Interactions and Synaptic Plasticity in the Dendritic Tuft Regions of CA1 Pyramidal Neurons. Neuron. 2009;62: 102–111. doi:10.1016/j.neuron.2009.03.007

32. Losonczy A, Magee JC. Integrative Properties of Radial Oblique Dendrites in Hippocampal CA1 Pyramidal Neurons. Neuron. 2006;50: 291–307. doi:10.1016/j.neuron.2006.03.016

33. Hines ML, Carnevale NT. The NEURON Simulation Environment. Neural Computation. 1997;9: 1179–1209. doi:https://doi.org/10.1162/neco.1997.9.6.1179

34. Druckmann S, Banitt Y, Gidon A, Schrümann F, Markram H, Segev I. A novel multiple objective optimization framework for constraining conductance-based neuron models by experimental data. Frontiers in Neuroscience. 2007;1: 7–18. doi:10.3389/neuro.01.1.1.001.2007

35. van Geit W, Moor R, Ranjan R, Roessert C, Riquelme L. Electrophys Feature Extraction Library. 2020. [cited 25. March 2020] GitHub repository [Internet] Available: https://github.com/BlueBrain/eFEL

36. Tripathy SJ, Savitskaya J, Burton SD, Urban NN, Gerkin RC. NeuroElectro: A window to the world's neuron electrophysiology data. Frontiers in Neuroinformatics. 2014;8: 1–11. doi:10.3389/fninf.2014.00040

37. Wheeler DW, White CM, Rees CL, Komendantov AO, Hamilton DJ, Ascoli GA. Hippocampome.org: A knowledge base of neuron types in the rodent hippocampus. eLife. 2015;4: 1–28. doi:10.7554/eLife.09960

38. Staff NP, Jung HY, Thiagarajan T, Yao M, Spruston N. Resting and active properties of pyramidal neurons in subiculum and CA1 of rat hippocampus. Journal of Neurophysiology. 2000;84: 2398–2408. doi:10.1152/jn.2000.84.5.2398

39. Dougherty KA, Islam T, Johnston D. Intrinsic excitability of CA1 pyramidal neurones from the rat dorsal and ventral hippocampus. Journal of Physiology. 2012;590: 5707–5722. doi:10.1113/jphysiol.2012.242693

40. Malik R, Dougherty KA, Parikh K, Byrne C, Johnston D. Mapping the electrophysiological and morphological properties of CA1 pyramidal neurons along the longitudinal hippocampal axis. Hippocampus. 2016;26: 341–361. doi:10.1002/hipo.22526

41. McDermott CM, Hardy MN, Bazan NG, Magee JC. Sleep deprivation-induced alterations in excitatory synaptic transmission in the CA1 region of the rat hippocampus. Journal of Physiology. 2006;570: 553–565. doi:10.1113/jphysiol.2005.093781

42. Graves AR, Moore SJ, Bloss EB, Mensh BD, Kath WL, Spruston N. Hippocampal Pyramidal Neurons Comprise Two Distinct Cell Types that Are Countermodulated by Metabotropic Receptors. Neuron. 2012;76: 776–789. doi:10.1016/j.neuron.2012.09.036

43. Golding NL, Mickus TJ, Katz Y, Kath WL, Spruston N. Factors mediating powerful voltage attenuation along CA1 pyramidal neuron dendrites. Journal of Physiology. 2005;568: 69–82. doi:10.1113/jphysiol.2005.086793

44. Bormann I. DigitizeIt: Digitizer software - digitize a scanned graph or chart into (x,y)-data. 2020. [cited 25. March 2020] [Internet] Available: https://www.digitizeit.de/

45. Jahr CE, Stevens CF. Voltage dependence of NMDA-activated macroscopic conductances predicted by single-channel kinetics. Journal of Neuroscience. 1990;10: 3178–3182. doi:10.1523/jneurosci.10-09-03178.1990

46. Hestrin S, Nicoll RA, Perkel DJ, Sah P. Analysis of excitatory synaptic action in pyramidal cells using whole-cell recording from rat hippocampal slices. Physiology. 1990; 203–225. doi:10.1113/jphysiol.1990.sp017980

47. Korinek M, Sedlacek M, Cais O, Dittert I, Vyklicky L. Temperature dependence of N-methyl-d-aspartate receptor channels and N-methyl-d-aspartate receptor excitatory postsynaptic currents. Neuroscience. 2010;165: 736–748. doi:10.1016/j.neuroscience.2009.10.058

48. Gevaert M, Kanari L, Palacios J, Zisis E, Coste B. NeuroM. 2020. [cited 25. March 2020] GitHub repository [Internet] Available: https://github.com/BlueBrain/NeuroM

49. Katz Y, Menon V, Nicholson DA, Geinisman Y, Kath WL, Spruston N. Synapse Distribution Suggests a Two-Stage Model of Dendritic Integration in CA1 Pyramidal Neurons. Neuron. 2009;63: 171–177. doi:10.1016/j.neuron.2009.06.023

50. Migliore M, de Blasi I, Tegolo D, Migliore R. A modeling study suggesting how a reduction in the context-dependent input on CA1 pyramidal neurons could generate schizophrenic behavior. Neural Networks. 2011;24: 552–559. doi:10.1016/j.neunet.2011.01.001

51. Poirazi P, Brannon T, Mel BW. Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell. Neuron. 2003;37: 977–987. doi:10.1016/S0896-6273(03)00148-X

52. Shah MM, Migliore M, Valencia I, Cooper EC, Brown DA. Functional significance of axonal Kv7 channels in hippocampal pyramidal neurons. Proceedings of the National Academy of Sciences of the United States of America. 2008;105: 7869–7874. doi:10.1073/pnas.0802805105

53. Gómez González JF, Mel BW, Poirazi P. Distinguishing linear vs. non-linear integration in CA1 radial oblique dendrites: It's about time. Frontiers in Computational Neuroscience. 2011;5: 1–12. doi:10.3389/fncom.2011.00044

54. Jeffrey M. Perkel. Why Jupyter is data scientists' computational notebook of choice. Nature. 2018; 5–6. Available: https://www.nature.com/articles/d41586-018-07196-1

55. Ecker A, Romani A, Sáray S, Káli S, Migliore M, Mercer A, et al. Data-driven integration of hippocampal CA1 synapse physiology in silico. Hippocampus. 2020. doi: Forthcoming

56. Amunts K, Ebell C, Muller J, Telefont M, Knoll A, Lippert T. The Human Brain Project: Creating a European Research Infrastructure to Decode the Human Brain. Neuron. 2016;92: 574–581. doi:10.1016/j.neuron.2016.10.046

57. Fragnaud H, Gonin J, Duperrier J, Legouee E, Davison AP, Appukuttan S. hbp-validation-framework. Zenodo; 2020. doi:http://doi.org/10.5281/zenodo.3888123

58. Gleeson P, Crook S, Cannon RC, Hines ML, Billings GO, Farinella M, et al. NeuroML: A language for describing data driven models of neurons and networks with a high degree of biological detail. PLoS Computational Biology. 2010;6: 1–19. doi:10.1371/journal.pcbi.1000815

59. Raikov I. NineML – a description language for spiking neuron network modeling: the abstraction layer. BMC Neuroscience. 2010;11: 2202. doi:10.1186/1471-2202-11-s1-p66

60. Davison AP, Brüderle D, Eppler J, Kremkow J, Muller E, Pecevski D, et al. PyNN: A common interface for neuronal network simulators. Frontiers in Neuroinformatics. 2009;2: 1–10. doi:10.3389/neuro.11.011.2008

61. Dai K, Hernando J, Billeh YN, Gratiy SL, Planas J, Davison A, et al. The SONATA Data Format for Efficient Description of Large-Scale Network Models. PLoS Computational Biology. 2019;16: 1–24. doi:10.2139/ssrn.3387685

62. Stimberg M, Brette R, Goodman DFM. Brian 2, an intuitive and efficient neural simulator. eLife. 2019;8: 1–41. doi:10.7554/eLife.47314

Supporting information

S1 Appendix. Example of running the SomaticFeaturesTest of HippoUnit using a Jupyter notebook

As the first step HippoUnit's test classes and `ModelLoader` class, along with a few additional Python packages must be imported:

```
from __future__ import print_function #needed only in Python 2
% matplotlib inline

from hippounit.utils import ModelLoader
from hippounit import tests
```

```python

import pkg_resources
import json
import collections
import numpy
```

Then the path to external mechanisms used by the Neuron implementation of the model (NMODL files) needs to be provided, which will be an argument to the `ModelLoader` class, so that the NMODL files can be compiled when the `ModelLoader` class is instantiated (if they are not compiled yet). Next, the variables related to the model are set. The initial voltage (`v_init`) and temperature (`celsius`) values specific to the model need to be set; otherwise, the default value in the corresponding capability method of the `ModelLoader` will be used. Setting the `cvode_active` boolean parameter to `True` or `False`, the user can decide to run the simulations using variable or fixed time step, respectively.

```python
# path to NMODL files
mod_files_path = "/home/saray/published_models/Ca1_Bianchi_2012/experiment/"

#all the outputs will be saved here. It will be an argument to the test.
base_directory = '/mnt/csoport31-2/Modellezo_csapat/Sara/published_models_validation_results/'

#Load cell model
model = ModelLoader(mod_files_path = mod_files_path )

# outputs will be saved in subfolders named like this:
model.name="Bianchi_et_al_2012"

# path to hoc file
# the model must not display any GUI!!
model.hocpath = "/home/saray/published_models/Ca1_Bianchi_2012/experiment/main_model.hoc"

# If the hoc file doesn't contain a template, this must be None (the default value is None)
```

```
model.template_name = None

# model.SomaSecList_name should be None, if there is no Section List in the model for the soma, or if the name of the soma section is given by setting model.soma (the default value is None)
model.SomaSecList_name = None
# if the soma is not in a section list or to use a specific somatic section, add its name here:
model.soma = 'soma[0]'

# For the PSP Attenuation Test, and Back-propagating AP Test a section list containing the trunk sections is needed
model.TrunkSecList_name = 'apical_trunk_list'
# For the Oblique Integration Test a section list containing the oblique dendritic sections is needed
model.ObliqueSecList_name = 'oblique_dendrites'
#  This will be argument to those tests, where dendritic locatins are selected according to distances. If not set, the end of the above given soma section will be used as reference point for distance determination
trunk_origin = ['soma[0]', 1]

model.v_init = -70
model.celsius = 34

# It is possible to run the simulations using variable time step (default for this is False)
model.cvode_active = True
```

The target experimental data and the configuration file are loaded from the JSON files to the *observation* and *config* dictionaries, which are arguments of the test class:

```
# Load target data
with open('/home/saray/target_features/feat_CA1_pyr_cACpyr_more_features.json') as f:
    observation  = json.load(f, object_pairs_hook=collections.OrderedDict)

# Load stimuli file
ttype = "CA1_pyr_cACpyr"
```

```

stim_file = pkg_resources.resource_filename("hippounit", "tests/stimuli/somafeat_stim/stim_" + ttype + ".json")
with open(stim_file, 'r') as f:
    config = json.load(f, object_pairs_hook=collections.OrderedDict)
```

Then the test class is instantiated, and its `judge()` function (inherited from SciUnit) is called to run the test. The number of parallel processes to be used can be controlled by the user by setting the `test.npool` parameter.

```
# Instantiate test class
test = tests.SomaticFeaturesTest(observation=observation, config=config, force_run=False, show_plot=True, save_all = True, base_directory=base_directory)

# test.specify_data_set is added to the name of the subdirectory (somaticfeat), so test runs using different data sets can be saved into different directories
test.specify_data_set = 'UCL_data'

# Number of parallel processes
test.npool = 30
#Run the test
score = test.judge(model)
#Summarize and print the score achieved by the model on the test using SciUnit's summarize function
score.summarize()
```

For further details on how to run the different tests of HippoUnit for the different models, see the Jupyter Notebooks available here: https://github.com/KaliLab/HippoUnit_demo/tree/master/jupyter_notebooks.