Working prototype of a Collaboratory app for graphical model building (D9.1.1 – SGA2)

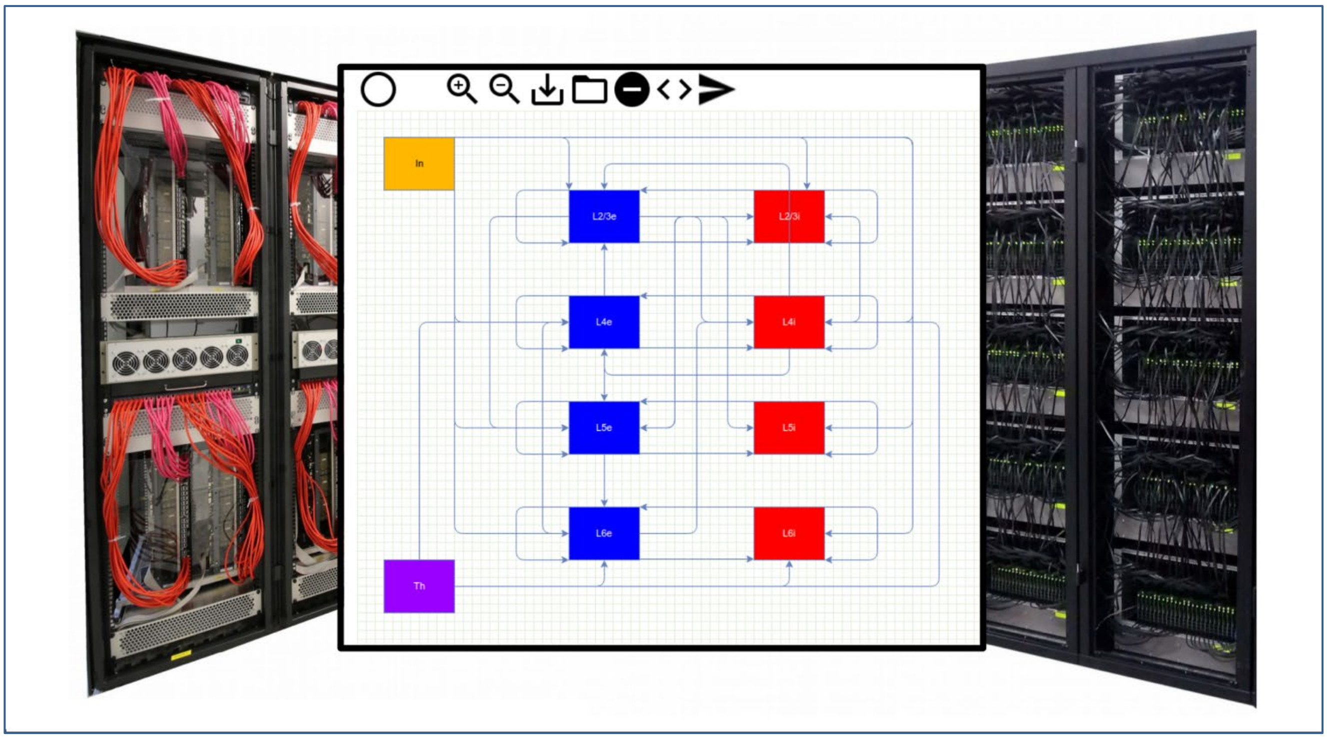


Figure 1: A screenshot of the PyNN model builder app, showing a model of a cortical column.

The model can be emulated on the BrainScaleS (left) and simulated on SpiNNaker (right), with simulations launched directly from the app.

Project Number:	785907	Project Title:	Human Brain Project SGA2
Document Title:	Working prototype of a Collaboratory app for graphical model building		
Document Filename:	D9.1.1 (D57.1 D90) SGA2 M12 ACCEPTED 190723		
Deliverable Number:	SGA2 D9.1.1 (D57.1, D90)		
Deliverable Type:	Other: software		
Work Packages:	WP9.1		
Dissemination Level:	PU = Public		
Planned Delivery Date:	SGA2 M12 / 31 Mar 2019		
Actual Delivery Date:	SGA2 M12 / 31 Mar 2019; ACCEPTED 23 Jul 2019		
Author(s):	Matthieu SENOVILLE, CNRS (P10), Andrew DAVISON, CNRS (P10), Michael THIES, UNIBI (P72)		
Compiled by:			
Contributor(s):	Jonathan DUPERRIER, CNRS (P10). Lead developer.		
SciTechCoord Review:	Martin TELEFONT, EPFL (P1)		
Editorial Review:	Guy WILLIS, EPFL (P1)		
Description in GA:	Collaboratory app usable by an SP9 member outside the developer team.		
Abstract:	This technical report summarises the work realised in developing a prototype Collaboratory app for graphical spiking network model building and neuromorphic simulation.		
Keywords:	Neuromorphic, Collaboratory, PyNN, GUI		
Target Users/Readers:	computational neuroscience community, students		

Table of Contents

Table of Figures

1. Introduction

The Neuromorphic Computing Platform executes simulations for which the model description and simulation protocol must be written as Python scripts, using the PyNN [Ref 1] application programming interface (API). However, PyNN necessarily requires familiarity with computer programming, which reduces the accessibility of both the models and the modelling approach for researchers and students lacking experience in programming. We here present an approach where users can create of models of point spiking neurons using a graphical representation of the model, without any programming, and then run simulations of these models on the HBP neuromorphic computing systems BrainScaleS and SpiNNaker. The app can also be used to generate and download scripts for the NEST and NEURON simulators. In this report, we present the first release of the "PyNN model builder" app from a user's perspective and then give a brief technical description of the implementation.

2. Description of the Collaboratory app

2.1 General overview

The PyNN model builder app provides a graphical interface for network modelling, with a one-to-one mapping of boxes representing Populations. The app functionality consists of 8 main functions, available in the toolbar at the top of the main programme window. Below is the mxGraph canvas where the user will build his/her network. The functions comprise:

- **Population creation:** One of the main functions of the toolbar. The default level of abstraction in PyNN is not the single neuron, but a Population of neurons of a given type, represented by the Population class. This function allows the creation of a Population, by clicking on the button, then dragging and dropping onto the canvas. A Population is represented by a rectangle. The left mouse button selects and allows these objects to be moved in the 2D graphic.
- **Zoom In:** This function is used for viewing a detailed part of the canvas, especially in case of a network with a large number of Populations.
- **Zoom Out:** The opposite of the function "Zoom In"; the purpose here is to view the entire canvas.
- **Save the project:** the model described can be saved in XML format. The resulting file contains the diagrams representing the neural network, as well as its complete description (Populations, Projections and all associated parameters).
- **Load the project:** the user can load an existing project in XML format (see above) in order to use directly or extend it.
- **Clean the project:** This function is used to remove everything and restart a project from scratch. All diagrams and network descriptions are deleted.
- **Create a Python script:** the purpose is to generate a PyNN description of the model. As the PyNN API provides a uniform interface to different simulators and hardware, the app generates a Python script (.py) compatible with, among others, NEST [Ref. 7], SpiNNaker and BrainScaleS [Ref. 8]. The user must specify the name of the Python file, the backend used and a simulation duration for a given number of milliseconds.
- **Job submission:** This function is used to submit directly the model to the Neuromorphic Platform. As the creation of the PyNN script above, the user must specify the name of the Python file, the neuromorphic hardware (SpiNNaker or BrainScaleS) used and a length of time, corresponding to a given number of milliseconds simulated. This function requires a compute quota for one or more of the HBP Neuromorphic Computing systems, which can be requested via the Resource Manager app, installed in the same collab workspace.

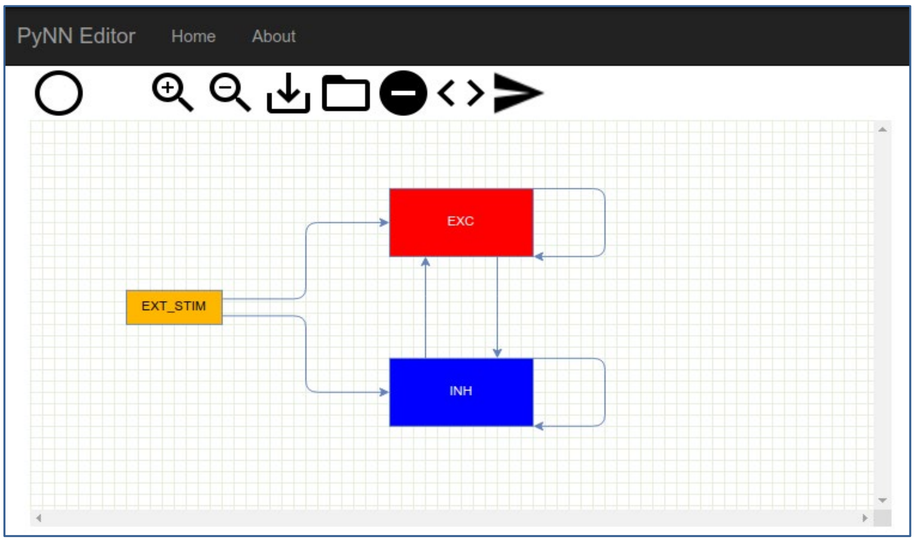


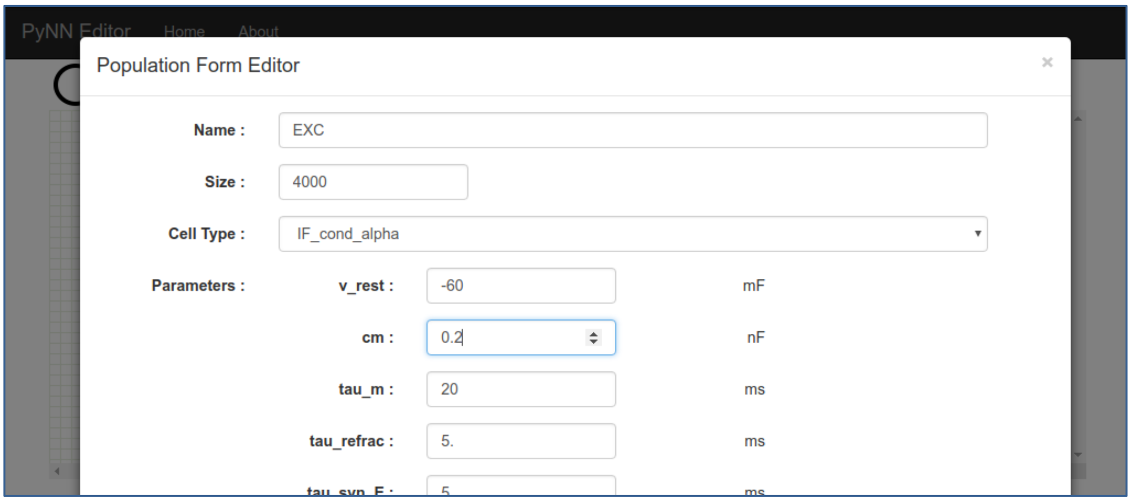
Figure 2: Screenshot of the app

2.2 Network description

The structural overview of a model described with PyNN is always as follows: the Populations of neurons comprising the model and the Projections (connections) between them. It is also possible to record the times of action potentials, and the values of state variables, of any neuron in the network. These are possible in our graphical description of the model.

2.2.1 *Populations*

Populations are first created using the associated function on the toolbar. By clicking with the right mouse button and selecting "Configure Population", a new window appears: this is a form which is used to determine all the parameters of a Population, including the Population name, the number of neurons in the Population, and the type of cell used by the Population. The form provides a drop down menu to select one of the “standard” cell types (see Standard models of PyNN). These work the same across all backend simulators (IF_curr_exp, IF_curr_alpha, IF_cond_alpha, IF_cond_exp, HH_cond_exp., etc.). The initial values of the state variables are individually set by hand or directly, by using the input's spin boxes. The form has default values corresponding to those of PyNN. At the end, a list of checkboxes allows the selection of the parameters to be recorded (e.g. spikes, potentials, etc.) during the simulation. In addition, Populations can be assigned a display colour to facilitate visual identification.



Population Form Editor

Name :	EXC		
Size :	4000		
Cell Type :	IF_cond_alpha		
Parameters :	v_rest :	-60	mF
	cm :	0.2	nF
	tau_m :	20	ms
	tau_refrac :	5.	ms
	tau_syn_E :	5	ms

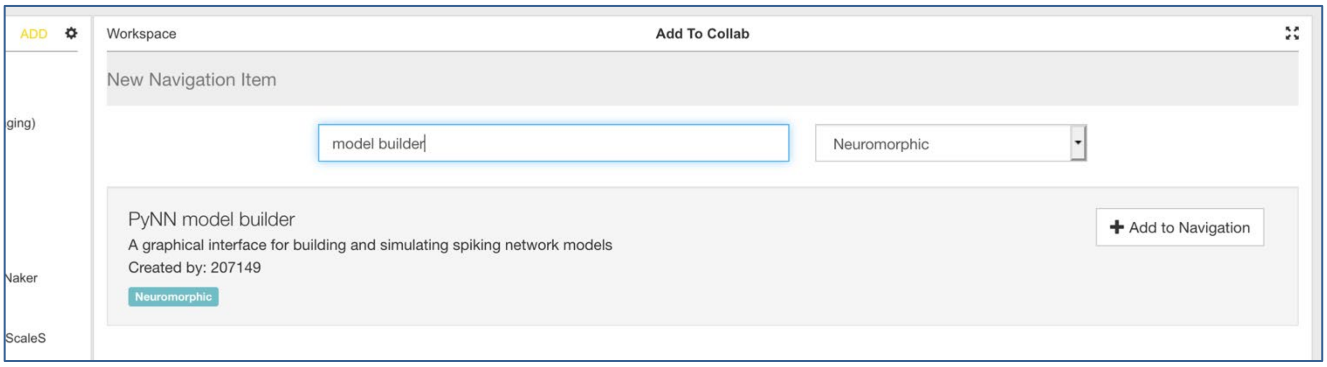
Figure 3: Screenshot of the Population editor

2.2.2 Projections

In order to create a connection between two Populations, the user has to click on the (pre-synaptic) source Population and drag it to the (post-synaptic) target Population. An arrow represents this Projection. Settings for a Projection are similar to those of the Population: by clicking with the right mouse button and selecting "Configure Projection", a form is displayed. In addition to the Projection name, the type of the connection is set by a drop-down menu (offering a choice of static or plastic synapses). The parameters (synaptic weight and delay) are set by hand or by using the input's spin box. The connection method is selected and all associated settings are displayed as spin boxes. A Projection from a Population to itself is possible from the options available by clicking on the right mouse button (Create Self Projection).

2.2.3 Installing the app in a Collab

In future, the app will be installed by default in all collab workspaces created from the Neuromorphic Computing Platform home collab. To install the app in an existing collab, the user can click on the “ADD” button, then search for “model builder” and click “Add to Navigation”.



PyNN model builder

A graphical interface for building and simulating spiking network models

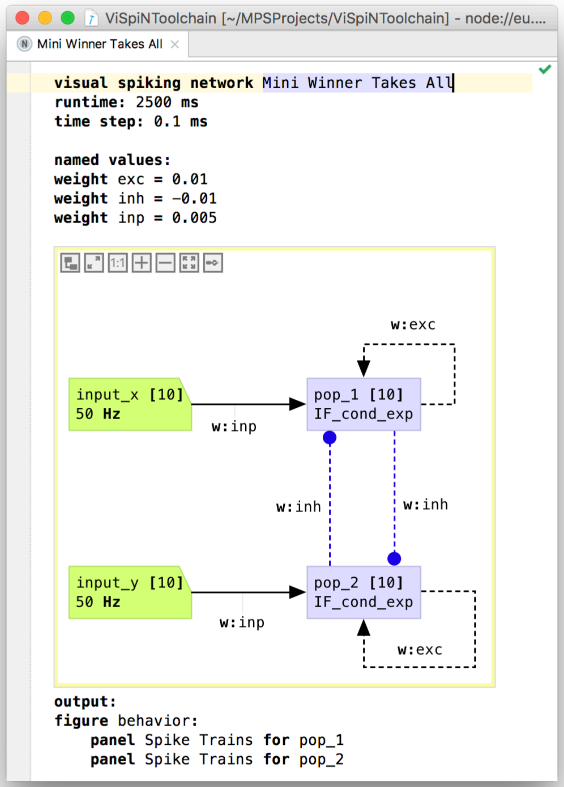
Created by: 207149

Figure 4: Screenshot demonstrating how to install the app in a collab workspace

3. Implementation

The PyNN GUI app is designed to be installed in any workspace (collab) in the HBP Collaboratory [Ref 2]. It is compatible with any web browser that supports the Collaboratory. If a compute quota for one or more of the Neuromorphic Computing Platform is associated with the collab, the user can submit jobs. Otherwise, the app can still be used, but only the script download will be possible. The source code is available online on GitHub [Ref 3]. The software is based on the AngularJS framework [Ref 4], using the mxGraph [Ref. 5] JavaScript diagramming library, which was chosen to allow the creation of shapes, vertex and edges, useful for graphically modelling a neural network.

To support exploration of orthogonal, hierarchically composable language features, an offline integrated development environment (IDE) has been built in the Meta Programming System MPS [Ref 11], which promotes evolutionary development and refinement of domain-specific programming languages (DSLs), including visual languages.



```
visual spiking network Mini Winner Takes All
runtime: 2500 ms
time step: 0.1 ms

named values:
weight exc = 0.01
weight inh = -0.01
weight inp = 0.005

output:
figure behavior:
    panel Spike Trains for pop_1
    panel Spike Trains for pop_2
```

Figure 5: Small Winner-takes-all network using named connection parameters in the MPS IDE

This approach to DSL development is relevant for language features that are either not inherently visual or have no obvious representation in the traditional graphical models of neural networks. The example network model in Figure 5 uses named parameters (similar to constants in statically typed textual programming languages) to support consistent exploration of weights for the three different connection roles (self excitation, cross inhibition, noise input) in a small winner-takes-all architecture.

4. Outlook

This first release of the app is a prototype. The app will be substantially improved in the remaining months of the SGA2 grant. Features to be implemented next include:

- Setting model parameters from random distributions;
- Saving and loading network descriptions to/from collab storage;
- Visualisation of simulation results;
- More robust error checking, and warning of features that are not available for a particular simulation engine.
- Handling of hierarchical structures, types and scopes in the visual programming language underlying the graphical representation.

Users may request additional features (and report bugs) at:

https://github.com/HumanBrainProject/pynn_graphical_editor/issues

5. References

[Ref. 1] Davison et al. (2009) Front. Neuroinform. 2:11

[Ref. 2] https://collab.humanbrainproject.eu

[Ref. 3] https://github.com/jonathanduperrier/pynn_graphical_editor

[Ref. 4] https://angularjs.org/

[Ref. 5] https://github.com/jgraph/mxgraph

[Ref. 6] https://angular-ui.github.io/bootstrap/

[Ref. 7] Eppler et al. (2009) Front. Neuroinform. 2:12

[Ref. 8] https://www.hbpneuromorphic.eu/home.html

[Ref. 9] Vogels and Abbott (2005) Journal of Neuroscience 25: 10786-10795.

[Ref. 10] Brette et al. (2007) Journal of Computational Neuroscience 23: 349-398

[Ref. 11] https://www.jetbrains.com/mps/