A Reduced Spiking Neural Network Architecture for Energy Efficient Context-Dependent Reinforcement Learning Tasks

2024 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-3099-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCAS58744.2024.10558211

Hira Rasheed*, Peyman Mirtaheri†, Ali Muhtaroglu †‡
Dept. of Machines, Electronics & Chemistry, Faculty of Technology, Art and Design (TKD)
Oslo Metropolitan University, PO Box 4, St. Olavs plass, 0130 Oslo, Norway
* Biomedical Engineering MS Program † Advanced Health Intelligence and Brain-Inspired Technologies (ADEPT)
‡ Email: ali.muhtaroglu@oslomet.no

***Abstract*—Neuromorphic circuits and systems involving spiking neural networks (SNN) have resulted in disruptive advances in performance/joule for relevant applications. A novel reinforcement learning (RL) digital hardware architecture is presented in this work that achieves energy consumption improvements through three fundamental techniques: The first two techniques comprise reduction in the complexity of the arithmetic unit for the optimization of recurring synapse and neuron cores in the network array, which is inspired by "crude" nature of the building blocks in the biological neurons that are tolerant to inaccuracy and noise. As the third technique, the RL SNN middle (hippocampus) layer is equipped with a simple scratchpad to facilitate temporal hysteresis in synaptic plasticity during the RL processes of long-term potentiation/depression (LTP / LTD). This feature is inspired by the non-temperamental behavior of biological synapses. The intelligent allocation significantly reduces learning time in a given task. Implementation on Intel Cyclone IV FPGA demonstrated significant advantages in cost, power dissipation and execution time, resulting in more than two orders of magnitude benefit in energy consumption for a context-dependent learning task on a 16-node 3-layer RL network presented in the literature.**

***Index Terms*—Reinforcement learning (RL), spiking neural network (SNN), neuromorphic hardware, low-power, low-cost, low-energy, context-dependent task.**

I. Introduction

Energy-efficient design of spiking neural networks (SNNs) have received increasing focus due to the present emphasis on brain-inspired computing for recognition and classification tasks at the edge through massively parallel and event driven structures [1]. An important concept in SNNs is Spike-Timing-Dependent Plasticity (STDP), through which temporal patterns between presynaptic and postsynaptic spikes are used to modify the strength of synaptic connections. Many models exist in the literature for spiking neurons with varying computational complexity to serve different applications [2]. The well-known Hodgkin-Huxley model [3] for example requires solving a minimum of four differential equations with tens of parameters using floating point arithmetic.

An artificial learning agent in a reinforcement learning (RL) system discovers which actions yield the most reward through trials [4]. Sequential processes in RL can improve decision making in in autonomous robots. RL problems involving control in simple autonomous systems do not require Hodgkin-Huxley complexity. The Leaky integrate-and-fire (LIF) model [5], for example, provides a good compromise between accuracy and complexity. Custom analog LIF models face challenges such as extended design time, process variations, and noise [6]. Recent digital realizations have targeted complexity reductions to improve energy efficiency and cost. ODIN, for example, demonstrates enormous energy efficiency benefits compared to software solutions, but extensive flexibility in the design comes at the cost of 55 neuron state bits, 70 configuration state bits, and 1 state bit for choosing between LIF and alternative neuron models [7]. Among platforms with Field-Programmable Gate Arrays (FPGAs), multiplier-less approaches have received focus in achieving higher speed with lower cost and power dissipation [8] – [10]. In both [9] – [10], a 32-bit fixed-point representation has been selected for representing digitized neuron voltage potential and synapse weights, and multiplication has been replaced with a shift operation. FPGA implementation in [10] builds on recent animal studies with short sequences to learn from single stimulus-response pairs across multiple contexts [11]. Although the required RL network size is modest, a synapse module in this work consumes 172 slice LUTs and 38 flip-flops (state elements).

Given a context-dependent task with distinct rewarding and non-rewarding action sequences [11], the following contributions are presented in this work:

i. *Temporal hysteresis in synaptic plasticity*: The proposed RL algorithm reduces the total learning time by avoiding execution of both potentiation and depression operations on the same synapses close together in time.
ii. *Coarse computation*: Processing logic is simplified to support essential features of the LIF neuron, synapse, and the RL network using a 5-bit integer architecture.
iii. *Reduced timing complexity*: STDP is implemented with a shorter time span, which significantly simplifies synapse implementation due to the elimination of timing counters and complex time-dependent adjustment factors.

Temporal hysteresis in synapse plasticity is inspired by the ”slow” biological changes associated with the development of synapses surrounding a neuron, which have implications to how efficiency may be related to careful selection of synapses that are oxygenated versus starved during the learning process [12]. The hardware reductions associated with coarse computation and reduced timing complexity are also bio-inspired in the sense that the biological brain does not generate tissues and signals beyond the requirement of the learning task, and is not biased by the traditional precise number systems of human-built computers. Although custom implementations have targeted aggressive logic reductions and compact footprints for neurons and synapses to save energy [7] [13], the problem of architectural reduction for energy saving remains for RL networks in FPGA-based edge computing.

The rest of the paper is organized as follows: Implementation of the general state-of-the-art context-dependent RL task and proposed architectural features (i-iii) are further discussed in the next section. Section III presents complexity analysis, verification tests, and results. Finally, the conclusions are summarized in the last section.

II. Implementation

A. Context-Dependent RL Task and Network

A relatively simple RL model for a context-dependent task has been selected based on [11], where the task has a total of four physical locations, two in context A (A1, A2) and two in context B (B1, B2), as depicted in Fig. 1(a). At each location item X or Y can be found. This results in eight triplets (e.g. A1X, A2X, A1Y, ...) as input combinations, which correspond to the activation of different pairs of sensory inputs in the SNN, as shown in Fig. 1(b). As the mouse chooses between the two contexts and a location within the context, it will be rewarded for exactly half of these triplets. Once activations from one of the sensory neuron pairs propogate through the second layer to the third, the motor neurons generate one of the ”dig” or ”move” actions.

Inhibitory interconnects among hippocampus and motor-layer neurons, illustrated with dashed lines in Fig. 1(b), support the winner-take-all (WTA) network as explained in [10]. Inhibitory non-plastic synapses were integrated into the network for this purpose with substantial negative weights in [10]. In the proposed organization, each inhibitory connection is implemented through a low-cost combinational dendrite link instead, with a fixed negative value activated by the neuron that fires first in a given layer. As the supported number resolution reduces it is expected that multiple neurons in a layer fire simultaneously. Prioritization is therefore implemented in the proposed scheme, which can be configured through multiplexed inhibition wires as part of the random initialization of the network. Integrating random WTA priority reduces the latency of the exploratory RL trials.

The machine learning algorithm comprises repeated executions through behavioral and replay modes, as depicted in Fig. 2. Random triplets are generated in the first phase (behavioral mode) to represent exploratory trials. There may be many generated ”Move” motor actions before the trial ends in a ”dig”, which transfers the machine to the replay mode (second phase). Depending on if the ”dig” is rewarding or not, the network goes through either potentiation or depression of involved synapses that led to the last ”move” and the ”dig”. This requires the machine to save all associated neuronal activity. Replay phase is illustrated in Fig. 3. The arrows in Fig. 3(a) indicate the time sequence in which different neurons are repeatedly potentiated through direct stimuli in forward order to increase the synapse weights for a rewarding action. Depression process for non-reward cases is illustrated in Fig. 3(b), where arrows indicate the time sequence for reverse replay through direct stimuli in reverse order to decrease the synapse weights on the shown paths.

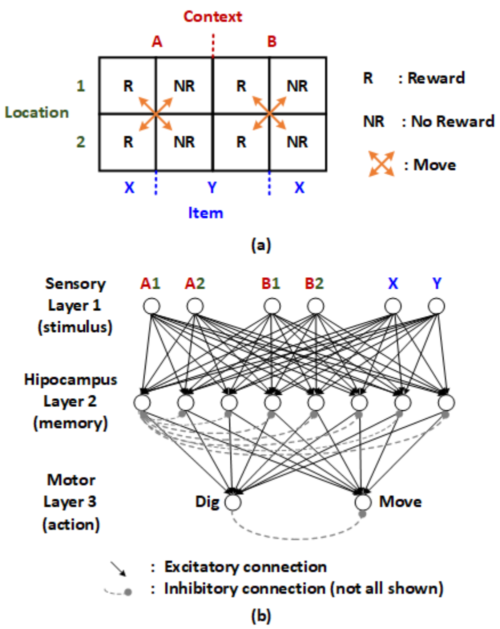

Fig. 1: Illustration of (a) the context-dependent task, and (b) SNN to realize RL for the given task [11].

B. Temporal Hysteresis in Synaptic Plasticity

Random initialization of neuron potentials and synapse weights may result in many iterations of moves within Phase 1 before any replay can be done for learning, which inevitably increases the learning latency. In addition, potentiation and

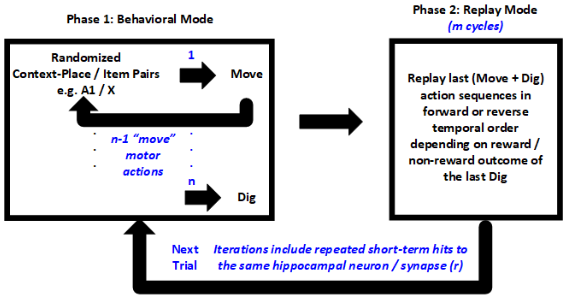
Phase 1: Behavioral Mode
Randomized Context-Place / Item Pairs e.g. A1 / X
1
Move
n-1 “move” motor actions
n
Dig
Phase 2: Replay Mode
(m cycles)
Replay last (Move + Dig) action sequences in forward or reverse temporal order depending on reward / non-reward outcome of the last Dig
Next Trial
Iterations include repeated short-term hits to the same hippocampal neuron / synapse (r)

Fig. 2: State of the art Machine RL algorithm [10].

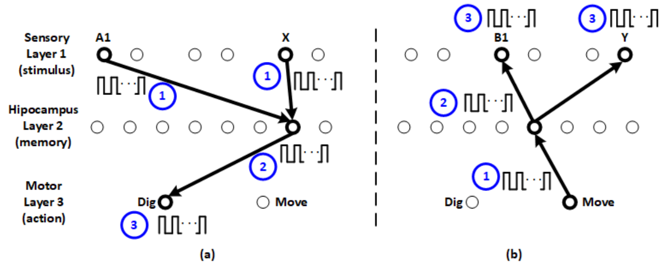

Fig. 3: Replay time sequence (a) in forward order for repeated potentiation in rewarding neural paths, for example, sensory triplet X1A leading to action ”dig”, and (b) in reverse order for repeated depression in non-rewarding neural paths as sensory triplet B1Y leading to action ”move”.

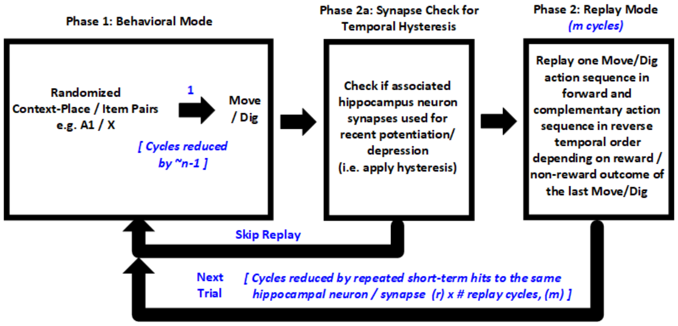
Phase 1: Behavioral Mode
Phase 2a: Synapse Check for Temporal Hysteresis
Phase 2: Replay Mode
(m cycles)
Randomized Context-Place / Item Pairs e.g. A1 / X
1
Move / Dig
[Cycles reduced by ~n-1]
Check if associated hippocampus neuron synapses used for recent potentiation/ depression (i.e. apply hysteresis)
Replay one Move/Dig action sequence in forward and complementary action sequence in reverse temporal order depending on reward / non-reward outcome of the last Move/Dig
Skip Replay
Next Trial
[Cycles reduced by repeated short-term hits to the same hippocampal neuron / synapse (r) x # replay cycles, (m)]

Fig. 4: Proposed algorithm to integrate temporal hysteresis into synapses to reduce the overall RL latency.

depression operations may be applied to the same synapses across consecutive replays, which is expected to resolve in the right direction in time, but time and energy are wasted in the process. The proposed algorithm, shown in Fig. 4, switches to Phase 2 when the SNN spiking results in either a ”move” or ”dig” action. Temporal hysteresis is implemented through an additional simple look-up table (LUT) that tracks if the last sequence’s synapses have recently been modified. If they are, the replay cycles are skipped. Once the LUT buffer is filled, the same synapses can be modified again, allowing plasticity to apply after fully utilizing the existing hippocampus neurons.

C. Coarse Computation

RL problem does not frequently require high accuracy. Studying the context-dependent RL task in [11] for example, the problem can be solved using a reduced SNN core with coarse computational capabilities. Proposed reductions compared to [10] are demonstrated by comparing fundamental design parameters in Table I, and are further illustrated in Fig. 5. Reductions in the synapse algorithm are further detailed in Section II.D. The most significant change is in the resolution from 32 to 5 bits. Further investigation of lower resolution is planned in the future, but 5-bit representation is selected in this work to demonstrate concepts with some resilience against noise. It is expected (and further evaluated in the next section) that core processor cost and power dissipation will scale down roughly in the ratio 32:5 due to these changes.

TABLE I: Parameter Values to Illustrate Reductions with the Coarse Computation Approach

Parameter	[10]	Proposed
$N.M.$ threshold, V_{th}	-50 mV	15
$N.M.$ reset potential, V_{rst}	-70 mV	5
$N.M.$ leakage voltage, $V_{leakage}$	$1.2 \times 10^{-7} V$	1
$S.A.$ when $\Delta t > 0, A^+$	$+2^{-10}$	1
$S.A.$ when $\Delta t < 0, A^-$	$+2^{-11}$	-1
$S.W.$ max. activation, W_{max}	1	2
$S.W.$ min. activation, W_{min}	0	0
Number representation, Nb	32 fixed-point	5-bit integer
# In / Hidden / Out Neurons, $N_{I/H/O}$	6 / 8 / 2	6 / 8 / 2
$I.V.$ for input neurons, V_{input}	1.28 mV	25
$I.V.$ for hidden neurons, V_{hidden}	1.48 mV	25
$I.V.$ for output neurons, V_{output}	1.64 mV	25

$N.M.$: Neuron membrane $S.A.$: STDP spike amplitude
$S.W.$: STDP Synaptic weight $I.V.$: Input voltage

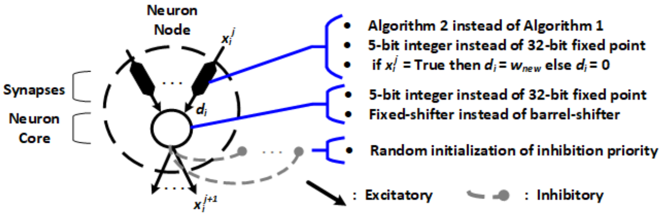
Neuron
Node
x_i^j
Synapses
Neuron
Core
d_i
x_i^{j+1}
Algorithm 2 instead of Algorithm 1
5-bit integer instead of 32-bit fixed point
if x_i^j = True then $d_i = w_{new}$ else $d_i = 0$
5-bit integer instead of 32-bit fixed point
Fixed-shifter instead of barrel-shifter
Random initialization of inhibition priority
: Excitatory
: Inhibitory

Fig. 5: Illustration of optimizations at each neuron node.

D. Reduced Timing Complexity

State-of-the-art methods require tracking of spike timing to implement the complete STDP. Synapse weights are modified for Long-Term-Potentiation (LTP) or Long-Term-Depression (LTD), depending on if the difference in pre- and post-synaptic spike time is negative or positive, respectively. The original modification function for STDP synapse weights decays exponentially with this time difference. A simplified form of this function, as implemented by the synapse algorithm in [10], is summarized in Algorithm 1. A^+ and A^- here were previously introduced in Table I. The approach is appealing since it replaces multiplication in weight modification with shift operation. Algorithm 2 illustrates further reductions in the complexity of event timing, as proposed in this work, because pre- and post-synaptic event timers, timing buffers, and difference calculators are eliminated. w_{adj} is a constant increment or decrement value that is equal to A^+W_{max} or A^-W_{max}, to relate to the proposed parameters in Table I. A or W are eliminated in the proposed algorithm. Spike causality tracking is achieved by delaying or staging the digital (1-bit) pre-synaptic spike signal sufficiently, which is only one clock cycle in the proposed implementation. The staged signal is compared with the post-synaptic spike signal and the unstaged version of the pre-synaptic signal at the same time.

III. Verification and Results

The synchronous RL network with 16 neuron nodes is implemented in SystemVerilog, where each neuron node consists of a 4-state neuron control FSM, a 5-bit neuron datapath, and a 5-bit synapse. The novel features compared to the architecture in

Algorithm 1 Simplified plastic synapse algorithm [10]

```
Input pre- and post-synaptic spikes, $w_{previous}$
Output $w_{new}$
if (Spike pre = True) then Store $T_{pre}$
if (Spike post = True) then Store $T_{post}$
$\Delta t \leftarrow T_{pre} - T_{post}$
if $\Delta t > 0$ then
    $w_{new} \leftarrow w_{previous} + (w_{previous} >> A^{+})$
else if $\Delta t < 0$ then
    $w_{new} \leftarrow w_{previous} - (w_{previous} >> A^{-})$
else
    $w_{new} \leftarrow w_{previous}$
```

Algorithm 2 Reduced timing complexity (proposed)

```
Input pre-, pre_staged- and post-synaptic spikes, $w_{previous}$, $w_{adj}$
Output $w_{new}$
if ((Spike pre & post = True) & (Spike pre_staged = False)) then
    $w_{new} \leftarrow w_{previous} - w_{adj}$
else if (Spike pre_staged & post = True) then
    $w_{new} \leftarrow w_{previous} + w_{adj}$
else
    $w_{new} \leftarrow w_{previous}$
```

[10] are discussed in Section II. The proposed RL algorithm in Fig. 4 runs offline using a SystemVerilog testbench to evaluate the approach. The network is initialized using semi-random synapse weight and neuron membrane voltage assignments as well as semi-random WTA prioritization, as explained in Section II.A. Close to 50 trials are executed during the behavioral (exploratory) phase to cover all possible input triplet/output action combinations which are followed by the replay phase. A replay is executed after each action that is not filtered by the temporal hysteresis in synaptic plasticity as discussed in Section II.B.

Table II quantifies the theoretical and FPGA-synthesized / simulated advantages of the reduced architecture in comparison to the multiplier-less architecture presented in [10]:

Cost: In addition to the coarse computation approach, secondary cost benefits are expected from the reduced timing complexity and simplicity of the RL algorithm. The latter is in direct compliance with the law of diminishing returns for energy efficiency [14]. Synapse cost reduction in the first two rows closely follows the theoretical expectation. The overall RL network (next two rows) synthesizes with better than theoretical gain because the learning algorithm (Fig. 2) is implemented in hardware in the case of [10] whereas the evaluation in this work was done through offline version of the proposed algorithm (Fig. 4) using a testbench.

f_{max}: Kintex7 FPGA in [10] uses a version of TSMC 28nm high-performance, low-power (HPL) technology with 1.0 V nominal core supply, while Cyclone IV GX utilized in this work is fabricated on 60nm low-power (LP) technology powered by 1.2 V core supply. The theoretical frequency scaling estimation in the table assumes a similar number of gates on the worst-case speed path and applies technology and voltage scaling based on [15]. Simulated synapse f_{max} using CAD timing analysis tools is consistent with the rough expectation, possibly reflecting some improvement in the worst-case synapse speed path in our design.

TABLE II: Normalized Theoretical and Synthesized / Simulated Benefits Compared to [10]

Design Metric	Normalized Theoretical Estimate		CAD Synthesized / Simulated	
	Reference ([10])	Improvement (Proposed)	Reference ([10])	This Work (Proposed)
Synapse Cost, *S$*	1	5 / 32	38 F $^{\alpha}$ 172 L	6 F 24 L
RL Cost, *RL$*	1	5 / 32	8096 F 19059 L	538 F 2648 L
Clk Freq, f_{max}^{γ}	1	2.2	151 MHz	355 MHz
Cycle Cnt, CC	1	$1 / (mnr)^{\beta}$	30000	2000
Pwr Cons, P	1	5 / 32	1.81 W	0.198 W
Engy Cons, E	1	$1 / 192^{\delta}$	2257 μJ	6 μJ

$^{\alpha}$ F =# slice flipflops, L:# slice LUTs $^{\beta}$ Fig. 4 $^{\gamma}$ Synapse frequency
$^{\delta}$ $E = CC \times P \ / \ f$ where CC ratio = 1/15 based on simulations

Cycle Count: As illustrated in Section II, Fig. 4, each trial ending in motor activation may result in a replay in the proposed RL algorithm. This may theoretically reduce trials by *O(n)*. Temporal hysteresis in synaptic plasticity will block some of the replays from happening in the proposed scheme because the associated synapse has recently been ”replayed” for learning. This would theoretically reduce trials by another factor of *O(r)*. Finally, if each replay procedure consists of O(m) clock cycles, this results in a theoretical reduction in cycle count as *O(mnr)*. Testbench simulations on the right-hand side of Table II indicate a cycle count reduction of about an order of magnitude. The number in the last column (2000) is partly simulated, with some estimated portion for the penalty of additional logic to be added to the hardware for implementing the RL algorithm in Fig. 4.

Power Consumption: The reduction in average power consumption in simulations is consistent with theoretical expectation, and the additional reduction is due to the high level control logic in [10], as discussed above.

Energy Consumption: The reductions in cycle count, average power dissipation and the increase in frequency all contribute to more than two orders of magnitude benefit in energy consumption in theory, as shown in the last row of the table.

IV. Conclusion and Discussion

Energy-efficient SNNs are critical before context-dependent RL can contribute to future autonomous robots and energy-autonomous edge AI. The presented work demonstrates new architectural techniques can take inspiration from resilient synaptic connectivity and approximate computation properties in biological brain to reduce hardware cost, size, power, and energy dissipation while simultaneously increasing performance. The complexity of the electronics is reduced to match the complexity of the learning task to attain more than two orders of magnitude simulated improvement in energy consumption on Intel Cyclone IV FPGA, using an example from a 3-layer 16-node RL network in the literature that modeled a previous traveling-mouse RL experiment.

References

[1] Y. Dan and M. M. Poo, ”Spike timing-dependent plasticity of neural circuits,” Neuron, vol. 44, no. 1, pp. 23–30, 2004.

[2] E. M. Izhikevich, “Which model to use for cortical spiking neurons?” IEEE Trans. Neural Netw., vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[3] A. L. Hodgkin and A. F. Huxley,“A quantitative description of mem-brane current and application to conduction and excitation in nerve,”J.Physiol., vol. 117, pp. 500–544, 1954.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Cambridge, U.S.A.: The MIT Press, 2018.

[5] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge, U.K.: Cambridge Univ. Press, 2002.

[6] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, ”Energy-Efficient Neuron, Synapse, and STDP Integrated Circuits,” IEEE Transactions on Biomedical Circuits and Systems, vol. 6, no. 3, pp. 246–256, 2012.

[7] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol, “A 0.086-mm2 12.7-pJ/SOP 64k-synapse 256-neuron Online-learning Digital Spiking Neuromorphic Processor in 28 nm CMOS,” IEEE Trans. Biomed. Circuits Syst., vol. 13, no. 1, pp. 145–158, Feb. 2019.

[8] H. Soleimani, A. Ahmadi, and M. Bavandpour, “Biologically Inspired Spiking Neurons: Piecewise Linear Models and Digital Implementation,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 12, pp. 2991–3004, Dec. 2012.

[9] E. Z. Farsa, A. Ahmadi, M. A. Maleki, M. Gholami, and H. N. Rad, “A Low-cost High-speed Neuromorphic Hardware Based on Spiking Neural Network,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 66, no. 9, pp. 1582–1586, Sep. 2019.

[10] H. Asgari, B. M. Maybodi, R. Kreiser, and Y. Sandamirskaya, “Digital Multiplier-Less Spiking Neural Network Architecture of Reinforcement Learning in a Context-Dependent Task,” IEEE J. on Emergng. and Selctd. Topcs. in Circuits and Systems, vol. 10, no. 4, pp. 498–511, Dec. 2020.

[11] F. Raudies and M. E. Hasselmo, “A Model of Hippocampal Spiking Responses to Items during Learning of a Context-dependent Task,” Frontiers Syst. Neurosci., vol. 8, pp. 1–12, Sep. 2014.

[12] J. E. Niven,”Neuronal Energy Consumption: Biophysics, Efficiency and Evolution, Current Opinion in Neurobiology,” Volume 41, Pages 129-135, 2016.

[13] M. Davies et al., “Loihi: A Neuromorphic Manycore Processor with On-chip Learning,” IEEE Micro, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[14] B. Sengupta, A. A. Faisal, S. B. Laughlin, J. E. Niven, ”The Effect of Cell Size and Channel Density on Neuronal Information Encoding and Energy Efficiency,” J. Cereb. Blood Flow Metab., vol. 33, no. 9, pp. 1465-73, Sep. 2013.

[15] A. Stillmaker and B. Baas, “Scaling Equations for the Accurate Prediction of CMOS Device Performance from 180 nm to 7 nm,” Integration, vol. 58, pp. 74–81, 2017.