Building Neuromorphic Embodied Cell Assembly Agents that Learn

Christian Huyck[1] and Ian Mitchell[1]

Middlesex University, London, UK
c.huyck & i.mitchell@mdx.ac.uk

Abstract
Neuromorphic embodied cell assembly agents that learn are one application being developed for the Human Brain Project (HBP). The HBP will build tools, available for all researchers, for building brain simulations. Existing simulated neural agents will be translated to the platforms provided by the HBP; these agents will then run on neuromorphic chips instead of a von Neumann based computer. The initial translation is relatively straight forward but non-trivial software engineering, as any computational system can be programmed from simulated neurons by setting the neural and synaptic topology. By running variants of these agents on different platforms and with the different simulated neural models, implicit assumptions in these models can be revealed. Once these Cell Assembly agents have been translated and embodied in a virtual environment, they will be extended to learn more effectively. These agents will also be able to make use of the neural hardware to simulate in real time many more neurons, providing a platform for exploration of more complex simulated neural systems.

Keywords: Cell Assembly, PyNN, Embodied Agent, and Neural Simulation

1 Introduction

The Cell Assembly (CA), initially proposed in 1949 [10, 14], is a key component in brain function linking neural behaviour to psychological behaviour. A CA is a set of neurons that are able to support firing amongst themselves; when activated the firing CA becomes a short-term memory that persists while many of the neurons continue to fire at an elevated rate. Each primitive concept, for instance *dog*, has a CA that represents it.

CAs are central to a series of video-game agents, the Cell Assembly robots (CABots) [12], implemented entirely in simulated neurons. Each CABot is an interactive agent in a virtual environment that learns, explores, plans and builds a cognitive map from sensory information. The agent can be directed by simple commands and can learn simple aspects of the environment. The third version, CABot3 (see Figure 1), is composed of four major sub-systems, and a simple control system to manage some inter-system behaviour. The major systems, each consisting of hundreds of CAs, are:

- Natural Language Processing (NLP): process user input for goal setting;
- Visual: identification and location of objects;
- Plan: goal-oriented, communication and action; and
- Spatial Cognitive Map: long-term mapping of the environment.

When commanded, the agent explores the environment, building the spatial cognitive map from information gathered via the visual field. The planning was completed using a Maes network [18]. The environment has four rooms and each room has a different object. Testing the system relies on the agent finding an object from its current location.

The agent is managed by sub-networks composed of simulated neurons, a technical possibility thought too complex in 1988 [22]. There are some compromises made on biological plausibility, but the agent is composed of and controlled entirely by simulated neurons.

Recently work has begun porting CABots from their Java based neural simulation, via PyNN [7], to neuromorphic chips [15]. This is one small component of the Human Brain Project [1] (HBP). Porting the existing CABots to the chips is a non-trivial engineering task. Once done, the agent will be extended to learn visual objects, learn plans, and will be tested on new cognitive modeling tasks.

This paper presents an initial prototype that views the environment and implements simple plans. The prototype agent has vision and planning. The work has so far focused on vision and simple plans in PyNN.

The longer term project is an effort to reproduce CABot3 [12] using PyNN. This project aims to embed the code on neuromorphic chips and therefore perform more tasks in richer environments aiming to provide further understanding of general neural cognitive architectures. This is an application making use of several components of the HBP.

2 NEAL, an Application in the Human Brain Project

The authors have been given a grant as part of the HBP to build Neuromorphic Embodied Agents that Learn (NEAL). The HBP, explained further below, is broken into several subprojects. NEAL is a part of the Applications Subproject, but is also aligned with the Neuromorphic Computing Subproject.

While NEAL is part of the Applications Subproject, and to some extent the Neuromorphic Computing Subproject, it is linked to other Subprojects. It takes advantage of the Neurorobotics, High Performance Computing, Brain Simulation and Cognitive Modelling Subprojects.

The HBP is a European Union sponsored project to further understanding of mammalian brains in general, and human brains in particular. As brains are complex and poorly understood, the project aims to build tools and platforms to support scientific exploration of them. The HBP and the American Brain Research through Advancing Innovative Neurotechnologies initiative are complementary projects that hopefully will lead to a significant advance in both understanding of brain function, and improved computer systems.

The HBP platforms include a neuromorphic computing platform, a Neuroinformatics Platform, a Brain Simulation Platform, a High Performance Computing Platform, a Medical Informatics Platform, and a Neurorobotics Platform. These computational platforms will all be developed for scientists in and outside the HBP to use. The Neuroinformatics Platform will contain neuroinformatics data such as ontologies, brain atlases and neural connectivity of both

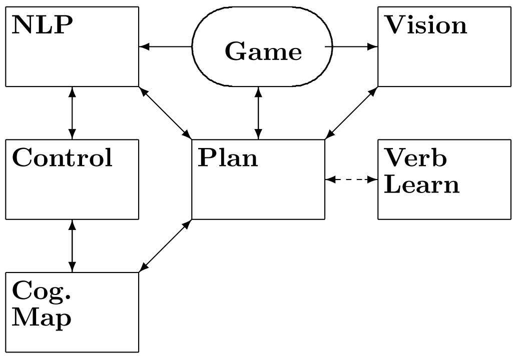

Figure 1: Gross Topology of CABot3. Boxes represent subysystems of subnets. The oval represents the environment.

human and mouse brains. The Brain Simulation Platform will use parallelized versions of simulators such as NEST to support large scale simulations of neural systems; this extends the Blue Brain Project [19]. While the Brain Simulation Platform focuses on software, the High Performance Computing Platform focuses more on hardware for the same purpose. The Medical Informatics Platform focuses on medical data for problems such as Alzheimer's disease.

While the authors' work will make use of these other platforms, it is most closely involved with the Neuromorphic Computing and Neurorobotics Platforms. The Neurorobotics Platform will develop virtual environments and virtual robots, that are linked to neural simulations. This is quite similar to the games environments used in the authors' earlier CABot work [12].

The Neuromorphic Computing Platform makes use of and provides access to two types of neuromorphic chips: SpiNNaker [8] and HICANN [3]. These chips are designed to simulate large numbers of neurons rapidly; they are radically different than typical von Neumann architectures.

The HBP is also broken into 13 Subprojects, and those most closely related to this work are described next. Several of the Subprojects are closely aligned to the platforms. These include the Neuromorphic Computing Platform Subproject, the Neurorobotics Platform Subproject, the Brain Simulation Platform Subproject, and the High Performance Computing Platform Subproject. Another is the Cognitive Architectures Subproject. The goal is to select studied cognitive tasks, simulate them in an artificial neural network, and align activity to known brain activation patterns.

3 CABot Translation

In earlier work, the authors and colleagues developed an agent in a virtual environment [12]. The virtually embodied agent used plans, took natural language commands, viewed the environment, and moved around the environment. Figure 1 shows the gross structure of the sub-systems, and how they connect.

The CABot agents were implemented entirely in simulated fatiguing leaky integrate and fire (FLIF) neurons. These neurons were implemented in Java and did all of the agents' processing. The environment was a simple three dimensional virtual space with objects implemented in a shareware games engine, Crystal Space [6]. The largest agent had approximately 100,000 neurons with several million synapses, and ran in roughly real time along with the game on a standard PC.

The first step in the NEAL project is to translate CABot agents from FLIF neurons simulated in Java, to a standard neural class in PyNN. This can then be run on the neuromorphic chips. PyNN (see section 3.1) is a Python package for simulator-independent specification of neuronal network models.

The Java CABot3 system was broken into 36 sub-networks with the systems described in Figure 1 consisting of multiple sub-networks. The sub-networks provide some modularity, aiding in software engineering, and in a few cases (e.g. primary and secondary visual cortices) link directly to cortical areas.

While the original Java neuron is a model of a biological neuron, albeit a point model, the network's behaviour does not readily translate to PyNN. For instance, the Java CABot models used a 10 ms. step, and when a neuron fires, neural activation is propagated to synaptically adjacent neurons in the next time step. PyNN and the underlying simulators do not allow this. So, a complete respecification of the neurons and synapses is needed.

3.1 PyNN

PyNN [7] is a type of middle ware. A neural simulation is described in PyNN. The simulation is then run through PyNN and passed onto a neural simulator such as NEST [9] or NEURON [5]. The neural simulator interacts with the original Python specification to manage interactions.

The HBP takes advantage of PyNN's ability to use multiple simulators for the backend with roughly the same results. While new neural models can be developed, PyNN specifies standard models and most backends can run these. Other backends can be developed for PyNN and simulators for the two neuromorphic chips have been developed. The PyNN description can run in these and this should mimic the behaviour of the hardware. Eventually, the PyNN description will be able to interact directly with the neuromorphic hardware. Moreover, this is not limited to neuromorphic hardware but can also be run on high performance computers. There may be differences in the three types of hardware, and indeed in the neural simulators, and these differences need to be understood.

PyNN specifies standard neural and synaptic models. These include exponential integrate and fire models [2], Izhikevich neurons [16], and Hodgkin Huxley neurons [11]. Most backends support most of these standard types of neurons.

PyNN supports current injection as this is often used in neural simulations. Current is injected into biological neurons with electrodes causing them to fire; simulations try to reproduce this behaviour. In this project, current injection is used to make sensory neurons fire so that the neural system can interact with the environment. The engineering in NEAL is about translating what worked in the Java FLIF model to PyNN, and then on to the other neural platforms.

3.2 Translation

The translation, like much of the original CABot work, is relatively straightforward software engineering. CABot3 showed that given enough neurons, anything can be programmed; neurons are Turing complete [4]. The work was to develop neural systems for vision, planning, and NL parsing. As these FLIF models do not directly translate to PyNN, new systems are being developed.

The vision systems, both the FLIF and PyNN versions, are based on known neural functioning. The retina functions as a set of on-off and off-on centre surround receptive fields. Simulated neurons mimic this behaviour, and are programmed by setting their synaptic weights and connectivity. A 3x3 on-centre off-surround neuron has an excitatory synapse from the centre, and inhibitory synapses from the surround. The simulated primary visual cortex merely gets inputs from these retinal detectors. The new version will have similar connectivity, but the synaptic weights are different, and individual neurons behave slightly differently. The PyNN version will have different numbers of neurons for some functions, and it is likely that some functions will have different numbers of sub-networks; the sub-net concept is somewhat blurred in PyNN, because it is not a predefined object in PyNN, while it is an object in the Java simulations.

The planning system is a neural implementation of a Maes net [18]. Facts, modules, goals and actions are all stored as simple CAs in separate sub-nets. The sub-nets are connected so that neural activation spreads implementing the appropriate plans. The CAs in these sub-nets are simple oscillators. For example, the CA for the *pyramid* fact is a set of neurons, so CAs implement facts in the Maes net. When the fact is on in the FLIF system, half of the neurons fire in one 10 ms. cycle, and these turn the other half on in the next 10 ms. cycle, while the initial neurons are off. This allows the fact to persist indefinitely without the neurons accumulating fatigue. This is a poor CA, but it enables short term memory that can be explicitly turned on and off.

There needs to be input from the environment to the system, and in turn there needs to be output from the agent to the environment. Input is in the form of a binary bitmap of the environment (the agent's camera), a bump sensor, and user commands in the form of typed words. Output is from the action sub-net from the planning system. If one of the four actions, turn left, turn right, move forward, or move backward, is on, a symbolic command is passed to the game agent. Here, 'on' means that a sufficient number of neurons in the CA for that action are firing. The system translates the subsymbolic firing behaviour of these neurons to the symbolic actions taken by the agent.

The first working PyNN system will consist of just vision and a simple version of planning. This should be able to run using the NEST simulator on a PC. While NEST runs much more slowly on a PC than the original Java code, this should still work effectively in simulation, as interaction with the game does not need to be real time; it will be slow, but verifiably correct. This simple agent will also enable porting to the SpiNNaker simulator, and SpiNNaker. In this case input will be from vision. The goal will be explicitly set, since it cannot be set by user language commands, because NLP remains to be implemented.

Translation of the remaining portions of CABot3 involve language, cognitive mapping, learning verbs, visual textures, and making it all work together. This work is not about solving new theoretical problems. The Java CABot3 had binding, and cognitive models for parsing, and CABot2 had verb learning. These will be reimplemented, but it will make no conceptual breakthrough, merely show that the basic idea is flexible across different neural models. It will, however, provide a system that can be extended and modified.

3.3 Early Protoype Status

There are preliminary engineering results from the translation. The initial goal is to make an agent with a visual system, and simple planning. This will then be translated to the SpiNNaker simulator and then onto the hardware. Code can be found on http://www.cwa.mdx.ac.uk/NEAL/code/bica.html.

The PyNN system currently uses the standard Izhikevich neurons. These are implemented in the current version of SpiNNaker, and they should be implemented in HICANN.

As the 10 ms. model from the Java implementation will not work, and there are some assumptions in SpiNNaker around a 1 ms. model, the PyNN simulations use a 1 ms. model. That is, input is considered, and activation spreads in discrete 1 ms. cycles. This in itself requires entirely new engineering. The connections will have to be respecified, and it is likely that the fine grained neural topology will also need to be respecified.

Progress has been made with the Izhikevich neurons, all of the on-centre off-surround and off-centre on-surround detectors have been implemented. For example, the 3x3 on-centre off-surround detector has 6 different behaviours. It fires maximally when the centre input is on, and the surrounding eight neurons are off. The firing rate and onset decrease as surrounding inputs are added. With five to eight surrounding inputs on there is no firing. The 3x3, 6x6, and 9x9 detectors are all implemented. Moreover, all six detectors have been integrated with images, with current injection turning on the input neurons.

Much of the FLIF version of the CABots used oscillators to compensate for fatigue. These are poor CAs from a psychological perspective, because, unlike short term memory, they persist forever. However, they are easy to work with as programming primitives. One set of, typically five, neurons would fire in one cycle, a second in the next, then the first again and so forth. Once the oscillator was turned on, it persisted until it was turned off. This has been implemented in PyNN, again with Izhikevich neurons. The first few spikes are more rapid, but the oscillator then persists indefinitely with each set, again of five, firing every 10 ms., or every 10 cycles; this is twice as fast in simulated time as the FLIF version, though the Java version takes two cycles.

The planning subsystem requires facts, goals and output actions based on Maes Networks [18]. Neurons are based on the Izhikevich model, and the CAs are oscillators of populations of these neurons. Thus far, four primitives have been formed and completed in PyNN. Each primitive has a goal CA, a module CA and an action CA. The goal CA is stimulated by injected current and represents a command issued by the user, which will eventually be provided by the language subsystem. When the goal CA is active, it stimulates the appropriate module CA. When the module CA is active, the appropriate action CA is activated. This occurs in a feed-forward manner via excitatory connections between each CA. When the action CA becomes active, the goal is fulfilled and there is a feedback process via inhibitory connections between each CA; this turns the goal and module CAs off. By injecting current in 100ms. cycles each of the four primitives activated the appropriate action.

3.4 Completing CABot1 and CABot3

Call the current system under translation CABot1, as it is similar to the first CABot. Some basic elements have already been translated to PyNN. All six retinal detectors have been translated and have been linked together so that 2500 of each are stimulated by the 50x50 bitmap input. Next comes the translation of the primary visual cortex (V1), edge and angle detectors. CABot3 had texture detection, but that will not be needed for this first version, because only one of each type of object in the environment is needed for the task. V1 needs to detect four orientations of edge, and four sorts of angles. The visual system then needs to detect two types of object of varying size and location in the visual field; these are up facing and down facing tetrahedrons (pyramids and stalactites). Finally, these objects need to activate facts for planning. Facts are oscillators, but V1 and object detection will require more work.

The simple plans will respond to primitive goals, compound goals, and two context sensitive goals. The primitive goals are turn left or right, and move forward or backward, and they have already been implemented. Compound goals are move left and right, a combination of turn left or right and move forward. The context sensitive goal is turn toward the pyramid, or stalactite. This requires a fact activated from the environment.

This will also require interaction between the virtual environment and the neural simulator.

PyNN provides a mechanism for the simulator to 'call-back' a function in Python. Input from the environment to the neural simulator is by clamping input neurons on. Output from the simulator is done by determining how particular 'output' neurons spike to determine the correct commands to send to the agent's body in the environment.

That will complete CABot1, but it then needs to be tested on the SpiNNker simulator, then the SpiNNaker chip. It is not clear how the environment will be supported with the chip.

Translation of CABot3 will require more re-engineering, but there is a potential problem with parsing. Context free parsing requires binding of some sort. In the Java version, binding was done by short-term potentiation. Our goal is to use the Tsodykys Markram synapses provided as standard models in PyNN for this.

Other learning mechanisms are required for cognitive map building, and for learning verb associations. Standard long-term potentiation rules should be sufficient.

Finally, an agent needs to work in HICANN in addition to SpiNNaker. Hopefully at this stage, the HICANN simulator and hardware will function seamlessly with the PyNN code, but it may not because HICANN may have made different assumptions. It is possible that it will all need to be re-engineered with a different neural model, as HICANN may not support the Izhikevich model.

This engineering and proposed engineering is interesting because different assumptions are implicit to each backend, and indeed to the initial Java CABots. The engineering reveals these assumptions, leading to a better understanding of large scale neurodynamics, and a better understanding of how to program these neural systems.

4 Extending CABot3

While the engineering work of the CABot translation is useful and interesting, the second phase of the project will lead to significant advancements. The CABot project showed that neural systems are Turing complete, but the real benefit of neural systems over standard symbolic computation is that they can learn. The CABot agents have some simple learning, but running agents on neuromorphic chips will support larger numbers of neurons behaving in real time. These will be able to learn in a psychologically and neurally plausible manner. The system will learn a range of visual categories, and learn them as more psychologically plausible CAs. The system will also learn new plans.

Since developing CABot, the authors have focused on learning. By modifying the FLIF model so that neurons spontaneously fire when hypo-fatigued, a system that learns categories has been developed [13]. Biological neurons spike without input, and hypo-fatigue is one mechanism that could drive this behaviour. By integrating this with earlier visual category learning [17], making use of more neurons, and further exploring dynamics, a wide range of visual categories from the environment can be learned.

Moreover, one flaw with existing CA simulations, is that the CAs do not persist for psychologically realistic durations. Ignited CAs are CAs with enough neurons firing to support persistent firing throughout the CA. They are the neural implemantation of psychological short-term memories, and the duration of the persistence depends on many factors. For example, they persist longer when more activated, and they can be reactivated. Some unpublished work with small-world topologies has led to more realistic persistence, and this may be a solution or part of the solution for more psychologically realistic CAs. Another avenue to explore is to use short term potentiation, to change persistence.

A first step at learning new plans will be to improve existing plans and plan elements via reinforcement learning. This is similar to verb learning done in CABot2 making use of reinforcement learning. A more significant advance may be made by using improved CAs to learn new plans. Here instead of oscillators, more sophisticated CAs will be used giving the system the ability to use hierarchical CAs. This will support the development of new planning CAs, which can then be combined to make new plans.

New cognitive models will also be developed. A simple model for learning categories based on Shepard's experiments [21] will link neural and cognitive category learning. The system will also be given a Wisconsin Card Sorting Task [20], and should perform this task reproducing human results. This will work through the virtual environment so the system will have to actually view the cards.

The Neurorobotics Platform simulator should replace Crystal Space. The sooner this happens the better, as it should be better supported by the Neuromorphic Platforms. Finally, it is hoped that the system will be ported to the High Performance Computing Platform. Hopefully, the PyNN implementation and Neurorobotics Platform will make this easy.

5 Conclusion

The main aim of NEAL is to build an agent, embodied in a virtual environment, based solely on simulated neurons running on neuromorphic chips. This will make use of many resources provided by the Human Brain Project, whose main goal is to build tools for building brain simulations.

The first phase of this project is about porting the existing CABot agents from a Java based neural simulation, to PyNN, and then onto neuromorphic chips. This will require re-engineering of the neural network, but should be relatively straight forward. Special care will need to be taken with interactions between the simulated agent and the virtual environment. Good initial progress has been made toward this goal.

The second phase of the project will involve extending the agent, so that it can better learn CAs. These CAs will be more psychologically plausible and will make the agent more flexible. CAs will be learned for visual categories so that a CA will persistently fire when an item of a particular category is presented. New plans will be learned.

This is another step along a path using a unified approach to develop embodied agents. All computation will be done in neurons; communication between neurons will be via spikes and different neural sub+ystems will communicate by this mechanism. The system will constantly learn. As part of the HBP it becomes a staging post for other work. Others can use this system as a starting point. Existing subsystems can be modified and new subsystems added. These can then be used to perform multiple tasks, including cognitive models, and embodied agents that learn their environment and communicate with a human via natural language.

Acknowledgements:
This work was supported by the Human Brain Project Grant 604102, Neuromorphic Embodied Agents that Learn.

References

[1] The human brain project, 2014.

[2] R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94:3637–3642, 2005.

[3] D. Brüderle, M. Petrovici, B. Vogginger, M. Ehrlich, T. Pfeil, S. Millner, A. Grübl, K. Wendt, E. Müller, M. Schwartz, et al. A comprehensive workflow for general-purpose neural modeling

with highly configurable neuromorphic hardware systems. *Biological cybernetics*, 104(4-5):263–296, 2011.

[4] E. Byrne and C. Huyck. Processing with cell assemblies. *Neurocomputing*, 74:76–83, 2010.

[5] N. Carnevale and M. Hines. *The NEURON Book*. Cambridge University Press, 2006.

[6] Crystal Space. $http://www.crystalspace3d.org/main/main\_page$, 2008.

[7] A. Davison, D. Brüderle, J. Eppler, E. Muller, D. Pecevski, L. Perrinet, and P. Yqer. PyNN: a common interface for neuronal network simulators. *Frontiers in neuroinformatics*, 2, 2008.

[8] S. Furber, D. Lester, L. Plana, J. Garside, E. Painkras, S. Temple, and A. Brown. Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467, 2013.

[9] M. Gewaltig and M. Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[10] D. Hebb. *The Organization of Behavior.* John Wiley and Sons, 1949.

[11] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[12] C. Huyck, R. Belavkin, F. Jamshed, K. Nadh, P. Passmore, E. Byrne, and D.Diaper. CABot3: A simulated neural games agent. In *7th Intl Workshop on Neural-Symbolic Learning and Reasoning, NeSYS'11*, pages 500–544, 2011.

[13] C. Huyck and I. Mitchell. Post and pre-compensatory Hebbian learning for categorisation. *Computational Neurodynamics*, 8:4:299–311, 2014.

[14] C. Huyck and P. Passmore. A review of cell assemblies. *Biological Cybernetics*, 107:3:263–288, 2013.

[15] G. Indiveri, B. Linares-Barranco, T. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S. Liu, P. Dudek, P. Häfliger, S. Renaud, et al. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5, 2011.

[16] E. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15:5:1063–1070., 2004.

[17] F. Jamshed and C. Huyck. Grounding symbols: Learning 2-d shapes using cell assemblies that emerge from fLIF neurons. In *Proceedings of the AISB*, 2010.

[18] P. Maes. How to do the right thing. *Connection Science*, 1:3:291–323, 1989.

[19] H. Markram. The blue brain project. *Nature Reviews Neuroscience*, 7:153–160, 2006.

[20] O. Monchi, M. Petrides, V. Petre, K. Worsley, and A. Dagher. Wisconsin card sorting revisited: Distinct neural circuits participating in different stages of the task identified by event-related functional magnetic resonance imaging. *Journal of Neuroscience*, 21(19):7733–7741, 2001.

[21] R. Shepard, C. Hovland, and H. Jenkins. Learninng and memorization of classifications. *Psychological Monographs*, 75:517:1–42, 1961.

[22] P. Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1:1–22, 1988.