Embodied Modeling With Spiking Neural Networks For Neuromorphic Hardware: A Simulation Study

C. M. Thibeault[1,2,3,*], F. C. Harris Jr.[2,3], and N. Srinivasa[1]

[1]*Center for Neural and Emergent Systems, Information and Systems Sciences Laboratory, HRL Laboratories LLC., Malibu, CA.*
[2]*Department of Electrical and Biomedical Engineering, University of Nevada, Reno. Reno, NV.*
[3]*Department of Computer Science, University of Nevada, Reno. Reno, NV.*

Abstract

Adding value to action-selection through reinforcement-learning provides a mechanism for modifying future decisions. This behavioral-level modulation is vital for performing in complex and dynamic environments. In this paper we focus on a class of biologically inspired feed-forward spiking neural networks capable of action-selection via reinforcement-learning. The networks are embodied in a minimal virtual agent and their ability to learn two simple games through reinforcement and punishment is explored. There is no bias or understanding of the task inherent to the network and all of the dynamics emerge based on environmental interactions. Value of an action takes the form of reinforcement and punishment signals. One novel aspect of these networks is that they obey the constraints of neuromorphic hardware currently being developed, including the DARPA SyNAPSE neuromorphic chips for very low power spiking model implementations. The simulation results demonstrate the performance of these models for a variant of classic pong as well as a first-person shooter. Embodying models like these in games creates virtual environments with varying levels of detail that are ideal for testing spiking neural networks. In addition, the results suggest that these models could serve as building blocks for the control of more complex robotic systems that are embodied in both virtual and real environments.

Keywords: *Embodied modeling, Neurorobotics, Spiking Neural Networks, Action-Selection, Reinforcement-Learning, Neuromorphic hardware*

1 Introduction

The combination of action-selection and reinforcement-learning in biological entities is essential for successfully adapting and thriving in complex environments. This is also important for the effective operation of intelligent agents. However, strategies for embedding artificial intelligence have resulted in agents with limited demonstrable emergent properties. Because of this, it is still unreasonable to deploy a neurorobotic entity and expect it to learn from and perform in its environment the same way biological entities can. Similarly, neural models require complex and varied input signals in order to accurately replicate the activity observed experimentally. One strategy for creating this complex stimuli is through immersing a model in a real or virtual environment capable of providing the feedback necessary for the model to extract value and interact appropriately. These are part of the motivations for the DARPA SyNAPSE program [1, 2]. Through the creation of low-power neuromorphic architectures both suitable for efficient remote operation and capable of replicating many of the biologically salient features of neural systems, the program can reduce the technological and theoretical barriers of embodied modeling.

Embodied modeling can be described as the coupling of computational biology and engineering. This can obviously be accomplished in a many different ways but games are one of the most beneficial for exploring those. The varying levels of complexity combined with quantifiable performance result in environments appropriate for testing many different levels of biological fidelity. Two of the most basic aspects of playing those games are action-selection and reinforcement learning. These are important for making decisions based on past experience to achieve the desired outcomes.

Action selection is the appropriate negotiation of competing signals. In the mammalian nervous system the complex circuitry of the Basal Ganglia (BG) is active in gating the information flow in the frontal cortex by appropriately selecting between input signals. This selection mechanism can affect simple action all the way up to complex behaviors and cognitive processing [3]. Although overly simplified, it can be helpful to relate the BG to a circuit multiplexer, actively connecting inputs to outputs based on the current system state.

Reinforcement or reward learning (RL) is the reinforcement of actions or decisions that maximizes the positive outcome of those choices. This is similar to instrumental conditioning where stimulus-response trials result in reinforcement of responses that are rewarded and attenuation of those that are not [4]. Reinforcement-learning in a neural network is an ideal alternative to supervised learning algorithms. Where supervised learning requires an intelligent teaching signal that must have a detailed understanding of the task, reinforcement learning can develop independent of the task without any prior knowledge. Only the quality of the output signal in response to the input signal and current contextual state of the network is needed.

In this work we focus on a class of small biologically inspired feed-forward spiking networks capable of action-selection and reinforcement-learning while immersed in a vir-

*Corresponding Author. cmthibeault@hrl.com

Table 1: Global model parameters.

Parameter	Value	
C_m	1.	(pF)
E_{exc}	0.	(mV)
E_{inh}	$-80.$	(mV)
V_{rest}	0.	(mV)
A_+	0.025	
A_-	0.026	
τ_+	20.	(ms)
τ_-	20.	(ms)

tual environment. These are suitable for realization on the neuromorphic hardware developed under the SyNAPSE project and provides a theoretical framework for testing future novel reinforcement-learning algorithms. The networks are embodied in a minimal virtual agent and the ability to learn a simple ping-pong game through reinforcement and punishment is explored. There is no bias or understanding of the task inherent to the network and all of the dynamics emerge based on interactions with the environment. Value of an action takes the form of simple reinforcement and punishment signals. This concept is then extended by exploring how these can be combined to perform more complex actions. Towards this goal, a first-person shooter was developed. A model combining multiple RL networks was then constructed and trained to target and shoot the most appropriate enemy.

In addition to supporting hardware validation, the resulting models are ideal for simple robotic embodiments and are capable of demonstrating action-selection via reinforcement-learning. Similarly, the two games developed for testing these networks illustrate the utility of embodied modeling in competitive environments.

There have been a number of research efforts aimed at utilizing games to explore action-selection and reward learning. For instance, Wiles *et al.* [5] developed a spiking neural model to control a rat animate performing phototaxis. The network was constructed to perform the task similar to a Braitenberg vehicle. Burgsteiner *et al.* [6] created a liquid state machine using a recurrent network with fixed internal synapses and plastic output synapses that learned a similar task.

The model of Arena *et al.* [7] consisted of three layers of Izhikevich neurons to control a virtual robot with several sensory modalities. The networks were constructed with an initial understanding of how to process low-level sensor input such as proximity and contact sensors as well as visual cues. These were used to direct the robot through the environment. Simultaneously, the network learns to perform this navigation using a range-finding sensors. The inherent low-level sensors basically train the network on how to respond to the high-level sensors.

Florian *et al.* [8] evolved a fully recurrent spiking neural network to control a simple virtual agent to seek out, push and the release balls in its environment. An evolutionary algorithm was used to calculate the synaptic weights of the network to accomplish the task.

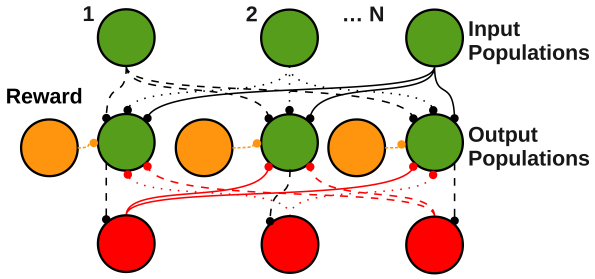

Figure 1: Lateral-inhibition network.

Barr *et al.* [9] implemented a mode of the basal ganglia on a neural processor array. Although not directly demonstrated in the hardware presentation the original software neural model was capable of performing action selection. However, there are no inherent mechanisms for reinforcement-learning and the micro-channels of the basal ganglia were predefined by the network.

Merolla *et al.* [1] presented a neuromorphic processor capable of playing a game of pong against a human opponent. This description was later extended by Arthur *et al.* [10]. The network was constructed off-line and once programmed on the hardware remained static. In that, a neural network, consisting of 224 neurons, that could also play a pong style game was created. The network was constructed off-line and was demonstrated on a neuromorphic processing core. Training involved teaching the network to predict different patterns of motion by the puck. Rather than simply tracking it, like the networks here, the model would plan where the paddle must be placed. The resulting networks however, are specialized for that task and cannot adapt to changing environments once embodied in hardware.

2 Design and Methods

2.1 Neuron model

The neural model supported by the initial SyNAPSE hardware is the Leaky-Integrate and Fire (LIF) neuron. The LIF model is defined by

$$C_m \frac{dV}{dt} = -g_{leak}(V - E_{rest}) + I. \qquad (1)$$

where

C_m	is the membrane capacitance.
I	is the sum of external and synaptic currents.
g_{leak}	conductance of the leak channels.
E_{leak}	is the reversal potential for the background leak currents.

As the current input into the model neuron is increased the membrane voltage will proportionally increase until a threshold is reached. At this point an action potential is fired and the membrane voltage is reset to the resting value. The neuron is placed in a refractory period for 2 milliseconds where no changes in the membrane voltage are allowed. If the current is removed before reaching the threshold the voltage will decay to E_{rest}. The LIF model is one of the least

computationally intensive neural models but is still capable of replicating many aspects of neural activity [11].

The connections between neurons are modeled by conductance-based synapses. The general form of that influence is defined as

$$I_{syn} = g_{max} \cdot g_{eff} \cdot (V - E_{syn}). \quad (2)$$

where

g_{max}	is the maximum conductance for the class of synapse.
g_{eff}	is the current synaptic efficacy between $[0, 1]$.
E_{syn}	is the reversal potential for that particular class of synapse.

Although the synapses are conductance based the buffering and reuptake of neurotransmitter is treated as a pulse event lasting one time step. In that way it is similar to current based synapse. For numerical integration An Euler method is used with time step $\tau = 1ms$.

Learning at the synaptic level is achieved through the spike-timing dependent plasticity rules defined by Song *et al.* [12]:

$$\dot{g}_{eff} = P_{ij} X_i(t) - D_{ij} X_j(t - \Delta_{ij}) \quad (3)$$

$$\dot{P}_{ij} = -\frac{P_{ij}}{\tau_+} + A_+ X_j(t - \Delta_{ij}) \quad (4)$$

$$\dot{D}_{ij} = -\frac{D_{ij}}{\tau_-} + A_- X_i(t), \quad (5)$$

where $X_j(t)$ is the spike train of neuron j defined as a sum of Dirac functions over the action potential times $t_j^{\mathrm{AP}_k}$ equal to $\sum_k \delta(t - t_j^{\mathrm{AP}_k})$, P_{ij} is the potentiation, modeling the influence of incoming spikes, and D_{ij} is the depression value, tracking the influence of outgoing spikes. The global parameter values used in this study are presented in Table 1. These were selected by hand-tuning from physiological ranges available in the target neuromorphic hardware.

The spiking neural networks were simulated using the HRLSim™ package [13]. HRLSim™ is a distributed CPU and GPGPU spiking neural simulation environment. HRLSim™ was developed to support the modeling aspects of the SyNAPSE project. It has also been effective in general neural simulation studies [14–17]. The experiments in HRLSim™ are defined in C++. This allows for higher performance as well as compile and run time optimizations. In addition, embodying the model can be accomplished using different mechanisms; including compiling the environment directly in to the experiment.

2.2 Networks

The network presented here consist of an input layer, an output layer with lateral inhibition and a reward modulating layer, see Figure 1. The input and output layers are divided into channels represented by a population of neurons. The connections from the inputs into the outputs however, are random and unstructured. This is done so there is no intentional bias between channels. A key aspect of this network are the diffuse connections of the inhibitory interneurons. These populations project to every other output population; excluding the channel of which they are a part of. This creates on-center off-surround activity where the most active population suppresses the other output populations. The parameters are presented in Table 2. The parameters were selected based on the restrictions of the target hardware.

Table 2: Parameters for the lateral-inhibition network.

A. Neuron parameters

Neural Region	Neurons Per Channel
Input	3
Output	3
Inhibition	3
Reward	1

B. Connections

Source → Destination	Synaptic Conductance $(g_{max}) \cdot (g_{eff})$	Number of Incoming Connections (total)
Input → Output	$(10.0) \cdot (0.25)$	15
Output → Inhibition	$(10.0) \cdot (1.0)$	15
Inhibition → Output	$(10.0) \cdot (1.0)$	15
Reward → Input	$(10.0) \cdot (1.0)$	1

Initially, the network has no knowledge or inherent understanding of their environment. The desired behavior is driven by a conditioned stimulus injection. Stereotyped spiking signals are sent to an input population and all of the reward populations. The timing of the signal is delayed for the target channel so the synaptic learning between the input population and the desired output populations is potentiated, while all other channels are depressed. The stimulus period lasts for either 300 or 500 ms.

Although simple, this class of network is capable of distinguishing competing inputs under noisy conditions. They can also be used as building blocks to perform more complex tasks. To illustrate this concept we combine three of the lateral-inhibition networks. Each is divided into multiple channels with the outputs of two of the channels directly connecting to the corresponding input channel of the third, Figure 2. The connections are made one-to-one at a weight of 0.5, with output channel 1 connected to input channel 1, output channel 2 to input channel 2, and so on. As illustrated in Figure 2, each of the three networks receives a different input signal. Through reinforcement the network can learn to appropriately respond to different combinations of inputs. In this case, these are used to play a first-person shooter, described below.

2.3 Games

2.3.1 Pong

To illustrate the capabilities of these networks a pong style virtual environment was implemented. This version of

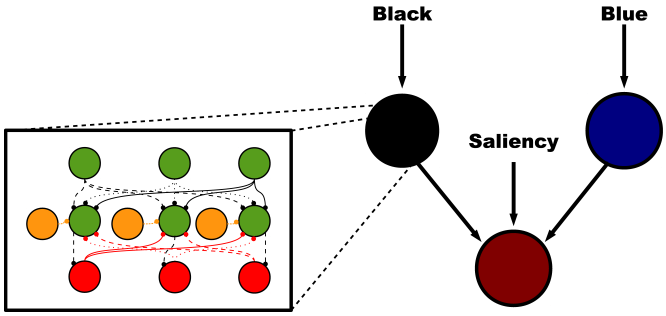

Figure 2: FPS Control Network

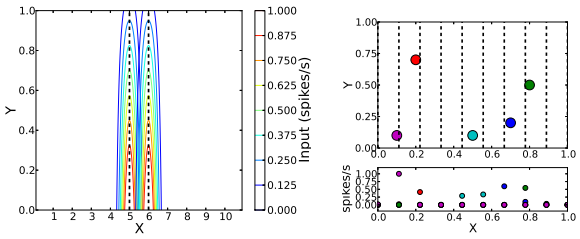

Figure 3: Pong game board disrectizations for a 10 channel network. The spatial width, c, is 0.035. (Left) The overlap between two consecutive channels. (Right) The location of the puck (top) translates to input stimulus for each of the 10 channels (bottom).

the game has a single player controlling the paddle at the bottom of the board. The puck bounces off of the left, right and top walls with minimal physics that change the speed of the puck based on the angle of incidence with the wall. The player has to move the paddle to block the puck from falling through the bottom of the game board.

The game was developed in different stages. First, A mock-up of the game was created in Python using PyGame [18]. A game controller was then developed in C++. However, that controller has no visualization capabilities. It compiles directly into the HRLSim experiment and provides the virtual environment for the networks. The output of the environment is recorded by the controller and can then be played back by the Python visualizer.

The position of the puck in the game space is sent to a number of discretized neural channels. Each of these channels represents a vertical column of the game board. The input signal is Poisson random spike events with a rate determined by a Gaussian curve, described below. This provides a noisy input signal with overlap between channels. The networks signal, through a winner-takes-all mechanism, the position of the paddle.

The stimulus into the network is determined by the location of the puck relative to each of the spatial channels. The location of the puck on the map determines the peak amplitude and center of a Gaussian function defined as

$$f_{X_c}(b) = ae^{-\left((X_c-b)^2/2c^2\right)} \tag{6}$$

where

a	Peak amplitude of the Gaussian function,
b	Center of the Gaussian function,
c	Spatial width or σ of the Gaussian function,
X_c	The non-dimensional location of the channel.

The peak amplitude and Gaussian center are defined as

$$a = Y^* \cdot R_{max} \tag{7}$$

$$b = X^* \tag{8}$$

where

Y^*	Non-dimensional location of the puck in the y dimension,
R_{max}	Maximum input stimulus in $^{Spikes}/_{s}$,
X^*	Non-dimensional location of the puck in the x dimension.

This is visualized in Figure 3 for a spatial widths, $c = 0.035$. The reward or punishment to the network arrives when the puck reaches the bottom of the game board.

The paddle is controlled by a simple proportional controller. The environment receives discrete locations from the neural network. The location on the screen that the paddle has to move to is calculated based on these discrete locations. Its velocity in the X direction is defined by

$$V_x = V_{max} \cdot P. \tag{9}$$

The variable P is the output of the proportional controller defined by

$$P = k \cdot e. \tag{10}$$

Where k is the gain variable and e is the error between the target and current locations

$$e = X_{Location} - X_{Target} \tag{11}$$

The output of the proportional controller, P, is a piecewise linear function that is dependent on the distance from the target.

$$P = \begin{cases} -1 & -e < -\frac{1}{k} \\ 1 & e > \frac{1}{k} \\ e - k & |e| \leq \frac{1}{k} \end{cases}$$

This ensures that the speed of the paddle does not exceed the maximum defined velocity. The pivot point $\frac{1}{k}$ is calculated by setting $k \cdot e = 1$. In addition, the proportionality constant k is less than 1 to ensure that the paddle slows down as it gets closer to its target.

2.3.2 Neuralstein first-person shooter

The first-person shooter (FPS), Neuralstein, is similar to one of the original FPS games, Wolfenstein [19]. This implementation is a rail-shooter where the player moves along a specified path. The player controls its forward movement along that path, where it is aiming and when to take a shot. Similar to the pong environment described above this was implemented at different levels of abstraction. The game engine and visualization was developed in Python, with the latter using PyGame [18] and the Pyggel library [20]. The game engine is abstracted away from the visualization to facilitate faster simulations. Communication with the simulations is provided through a socket server. The engine and the simulation are synchronized so the performance is determined by the slowest component.

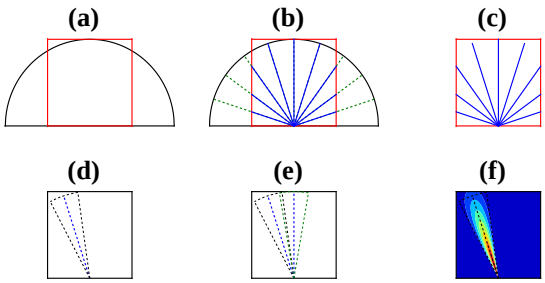

Figure 4: FPS Discretization. (a) A rectangular frame is taken from the hemispherical point-of-view (POV) of the player. (b) The POV space is discretized into equal segments (channels). (c) The resulting frame segments the players view of the world. (d) Each of the channels is centered along equal angular steps about the space with arc-lengths defining the stimulus regime for that channel. (e) The channels are constructed with overlapping stimulus regions to create a noisier environment for the networks to negotiate. (f) The stimulus space for a single channel is defined by a Gaussian function that is railed to the segment boundary.

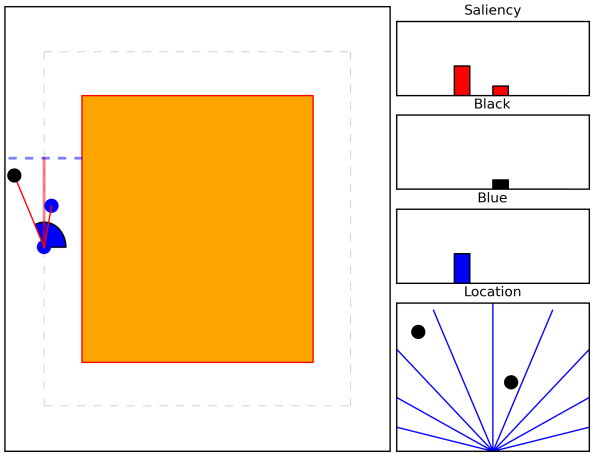

Figure 5: Example stimulus encoding and FPS game board.

The game board is discretized based on the players perspective. The hemispherical point-of-view (POV) for the player is partitioned into a rectangular region, Figure 4 (a). The POV is then segmented into discrete channels with centers at equally spaced angles along the hemisphere, Figure 4 (b). This defines the center for each of the channels that are represented by the network, Figure 4 (c). The channels creates a pie shaped region of interest, Figure 4 (d), which have arc lengths with a 10% overlap between channels, Figure 4 (e). Each of the segments defines that channels stimulus map, which is described by a Gaussian function, Figure 4 (f).

$$f_{\Theta_c}(b) = ae^{-\left((\Theta_c - b)^2/2c^2\right)} \tag{12}$$

Where

a	Peak amplitude of the Gaussian function.
b	Center of the Gaussian function.
c	Spatial width or σ of the Gaussian function.
Θ_c	The non-dimensional angular location of the channel.

The peak amplitude and Gaussian center are defined as

$$a = r^* \cdot R_{max} \tag{13}$$

$$b = \Theta^* \tag{14}$$

Where

r^*	Non-dimensional location of the element in the radial dimension.
R_{max}	Maximum input stimulus in $Spikes/s$.
Θ^*	Non-dimensional angle of the element relative to the player.

The overall arena is a square track with equal width, Figure 5. As the player moves through the environment game elements enter into the view of the player. Elements in the players POV are picked up and their location in that view creates the input stimulus injected into the saliency channels of the network.

There are two types of game elements in the current version. The primarily black characters are considered dangerous and the characters with blue accenting are considered innocuous. Each of these creates a different input into the black and blue channels respectively. It is assumed that a separate mechanism identifies the element and determines which channel is stimulated. For this implementation the game engine directs the stimulus. Figure 5 illustrates what the stimulus for two different game elements would be.

3 Results

3.1 Pong performance analysis

There are a number of additional factors that determine how well the network performs in the game task. The first is the spatial width of the Gaussian stimulus curve, c. This affects the overlap between channels, the larger the value of c the larger the overlap. For testing we use three spatial widths, 0.025, 0.035, 0.045. The next factor is the peak of the Gaussian stimulus curve; where the larger the value the more active the input channels become. Two input peaks, R_{max}, are used, 10 Hz and 40 Hz. Finally, the length of reward is another important factor. This determines how long a feedback stimulus lasts and can affect the magnitude of the change in synaptic efficacy. Two values are chosen for this, 300 ms and 500 ms. For each combination of these parameters, 5 simulations of 500 seconds were run. The accuracy, $(saves/opportunities)\cdot 100$, is computed for 25 second windows. The average of the 5 simulations is plotted.

Figure 6 presents the pong performance results. For the 10 Hz stimulus the network performs well throughout the different spatial width/reward period combinations. However, when the peak input stimulus is raised to $40Hz$ the performance with the lower 300 ms reward period drops considerably. For both stimulus peaks the overall performance throughout the parameter space is surprisingly consistent when the reward time is increased to 500 ms. The slopes in the accuracy curves are slightly different but all approach an accuracy of 100%.

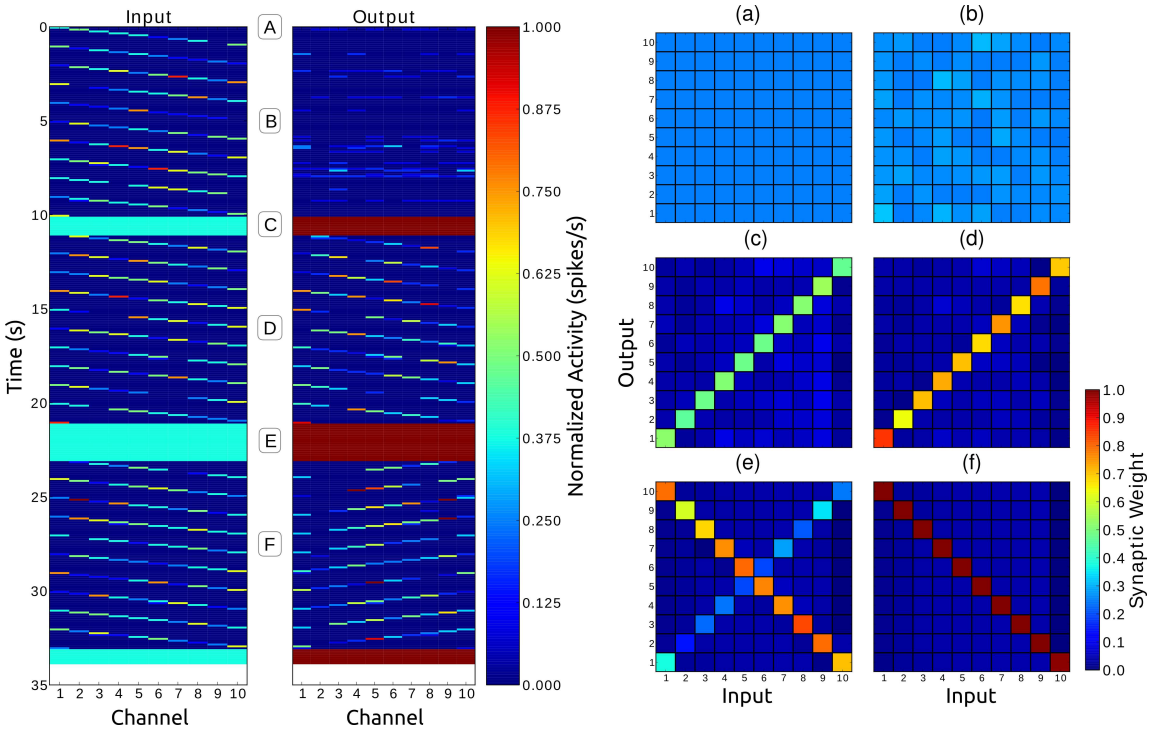

Figure 7: Left: Example lateral inhibition network reward-learning scenario. Activity rate map of the example scenario. Activity was calculated using a moving Gaussian weighted window. Right: Average and maximum synaptic weights between input/output pairs after learning. (a) 0 sec. (b) 10 sec. (c) 11 sec. (d) 21 sec. (e) 22 sec. (f) 33 sec.

3.2 Learning capabilities

An important characteristic of this class of networks is the ability to not only learn arbitrary pairs but then later learn new ones. The rules of the game can be changed and through the same feedback mechanisms the networks will adjust to the new rules. This scenario is illustrated by the spiking activity presented in Figure 7. The stages, marked by the letters in the center are:

A. The network is initialized with all input/output connections have a synaptic USE value of 0.25; as illustrated in Figure 7a by the heat map of the average weights between input/output populations.
B. A Poisson random input is injected into consecutive channels for 10 seconds to establish the basal activity of the network. The resulting average synaptic weight matrix is shown in Figure 7b
C. Alternating reward signals are sent to establish single input/output pairs. The weight matrix is now dominated by the diagonal shown in Figure 7c.
D. The repeated Poisson input signals from B are injected for 10 seconds. After this, the weight matrix shown in Figure 7d demonstrates further potentiation of the established input/output pairs and a continued depression of the other connections.
E. An opposite set of input/output associations are established using alternating reward signals. For stable retraining of the network the reward protocol needs to be about twice as long as the original training. The new weight matrix is shown in Figure 7e.
F. 10 seconds of the repeated Poisson inputs illustrate the newly established input/output pairs, Figure 7f.

The importance of this capability should not be overlooked. Adapting in changing environments is essential for an entity to thrive. This adaptation is similarly vital for artificial agents and for the successful deployment of neuromorphic models.

3.3 Neuralstein first-person shooter

The combination of three LI networks allows for more complex decision making. The individual networks can learn to weight different classes of input information based on reward feedback and the results can be combined to perform different tasks. In the network presented here each of the subnetworks has 9 channels, with the Black and Blue subnetworks both feeding into the action selection (AS) subnetwork. The AS subnetwork also receives saliency information from the environment.

Using the same stereotyped reward mechanisms, the FPS network can be trained to perform more complex action selection tasks. In this case the Black and AS subnetworks learn a one-to-one correlation, while the Blue subnetwork is effectively disconnected. The result is that the saliency information alone is not enough to cause the AS network to cross the selection threshold. A complementary input is required from one of the other subnetworks, in this instance only a black game element can contribute, Figure 8. The resulting network learns to ignore the innocuous blue elements while focusing on the dangerous Black ones.

When placed in the Neuralstein environment the network can move through the environment and target enemies when in

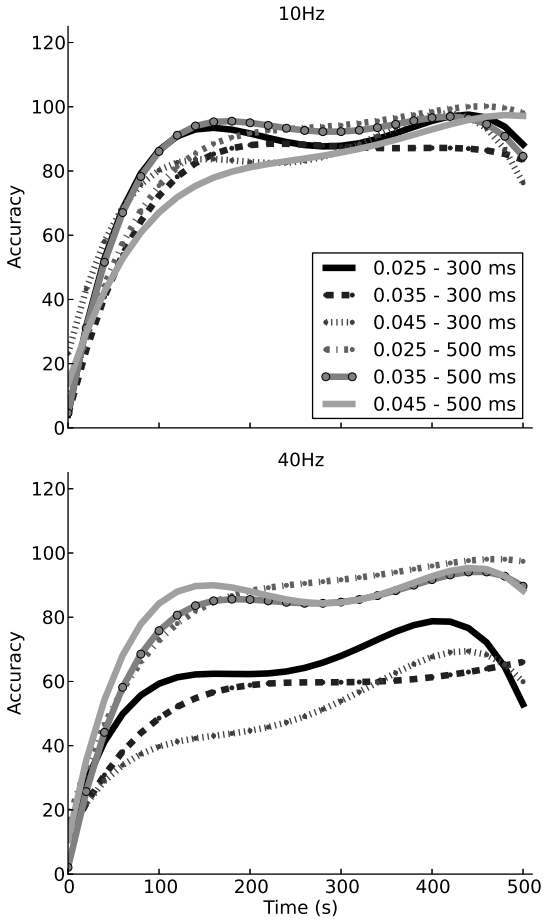

Figure 6: LI network pong performance for 300 ms reward. Top: 10 Hz stimulus peak. Bottom: 40 Hz stimulus peak. The y-axis is the accuracy of the network. A value of 100 means the network blocked all of the pucks in that 25 second block.

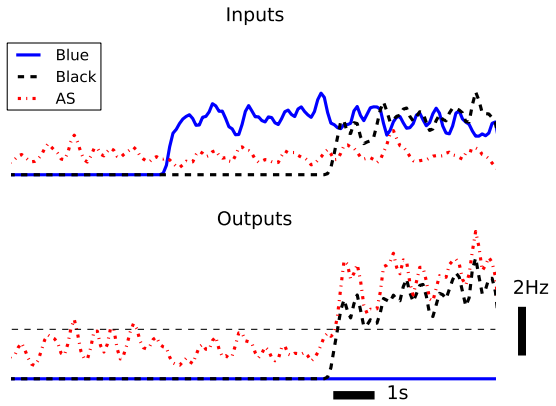

Figure 8: FPS single channel activity after training. The saliency input alone is not enough to push the AS subnetwork above the selection threshold (dashed gray line). The addition of a blue stimulus is ignored and thus does not contribute to the AS subnetwork activity. When a black element stimulus is added the activity of the AS subnetwork is driven passed the selection limit and that channel is selected.

view (not shown). In addition, when presented with both types of game elements, the network can appropriately select the black element, even when the blue one is closer to the player, as seen in Figure 9.

4 Conclusion

4.1 Pong

The learning of channel associations is somewhat arbitrary in the examples presented here. The correlation between input and output populations can in fact be engineered to have more complex relationships than a simple pair. As illustrated by the FPS network results, other combinations can be created as well as mechanisms for more intricate information processing.

The tracking of the puck in the pong networks is reactive, with movements made based on the current position in the game. In the future this concept will be extended to include predictive control of the paddle. A recurrent network capable of learning these kinds of associations could be included along side the reactive networks presented here to achieve this. Initially all of the weights would be random. Through the feedback mechanisms demonstrated here the reactive networks can be trained to track the position of the puck. This learned behavior can then be used as an training signal to the predictive networks.

4.2 Neuralstein

First-person shooters have been extremely popular in Artificial Intelligence (AI) research. The complex interactions between the environment, game elements and multiple players, challenge non-player controllers in unique ways. This popularity has even led to competitions, such as the Botprize, where the goal is to create the most “human like” AI controller [21].

Due to the different strategies required to successfully play a modern FPS, traditional AI domains have dominated [22, 23]. It is the complexity of the task that makes it attractive to embodied modeling. The approach taken here relies on abstracting some of that complexity away. As the networks become more capable other aspects of the FPS paradigm can be added.

4.3 Future work

These simple feed-forward networks are a satisfactory start to employing the SyNAPSE neuromorphic architecture in embodied modeling. Alone, they can be utilized as configurable controllers but their real potential lies in their use as building blocks in more complex control systems. We have already demonstrated how these can be connected together in a simple configuration but in the future these will be combined with more sophisticated networks. For example, recurrent networks can provide, through feedback, state information of the system. This basic form of short-term memory can process the temporal aspects of a system’s inputs and allow for more intelligent processing.

Finally, the feedback for the networks was dependent on conditioned input stimulus to the reward modulation populations. The games played the role of the critic. In the future, more sophisticated reward and punishment signals, such as those in Florian *et al.* [24] and Friedrich *et al.* [25], will be

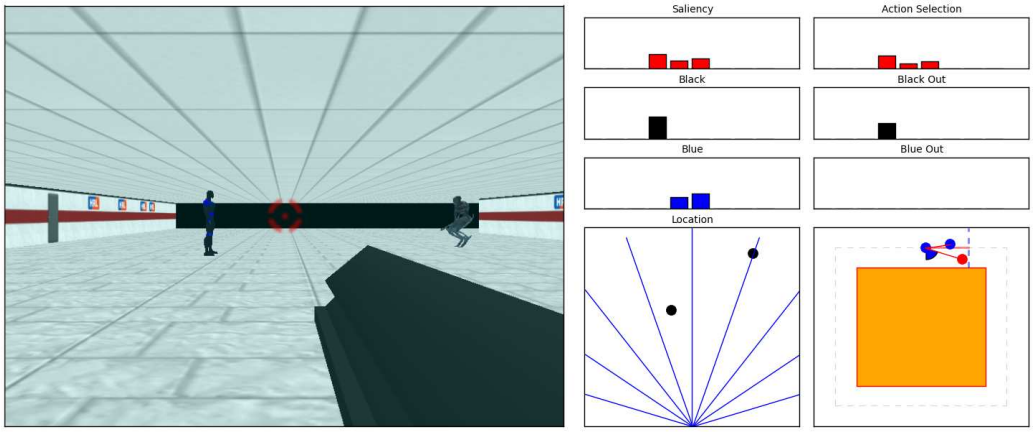

Figure 9: FPS Game Play. Although the blue character is more salient in the player's field of vision the network appropriately targets and shoots the enemy player.

implemented to find a generic reward critic and more efficient controllers.

Acknowledgments

The authors gratefully acknowledge the support for this work by Defense Advanced Research Projects Agency (DARPA) SyNAPSE grant HRL0011-09-C-001. This work is approved for public release and distribution is unlimited. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the DARPA or the Department of Defense.

References

[1] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D.S. Modha. A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm. In *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pages 1 –4, sept. 2011.

[2] N. Srinivasa and J.M. Cruz-Albrecht. Neuromorphic adaptive plastic scalable electronics: Analog learning systems. *Pulse, IEEE*, 3(1):51 – 56, jan. 2012.

[3] Michael X. Cohen and Michael J. Frank. Neurocomputational models of basal ganglia function in learning, memory and choice. *Behavioural Brain Research*, 199(1):141 – 156, 2009.

[4] V. Chakravarthy, Denny Joseph, and Raju Bapi. What do the basal ganglia do? a modeling perspective. *Biological Cybernetics*, 103:237–253, 2010.

[5] Janet Wiles, David Ball, Scott Heath, Chris Nolan, and Peter Stratton. Spike-time robotics: A rapid response circuit for a robot that seeks temporally varying stimuli. *Australian Journal of Intelligent Information Processing Systems*, 11(1), 2010.

[6] Harald Burgsteiner. Imitation learning with spiking neural networks and real-world devices. *Engineering Applications of Artificial Intelligence*, 19(7):741 – 752, 2006.

[7] P. Arena, L. Fortuna, M. Frasca, and L. Patane. Learning anticipation via spiking networks: Application to navigation control. *Neural Networks, IEEE Transactions on*, 20(2):202 –216, feb. 2009.

[8] RăzvanV. Florian. Spiking neural controllers for pushing objects around. In Stefano Nolfi, Gianluca Baldassarre, Raffaele Calabretta, JohnC.T. Hallam, Davide Marocco, Jean-Arcady Meyer, Orazio Miglino, and Domenico Parisi, editors, *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 570–581. Springer Berlin Heidelberg, 2006.

[9] D.R.W. Barr, P. Dudek, J.M. Chambers, and K. Gurney. Implementation of multi-layer leaky integrator networks on a cellular processor array. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 1560 –1565, aug. 2007.

[10] J.V. Arthur, P.A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S.K. Esser, N. Imam, W. Risk, D.B.D. Rubin, R. Manohar, and D.S. Modha. Building block of a programmable neuromorphic substrate: A digital neurosynaptic core. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1 –8, june 2012.

[11] N. Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biol. Cybern.*, 95:1–19, June 2006.

[12] Sen Song, Kenneth D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926, 2000.

[13] K. Minkovich, C.M. Thibeault, A. Nogin, Youngkwan Cho, M. J. O'Brien, and N. Srinivasa. HRLSim: A high performance spiking neural network simulator for GPGPU clusters. *Neural Networks and Learning Systems, IEEE Transactions on, to appear*, 2013.

[14] N. Srinivasa and Y. Cho. Self-organizing spiking neural model for learning fault-tolerant spatio-motor transformations. *Neural Networks and Learning Systems, IEEE Transactions on*, PP(99):1, 2012.

[15] M. J. O'Brien and N. Srinivasa. A spiking neural model for stable reinforcement of synapses based on multiple distal rewards. *Neural Computation*, 25:123–156, 2013.

[16] Corey Michael Thibeault and Narayan Srinivasa. Using a hybrid neuron in physiologically inspired models of the basal ganglia. *Frontiers in Computational Neuroscience*, 7(88), 2013.

[17] Narayan Srinivasa and Qin Jiang. Stable learning of functional maps in self-organizing spiking neural networks with continuous synaptic plasticity. *Frontiers in Computational Neuroscience*, 7(10), 2013.

[18] Pete Shinners. Pygame. `http://pygame.org/`, 2012. Accessed: 10/22/2012.

[19] id Software. Wolfenstein 3D. `http://www.idsoftware.com/games/wolfenstein/wolf3d`, 2012. Accessed: 11/2/2012.

[20] Pyggel group. Pyggel. `http://www.pygame.org/project-PYGGEL-968-.html`, 2012. Accessed: 10/22/2012.

[21] BotPrize. The 2K BotPrize. `http://botprize.org/index.html`, 2012. Accessed: 10/16/2012.

[22] N. van Hoorn, J. Togelius, and J. Schmidhuber. Hierarchical controller learning in a first-person shooter. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 294 –301, sept. 2009.

[23] J. Schrum and R. Miikkulainen. Evolving agent behavior in multiobjective domains using fitness-based shaping. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 439–446. ACM, 2010.

[24] Rzvan V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 6 2007.

[25] Johannes Friedrich, Robert Urbanczik, and Walter Senn. Spatio-temporal credit assignment in neuronal population learning. *PLoS Comput Biol*, 7(6), 06 2011.