D9.3.1: PyNN 0.9 on SpiNNaker

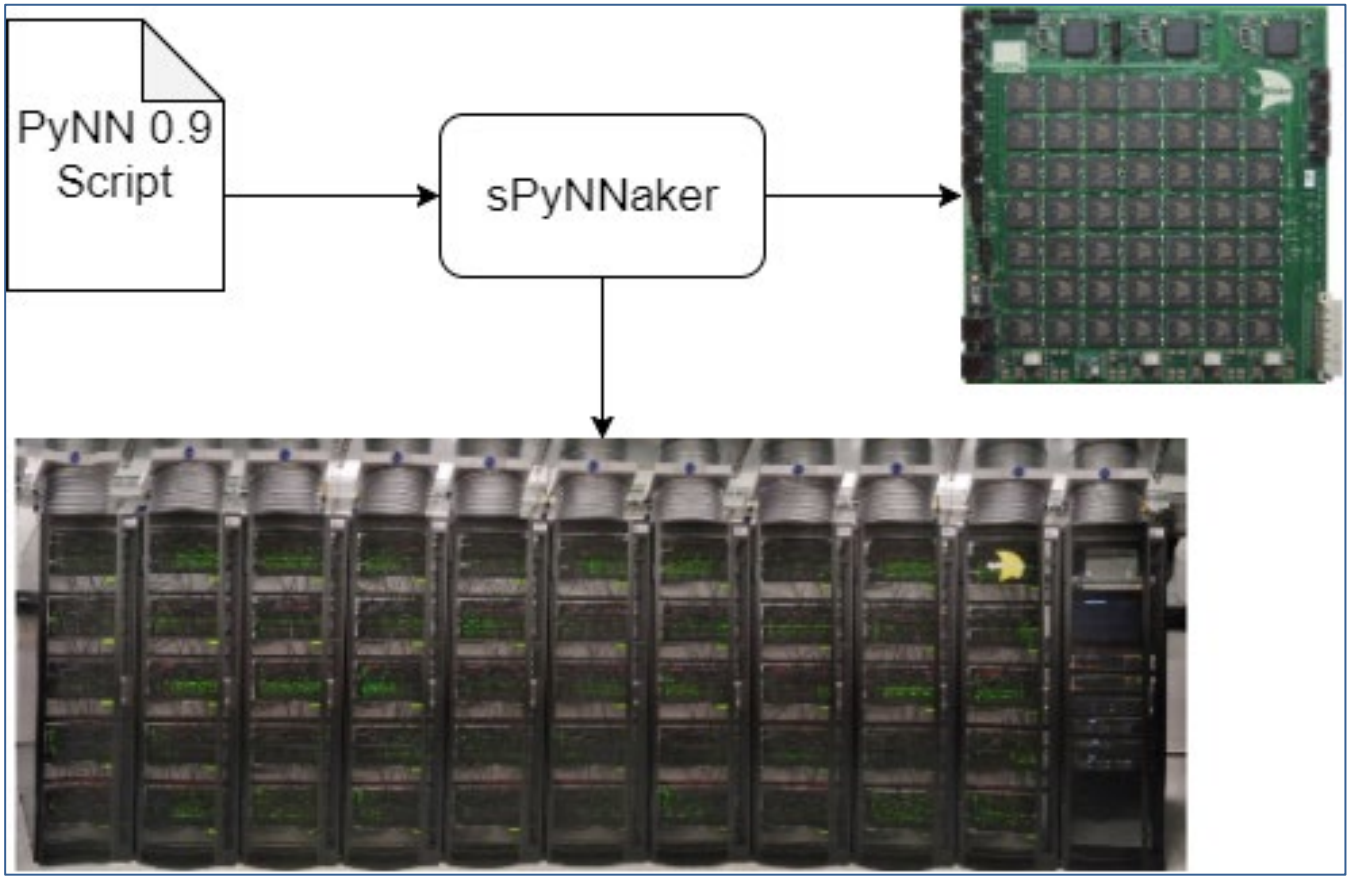

Figure 1: A PyNN 0.9 Script running on SpiNNaker.

The script is executed by the sPyNNaker software, which translates the script into data for executables to be run on the SpiNNaker machine.

Project Number:	785907	Project Title:	Human Brain Project SGA2
Document Title:	D9.3.1: PyNN 0.9 on SpiNNaker		
Document Filename:	D9.3.1 (D59.1 D92) SGA2 M12 ACCEPTED 190723		
Deliverable Number:	SGA2 D9.3.1 (D59.1, D92)		
Deliverable Type:	Other		
Work Packages:	WP9.3		
Dissemination Level:	PU = Public		
Planned Delivery Date:	SGA2 M12 / 31 March 2019		
Actual Delivery Date:	SGA2 M12 / 1 March 2019; ACCEPTED 23 Jul 2019		
Author(s):	Andrew ROWLEY, UMAN (P63)		
Compiled by:	Andrew ROWLEY, UMAN (P63)		
Contributor(s):	Andrew ROWLEY, UMAN (P63)		
SciTechCoord Review:	Marc MORGAN, EPFL (P1)		
Editorial Review:	Guy WILLIS, EPFL (P1)		
Description in GA:	As complete a PyNN 0.9 implementation on SpiNNaker as possible (Task T9.3.3)		
Abstract:	PyNN is the language used to describe spiking neural networks (SNNs) of point neurons which can then be simulated on various SNN simulators, including neuromorphic hardware. PyNN version 0.9 has been available since April 2017, but the SpiNNaker implementation requires special handling of some features, and so tends not to be released on the same schedule. The SpiNNaker backend has now been updated to comply with PyNN 0.9 as much as is possible, with the exception of a few less-used features that are more difficult to implement on neuromorphic hardware.		
Keywords:	PyNN, SpiNNaker, Neuromorphic Computing		
Target Users/Readers:	Computational Neuroscientists, Neuromorphic Computing Community, Platform Users		

Table of Contents

1. Introduction 4
2. PyNN 0.9 Implementation, as of 27 Feb 2019 4
3. Testing 5
4. Documentation 5
5. Conclusion 6

1. Introduction

PyNN[1] is a Python-based API used to describe and execute simulations of spiking neural networks. The API allows users to implement their network in a platform-agnostic manner, and then execute the same network on many platforms. This ability relies on the backend platforms implementing the same version of the API, as well as all the features of the API; PyNN does not require the latter in its entirety, with the only requirement being that at least two platforms implement any given feature for it to appear in the PyNN API.

A PyNN backend is available for the SpiNNaker platform in the form of the sPyNNaker software and associated libraries. This implementation is done at the API level as shown in Figure 1, meaning that there is little tie-in with the actual PyNN library itself. This is because some of the implementation details of the PyNN API can be implemented more efficiently, both in time and space, when it is known that SpiNNaker is the target platform. The sPyNNaker implementation does not fully implement all the features of the PyNN API, as some features are more difficult to implement on SpiNNaker, such as the full Hodgkin-Huxley model.

PyNN version 0.9 has been available since April 2017 but, until now, sPyNNaker had only implemented versions 0.7 and 0.8.

2. PyNN 0.9 Implementation, as of 27 Feb 2019

The software has now been updated to support version 0.9 of the PyNN API (PyNN 0.9.3, which is the latest release as of the date of this document). Although not all features of the API have been implemented, the most commonly used and important features have.

As of Feb 2019, the features that are missing include:

- PyNN Assembly objects[2]. These are groupings of Population, PopulationView and other Assembly objects into a single item with a common index. We have not had any users requesting these objects to date.
- Projection[3] between PopulationView objects. A PopulationView contains a subset of the neurons in a Population. Projections between PopulationView objects would allow the user to better specify connectivity between these subsets. However, it is possible for users to specify this connectivity using a FromListConnector, which is supported, or else by separating the Population objects into several sub-populations in advance, where this makes sense. Although there has been some interest in this feature, users have so far managed without it.
- TsodyksMarkramSynapse[4]. This is used to specify short-term plasticity. An initial implementation of this is available but this still requires testing.
- Various neuron models. Some of the neuron models are hard to implement within the limited code space of the SpiNNaker cores, and without floating point support. These include Integrate and Fire models with adaptation (with the exception of the Izhikevich model, which is implemented), and the Hodgkin-Huxley model.
- SpikeSourceInhGamma[5]. We have so far had no requests for this to be implemented.
- Current Sources[6]. All the neuron models support the use of i_offset, which is the equivalent of a DCSource. This can also be varied through the ability to call run a number of times.

[1] http://neuralensemble.org/docs/PyNN/
[2] http://neuralensemble.org/docs/PyNN/reference/populations.html#assemblies
[3] http://neuralensemble.org/docs/PyNN/reference/projections.html#projections
[4] http://neuralensemble.org/docs/PyNN/reference/plasticitymodels.html#short-term-plasticity-mechanisms
[5] http://neuralensemble.org/docs/PyNN/reference/neuronmodels.html#spike-sources
[6] http://neuralensemble.org/docs/PyNN/reference/electrodes.html#current-sources

Please note that, in spite of these features not being implemented, we nevertheless consider the implementation to date to be complete with respect to the API.

3. Testing

Figure 2: Output of Jenkins, the integration testing environment used to test the PyNN API compatibility.

A combination of unit tests and integration tests are set up to be performed on the software using the Jenkins CI continuous integration suite. These tests were used to verify the compatibility with the PyNN API, and continue to be executed when changes are made to ensure that those changes do not break the software or the compatibility with the PyNN API. The tests performed can be seen at https://github.com/SpiNNakerManchester/sPyNNaker8/tree/master/p8_integration_tests. An example of the output of these tests run on the Jenkins system is shown in Figure 2.

4. Documentation

The documentation for this software consists of many elements, including:

- The HBP neuromorphic guidebook[7]. This is online documentation that evolves with the software, and includes links to the PyNN API and example code.
- The sPyNNaker code-level documentation[8]. This details the modules available in the software, which includes PyNN extensions.
- The SpiNNaker software pages[9]. This includes installation guides and general usage instructions and limitations.
- The SpiNNaker Workshop slides and tutorials[10]. This includes training material with slides and lab manuals to help users learn to use PyNN with SpiNNaker.

5. Conclusion

The PyNN 0.9 API has been implemented on SpiNNaker. There are a few limitations to this interface, but they can be addressed through the ongoing maintenance of the software, as the need arises. The testing framework and documentation have been updated to match the implementation.

The changes made will make it easier for users to use SpiNNaker. With these changes, users now have the ability to reduce the data extracted from the machine through the implementation of the PopulationView object which enables selective recording, which will make the execution of scripts faster. The fact that the latest API has been implemented means that users can execute the same scripts on the SpiNNaker platform as they can on other platforms that support the API, with only minor changes required should their scripts happen to rely on the unimplemented features.

[7] https://collab.humanbrainproject.eu/#/collab/51/nav/1069
[8] https://spynnaker8.readthedocs.io/en/latest/
[9] http://spinnakermanchester.github.io/
[10] http://spinnakermanchester.github.io/workshops/eighth.html