Training Spiking Neural Networks with a Multi-Agent Evolutionary Robotics Framework

Souvik Das
Purdue University
West Lafayette IN 47907, USA
souvik@purdue.edu

Anirudh Shankar
Purdue University
West Lafayette IN 47907, USA
shanka19@purdue.edu

Vaneet Aggarwal
Purdue University
West Lafayette IN 47907, USA
vaneet@purdue.edu

ABSTRACT

We demonstrate the training of Spiking Neural Networks (SNN) in a novel multi-agent Evolutionary Robotics (ER) framework inspired by competitive evolutionary environments in nature. The topology of a SNN along with morphological parameters of the bot it controls in the ER environment is together treated as a phenotype. Rules of the framework select certain bots and their SNNs for reproduction and others for elimination based on their efficacy in capturing food in a competitive environment. While the bots and their SNNs are not explicitly trained to survive or reproduce using loss functions, these drives emerge implicitly as they evolve to hunt food and survive. Their efficiency in capturing food exhibits the evolutionary signature of punctuated equilibrium. We use this signature to compare the performances of two evolutionary inheritance algorithms on the phenotypes, Mutation and Crossover with Mutation, using ensembles of 100 experiments for each algorithm. We find that Crossover with Mutation promotes 40% faster learning in the SNN than mere Mutation with a statistically significant margin.

KEYWORDS

Spiking Neural Networks, Multi-Agent Evolutionary Robotics, Mutation, Crossover

ACM Reference Format:
Souvik Das, Anirudh Shankar, and Vaneet Aggarwal. 2021. Training Spiking Neural Networks with a Multi-Agent Evolutionary Robotics Framework. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3449639.3459329

1 INTRODUCTION

Darwinian evolution through natural selection serves as the broad inspiration for the field of evolutionary computation in defining searches for solutions to optimization problems in high dimensional spaces. Evolutionary algorithms have been used to train the weights and biases of deep artificial neural networks [18]. In this paper, we demonstrate the use of multi-agent evolutionary algorithms, inspired by competition in nature, to train Spiking Neural Networks (SNN) as forms of artificial intelligence. SNNs are a special class of ANNs that mimic the biological dynamics of discrete signaling events between neurons known as spikes [17]. This is in contrast to currently popular ANNs which use real numbers to represent average spiking frequencies. SNNs thus allow for encoding information in the temporal sequence of spikes and offer higher computational capacity per neuron than generic ANNs. The temporal sparseness of spikes also make SNNs attractive candidates for low-energy, neuromorphic hardware implementations [19]. Alluring though SNNs may be, training them requires novel methods since unlike generic ANNs which use continuous and differentiable activation functions in their neurons that lend themselves to gradient descent methods for learning, SNNs define the activation mechanics of their neurons in terms of the time evolution of their membrane potentials. Hence, adapting gradient descent methods for SNNs are not trivial. This motivates us to search for nature-inspired paradigms within multi-agent Evolutionary Robotics to train them.

1.1 Evolutionary Robotics

The field of Evolutionary Robotics (ER) considers the co-evolution of robot morphology and intelligence within an environment of selection, inheritance and mutation [3]. ER may be considered a confluence of evolutionary computation and robotics. In this work, SNNs provide the intelligence of simulated robots in a multi-agent ER arena. We demonstrate a simple ER arena, consisting of an environment and rules, where the SNNs evolve to meet the criteria for reproduction with increasing efficiency. Multi-agent arenas can be challenging to learn in since the actions of one agent affect the options available to another. This sets up indirect interaction between the agents. Our work is the first to bring SNNs to a multi-agent ER arena for effective training.

In this work, synaptic weights of the SNN and morphological parameters of the robot (henceforth referred to as the "bot") together constitute each bot's phenotype. The phenotype is identical to the genotype in our setup. A population of initially random phenotypes are created and let loose in the ER arena as described in Section 2. We investigate two evolutionary inheritance algorithms, Mutation, and Crossover with Mutation, described in Section 3. Learning behavior is seen to emerge in a few generations, including the evolutionary signature of punctuated equilibria. This is described in Section 4. Features of the punctuated equilibria are used to compare performances of the inheritance algorithms.

1.2 Spiking Neural Networks

Spiking Neural Networks are considered to be the third generation of neural networks [12]. The first generation of neural networks used was a binary classifier more commonly known as the perceptron. The second generation of neural networks, more commonly

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07.
https://doi.org/10.1145/3449639.3459329

referred to as artificial neural networks employ continuous non-linear activation functions. Experimental results from neurobiology have paved the way for the more biologically realistic spiking neurons [12]. The temporal sequence of spikes are known to play a role in computation in brains [1, 8, 13]. SNNs have found success in various pattern recognition applications, including image processing and medical diagnosis [4–6, 9, 14, 21]. SNNs may be configured in convolutional, recurrent and deep-belief network forms as well [19]. SNNs are a natural fit for robotics as individual spikes can trigger discrete motor movements, and sequences of spikes at different motor neurons can articulate complex, composite motions.

Learning in SNNs is achieved by optimizing the synaptic weights and spontaneous firing rate of neurons. This may be accomplished by local methods like Spike Timing Dependent Plasticity [7], adaptations of gradient descent techniques [11, 19], or global techniques like evolutionary algorithms [10]. Gradient descent techniques rely on differentiable surrogates for the SNN activation mechanism [2, 15, 16]. Although surrogate gradients have paved the way to perform training, the problem of training multi-layered SNNs efficiently remains challenging. While some forms of evolutionary algorithms have been used to train SNNs, our work distinguishes itself by the use of a multi-agent ER framework.

1.3 Main Contributions

The main contributions of this work are as follows.

(1) Demonstration of a multi-agent ER framework, inspired by competition in nature, to train SNNs. The framework is kept as simple as possible with the smallest set of parameters so we may arrive at general conclusions.
(2) Quantitative characterization of evolutionary learning by fitting punctuated equilibria to logistic curves.
(3) Comparison between the performances of two evolutionary algorithms for training the SNNs: Mutation versus Crossover with Mutation.

2 SYSTEM COMPONENTS

The multi-agent ER framework within which we investigate the efficacy of evolutionary algorithms for SNN training is described in this section. Experiments are performed in a simulated arena consisting of a group of bots, each with an SNN, competing for the capture of "food" in a "game environment" with certain rules. The bot is described in Section 2.1, the SNN is described in Section 2.2, and the game environment and food in Section 2.3. Since the food is replenished after a capture event, the experiments can run indefinitely. The rules of evolution that kick in at each capture event are described in Section 3. The code for this setup may be found at https://github.itap.purdue.edu/das69/EvolutionarySNN.

2.1 Bots

Each bot occupies a circular area (of 40 units squared) and has a position (x, y) and angular orientation θ within a 2D game environment (of 500 units × 500 units). The movement of the bot, in response to sensory input, is governed by motor output from the SNN that controls it. Its sensory input is received through its field of view as illustrated in Fig. 1. The field of view is segmented into 9 parts; 3 radial ranges, and 3 angular ranges. The 3 radial ranges

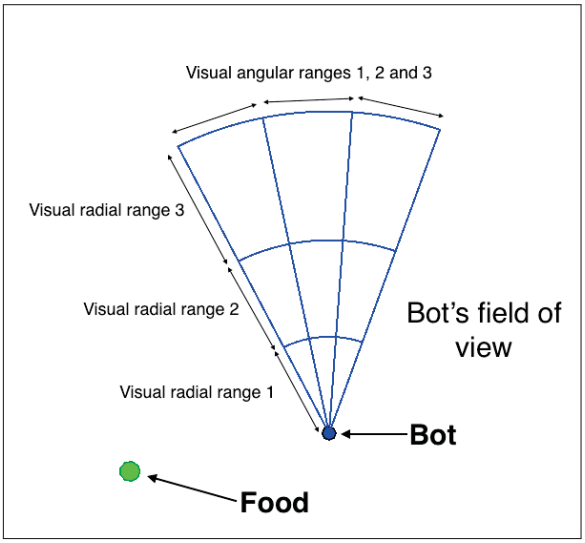

Figure 1: Graphical representation of a bot. The circular dot represents its areal extent in the game environment. The blue quadrant represents its field of view that is segmented as described in Section 2.1. Each visual segment activates a different combination of the sensory neurons of its SNN.

extend from 0 - 30, 30 - 60, and 60 - 100 units. The presence of food within the field of view triggers a different neuron for each of the radial ranges. The opening angle of the field of view, v, is considered a morphological parameter of the bot and is allowed to evolve along with its SNN. The angle is trisected for 3 angular ranges and the presence of food within each of them triggers a different sensory neuron. Thus, a total of 6 sensory neurons are dedicated for the bot's vision.

Four motor neurons control the movement of the bot. The first one, when fired, advances the bot by 1 unit in its orientation direction. The second makes the bot take 1 step back. The third and the fourth rotate the bot clockwise and anti-clockwise by 0.1 radians, respectively.

2.2 The Spiking Neural Network

Each bot has a SNN that controls it. Each SNN consists of 30 neurons in a fully-connected, directed network, as illustrated in Fig. 2. The edges of the network are associated with weights w_{ij}, and this matrix is allowed to evolve. Neurons in this network do not connect to themselves, hence $w_{ii} = 0$. Of the neurons, 6 are sensory and 4 are motor as has been described. The SNN operates in discrete time steps that also correspond to time steps in the motion of the bot. Each neuron has a membrane potential $V(t)$ whose dynamics is governed by the Leaky Integrate and Fire (LIF) model [10]. The LIF model may be described by

$$\frac{dV(t)}{dt} = \frac{1}{C_m}\frac{dq}{dt} - \frac{V(t)}{R_m C_m} \tag{1}$$

where dq/dt is the input current, C_m is a measure of the neuron's membrane capacitance, and R_m is its membrane resistance. The first term expresses the increase in membrane potential from the

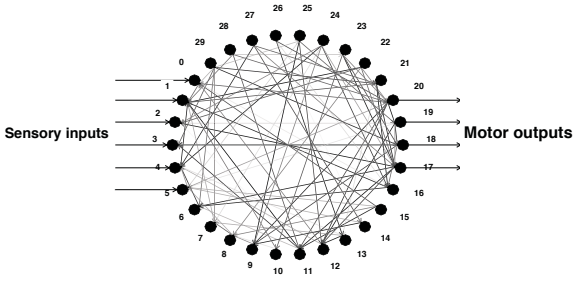

Figure 2: The structure of a SNN that controls a single bot shown at a representative state in its evolution. It consists of 30 spiking neurons in a directed network. The shade of the edges correspond to their weights at a particular generation. 6 neurons are connected to the sensory inputs of the bot and 4 to its motor output.

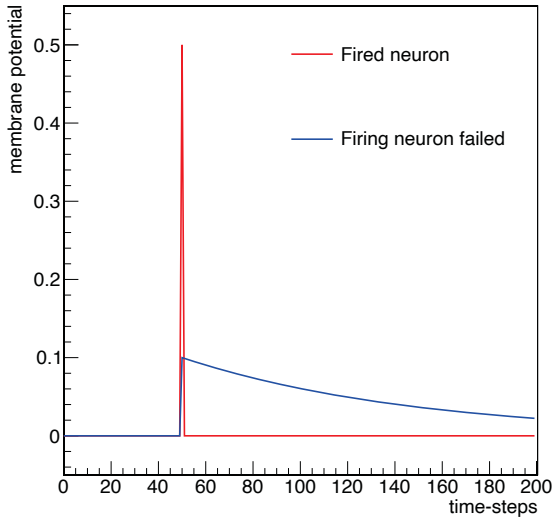

Figure 3: The membrane potential of a single simulated neuron as a function of time-steps when fired with two values of incoming charge. In red is when the incoming charge corresponds to an increase in potential that exceeds the threshold V_{th}, and in blue is when it does not.

rate of charge deposition from incoming spikes. The second term reflects the decay of membrane potential due to the spontaneous neutralization of charge. In our model, we approximate LIF in the limit of infinitesimal time-steps using the difference equation:

$$V(t+1) - V(t) = q(t) - \beta V(t) \tag{2}$$

where β contains the $R_m C_m$ decay constant and is set to 1%. The capacitance is set to unity in our simulation with no loss of generality as the scale of V is set by the voltage threshold V_{th} beyond which the neuron fires.

The incoming charge for neuron i at time-step t is given by the the sum of arriving spikes weighted by w_{ij}

$$q_i(t) = \sum_j w_{ij} A_j(t) \tag{3}$$

where $A_j(t)$ is 1 if the j^{th} neuron has fired in time-step t and 0 otherwise.

A neuron fires if its membrane voltage exceeds the threshold $V_{th} = 0.4$ or randomly at a spontaneous rate of $b \approx 1\%$. The spontaneous rate, which corresponds loosely to the bias term for each neuron in traditional neural networks, is found to be important to avoid trapping the SNN in states where no neurons are firing. This spontaneous firing rate, b, is allowed to evolve. Thus, for the i^{th} neuron at time t,

$$A_i(t) = \begin{cases} 1 & \text{if } V_i(t) > V_{th} \text{ OR } r > b \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where r is a uniform random number from 0 to 1. When the neuron fires, the membrane potential V is set back to 0 at the next time-step. We illustrate the firing behavior of a single simulated neuron in Fig. 3 by plotting its membrane potential by time-step when fired with an incoming charge corresponding to potential increases greater and lesser than V_{th}.

2.3 The Environment

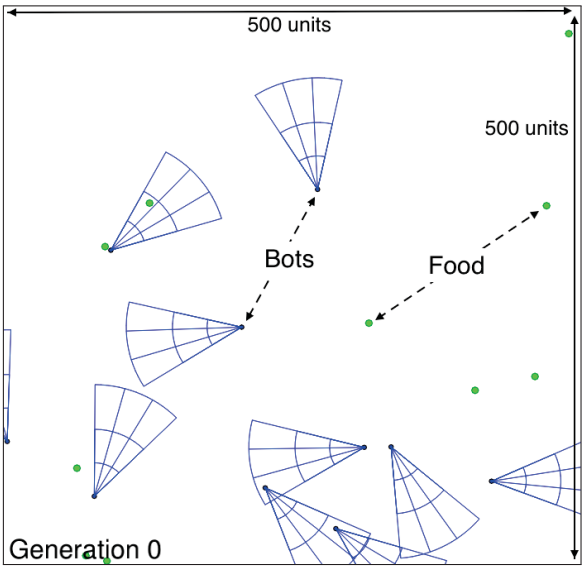

Figure 4: Snapshot of the multi-agent environment within which the bots and their SNNs evolve. The physical space is 500 units x 500 units, and is populated here with 10 bots and 10 pieces of food, all in constant motion. The walls are reflective, as described in the text. The rules of evolution implemented by the environment are described in Section 3.

The evolutionary environment in which our bots operate is a 2D square of 500 units × 500 units, as shown in Fig. 4. The walls are reflective, i.e. when bots run into the vertical walls their θ is

changed to $\pi - \theta$, and when they run into horizontal walls their θ is multiplied by -1.

The environment contains entities that result in the reproduction of a bot if captured. We call these entities "food" for the remainder of the paper. Like the bots, they each have (x, y) coordinates, a fixed orientation angle θ, and a randomly chosen speed. A capture occurs when the square of the Euclidean distance between a food and a bot, $(x_{bot} - x_{food})^2 + (y_{bot} - y_{food})^2$, is less than 13. The procedures implemented in the reproduction of the bots at each capture is described in Section 3. The food is replenished in the environment by placing a new instance in a random position and orientation with a randomly chosen speed.

3 EVOLUTIONARY ALGORITHMS

Two evolutionary inheritance algorithms are investigated in this paper: we call the first one "Mutation" and the second one "Crossover with Mutation". In both algorithms, when a capture event occurs as described in Section 2.3, three procedures kick in: a selection procedure, a reproduction procedure, and an elimination procedure. This results in a new "generation" of bots which then continue to compete in the environment till the next capture event. The phenotypical parameters of the bots, i.e., its SNN weight matrix w, the spontaneous firing rate b, and its visual angle v, are initially random. Thus, initially, there is there no correlation between what a bot senses in its field of view and what it does. *No explicit reward* in the form of weights reinforcement is given to an individual bot when it captures food. The successful bot(s) are reproduced with purely random mutations, depending on the inheritance algorithm, and bot(s) eliminated according to a fitness function to keep the population constant. With this bare minimum of evolutionary pressure, we expect the SNNs to learn to drive the bots to food with increasing efficiency in the course of a few generations. The fitness function used is

$$f = N/\tau, \tag{5}$$

where N is the number of times it has captured food and τ is its age in time-steps. During elimination, bots with the lowest values of f are removed from memory.

3.1 Mutation

In this inheritance algorithm, the bot that captured food is selected for reproduction. The phenotype of the bot is duplicated with random mutations to create a new bot. Components of the weight matrix w and b are modified with random Gaussian variations of standard deviation σ_{mod}. The visual angle, v, is modified similarly with the parameter σ_{visual}. This is summarized in Algorithm 1. One bot in the population is removed according the fitness function f as described earlier.

3.2 Crossover with Mutation

This inheritance algorithm involves waiting for two bots to capture food and mixing their phenotype parameters to create two child bots. Two bots with the lowest fitness values, f as described in Eq. 5, are then eliminated. There exist a variety of crossover operations on matrices that exist in literature [20]. In our algorithm, the weight matrices of the SNNs of the two bots, w^1 and w^2, are partitioned in half and interchanged as illustrated in Fig. 5. The

Algorithm 1: Evolutionary inheritance algorithm of Mutation

Input: Selected bot has phenotype $(w^{old}, b^{old}, v^{old})$;
Output: New bot made with phenotype $(w^{new}, b^{new}, v^{new})$;
for *each connection (i, j) in w* **do**
  $w_{ij}^{new} \leftarrow w_{ij}^{old} + \mathcal{N}(0, \sigma_{mod}^2)$;
end
$b^{new} \leftarrow b^{old} + \mathcal{N}(0, \sigma_{mod}^2)$;
$v^{new} \leftarrow v^{old} + \mathcal{N}(0, \sigma_{visual}^2)$;

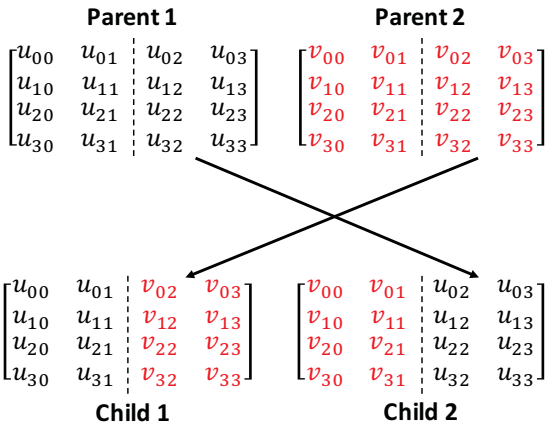

Figure 5: Illustration of the crossover procedure that mixes the SNN weights of two bots for the "Crossover and Mutation" strategy described in Section 3.2. While the illustration is with 4×4 matrices, the SNN weight matrices are 30×30.

new weight matrices, spontaneous rate and visual angle are then mutated in exactly the same way and with the same parameters σ_{mod} and σ_{visual} as described in Section 3.1. This is summarized in Algorithm 2.

4 EVALUATIONS

We evaluate the two evolutionary algorithms by measuring the average number of time-steps, T, needed by a bot to capture food at each generation. For our analysis, since Crossover with Mutation requires 2 bots to capture food to advance a generation, we consider the time taken for 2 consecutive captures in the definition of T for both strategies. Thus, we define T as

$$T = \langle t_2 - t_1 \rangle_{50 \text{ generations}}, \tag{6}$$

where t_1 is the time-step at which piece of food is captured by a bot, and t_2 is the time-step at which another piece of food has been captured by any other bot and then yet another piece captured by any bot. This quantity is averaged over 50 generations and studied. As the SNNs learn, this is expected to decrease with the number of generations. Since this is evolutionary learning, we also expect features of punctuated equilibria which we fit to the logistic function.

Algorithm 2: Evolutionary inheritance algorithm of Crossover with Mutation.

Input: Selected bots have phenotypes (w^1, b^1, v^1), (w^2, b^2, v^2);
Output: New bots made with phenotypes (w^3, b^3, v^3), (w^4, b^4, v^4);
$w^3 \leftarrow w^1$;
$w^4 \leftarrow w^2$;
for *each connection (i,j) in* w^1 **do**
  if $j > k/2$ **then**
    $w^4_{ij} \leftarrow w^1_{ij}$;
    $w^3_{ij} \leftarrow w^2_{ij}$;
  end
  $w^3_{ij} \leftarrow w^3_{ij} + \mathcal{N}(0, \sigma^2_{mod})$;
  $w^4_{ij} \leftarrow w^4_{ij} + \mathcal{N}(0, \sigma^2_{mod})$;
end
$b^3 \leftarrow b^1 + \mathcal{N}(0, \sigma^2_{mod})$;
$b^4 \leftarrow b^2 + \mathcal{N}(0, \sigma^2_{mod})$;
$v^3 \leftarrow v^1 + \mathcal{N}(0, \sigma^2_{visual})$;
$v^4 \leftarrow v^2 + \mathcal{N}(0, \sigma^2_{visual})$;

Experiments for this paper are conducted in the previously described 500 units × 500 units environment with 10 bots and 5 pieces of food. Each bot is controlled by a SNN. The global mutation parameters, σ_{mod} and σ_{visual} defined in Section 3.1, are set to 2.5×10^{-3} and 6.4×10^{-5}, respectively. We arrived at these values by rough optimization of the final T after 10,000 generations of evolution to obtain a fairly efficient learning environment. Our results, especially in their qualitative features, do not lose generality in the neighborhood of this parameter set.

4.1 One experiment of Mutation

Much can be learned by observing the outcome of one experiment with the Mutation inheritance algorithm. As seen in the **video** accompanying this paper, bots are initially seen to execute random motions with no regard for food in their fields of view. As bots accidentally capture food and reproduction begins, small mutations in a bot's phenotype that make food capture more probable allow that bot to have more offspring. Thus, after roughly 100 generations, food capture becomes less accidental and more apparently intentional as the SNNs structure themselves to make use of sensory data from their field of view. Around generation 1,400, we observe the development of hunting behavior as the bots learn to cover ground and spin their fields of view in search of food. Bots that do not hunt have lower fitness values f and are eventually culled. This development results in a rapid improvement in efficiency and decrease in T, as defined in Eq. 6.

In Fig. 6, we study the variation of T as a function of generation up to 10,000 generations. While there is a large variance in T initially, as the population of bots branches into lineages that sometimes work well and sometimes do not, we note a sharp drop around generation 1,214 when a bot discovers hunting. Thereafter, the bot that discovered hunting dominates the population with its offspring

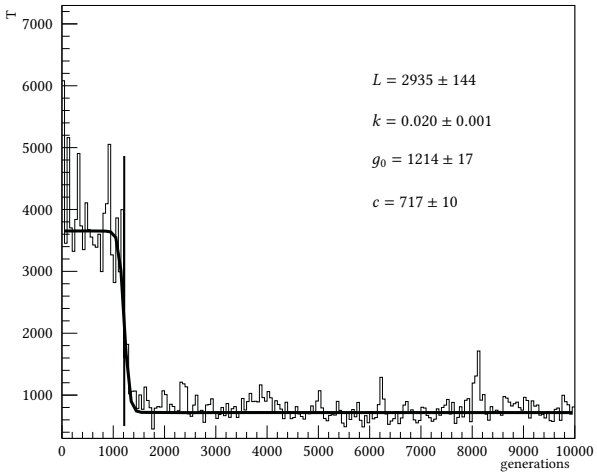

Figure 6: The average time-steps to capture food, T as defined in Eq. 6, as a function of the number of generations using the evolutionary inheritance algorithm of Mutation. A fit to a logistic function is used to extract quantitative features of the punctuated equilibria.

and T remains relatively stable up to 10,000 generations. Thus, we observe two periods of equilibrium connected by a punctuation, as expected in evolutionary systems. We extract broad features of this punctuated equilibria by fitting the graph with a logistic function on a flat pedestal of the form

$$f(g) = \frac{L}{1 + e^{k(g-g_0)}} + c. \tag{7}$$

The center of the punctuation, or the Inflection Point, is given by g_0 in generations. The sharpness of the punctuation is given by the slope of the inflection, k. The final equilibrium value of T is given by c, and we call this the Convergence Point. The initial equilibrium value of T is given by $L + c$. A minimum χ^2 fit returns $g_0 = 1214 \pm 17$ generations, $k = 0.020 \pm 0.001$ time-steps / generation, $L = 2935 \pm 144$ time-steps and $c = 717 \pm 10$ time-steps.

4.2 One experiment of Crossover with Mutation

We repeat the experiment with the inheritance method of Crossover with Mutation and observe similar behavior in the bots as they learn to capture food. Faster learning is observed as hunting behavior emerges around generation 469. A minimum χ^2 fit with Eq. 7 returns $g_0 = 469 \pm 33$ generations, $k = 0.020 \pm 0.001$ time-steps / generation, $L = 5164 \pm 669$ time-steps and $c = 780 \pm 10$ time-steps.

4.3 Comparison over experimental ensembles

The trajectories of these experiments and the quantitative features of the punctuated equilibria depend sensitively on the random number generator that dictate the initial phenotypes of the bots and their mutations. Therefore, to establish any significant quantitative difference between the two evolutionary algorithms, a statistical study is performed. The experiments with Mutation, and Crossover

Evolutionary Strategy	Inflection Point (generations)	Convergence Point (time-steps)
Mutation	512 ± 68	699 ± 5
Crossover with Mutation	300 ± 9	759 ± 25 and 1967 ± 31

Table 1: Summary of mean Inflection Point and Convergence Point for the evolutionary strategies of Mutation and Crossover with Mutation.

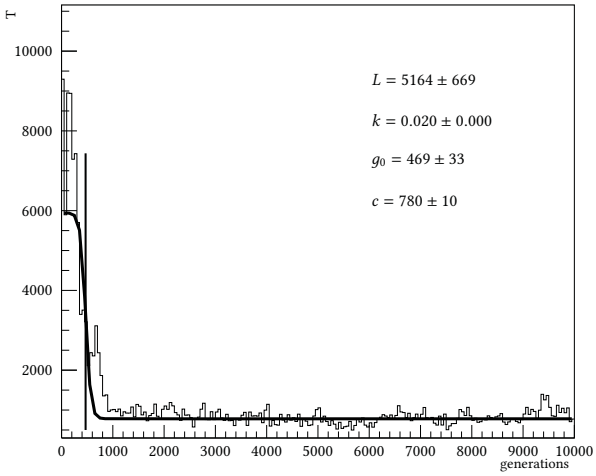

Figure 7: The average time-steps to capture food, T, as a function of the number of generations using the evolutionary strategy of Crossover and Mutation. A fit to a logistic function is used to extract quantitative features of the punctuated equilibria.

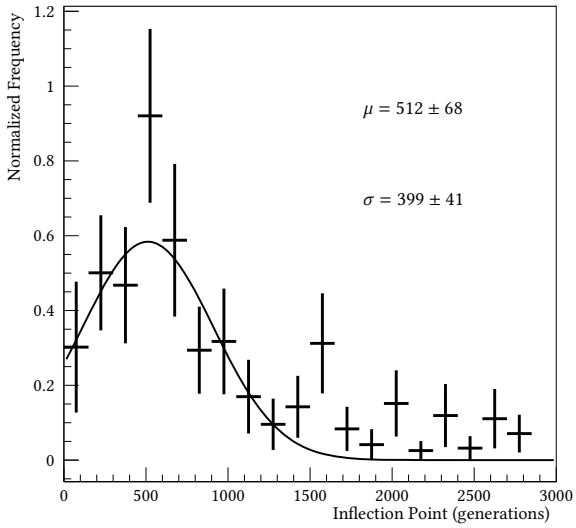

Figure 8: Distribution of Inflection Points (in generations) in an ensemble of 100 experiments with the evolutionary strategy of Mutation. The histogram is binned by 150 generations and fitted with a Gaussian to estimate its mean and standard deviation.

with Mutation are each repeated 100 times with different random number seeds. This results in an ensemble of trajectories for each approach.

One may be naively tempted to consider the average of T at each generation over the 100 trajectories for each ensemble. However, since the inflection happens at a different point in each experiment, such averaging would result in a soft falling curve and would thus lose information on where the inflections occur. To avoid this, we fit each of the 100 trajectories with the logistic function, extract the g_0 and c, and plot their distributions for comparison between the two evolutionary strategies.

Fig. 8 and 9 show the distributions of the Inflection Points in the 100 experiment ensembles for the inheritance algorithms of Mutation, and Crossover with Mutation, respectively. They are both fitted with Gaussians to extract the means and standard deviations of these distributions. We note that while Mutation inflects at 512 ± 68 generations, Crossover with Mutation inflects significantly earlier at 300 ± 9 generations. Thus, one may say Crossover with Mutation results in 40% faster learning than just Mutation.

Fig. 10 and 11 show the distributions of the Convergence Points for Mutation, and Crossover with Mutation, respectively. While the distribution for Mutation may be fitted to a simple Gaussian with mean at 699 ± 5 time-steps, the distribution for Crossover with Mutation is clearly bi-modal. We fit the latter with the sum of two Gaussians and find that their means are at 759 ± 25 and 1967 ± 31 time-steps, respectively. This and the lack of a bi-modal distribution in Fig. 9 imply that in a fair fraction of cases, Crossover with Mutation converge to a less-than-optimal solution though it starts the learning process faster. By comparing areas under the two peaks of the bi-modal distribution, we find that fraction to be 29%. We summarize these results in Table 1.

5 CONCLUSIONS

Spiking neural networks are the third generation of neural networks. They allow for encoding information in the temporal sequence of spikes and thus offer higher energy efficiency. Further, the sparseness of spikes make them energy efficient and thus appropriate for neuromorphic applications. However, training them requires novel methods. In this paper, we have demonstrated a multi-agent ER based framework inspired by evolutionary rules and competitive intelligence to train SNNs for performing a task efficiently.

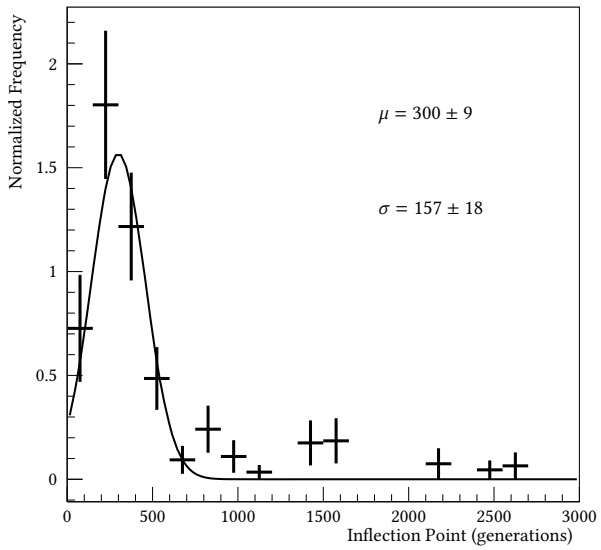

Figure 9: Distribution of Inflection Points (in generations) in an ensemble of 100 experiments with the evolutionary strategy of Crossover and Mutation. The histogram is binned and fitted identically to Fig. 8

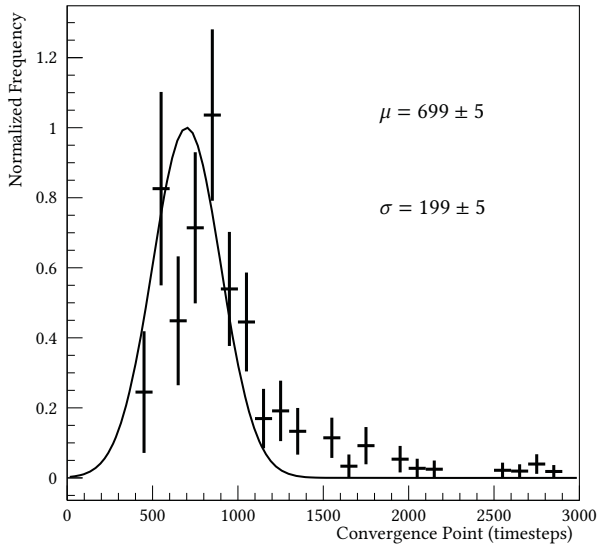

Figure 10: Distribution of Convergence Points (in time-steps) in an ensemble of 100 experiments with the evolutionary strategy of Mutation. The histogram is binned by 100 time-steps and fitted with a Gaussian to estimate its mean and standard deviation.

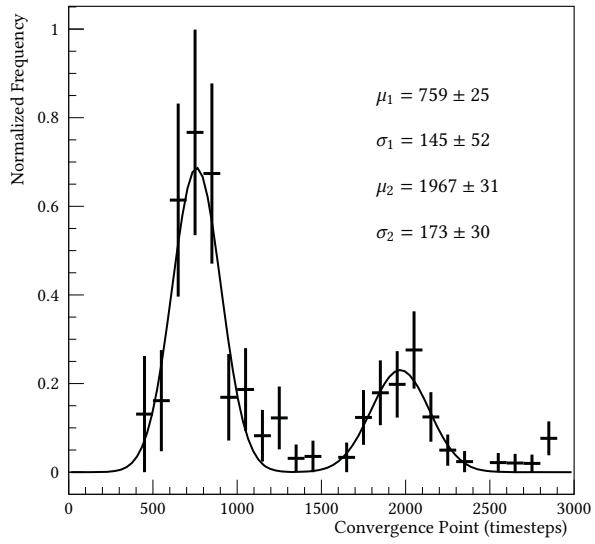

Figure 11: Distribution of Convergence Points (in time-steps) in an ensemble of 100 experiments with the evolutionary strategy of Crossover and Mutation. The histogram is binned by 100 time-steps. It is bi-modal and fitted with two Gaussians. Their means and standard deviations are reported.

Two evolutionary inheritance algorithms, Mutation and Crossover with Mutation, are demonstrated and their respective performances are compared over statistical ensembles. We find that Crossover with Mutation promotes 40% faster learning in the SNN than mere Mutation with a statistically significant margin. We also note that Crossover with Mutation results in 29% of experiments converging to a less-than-optimal solution.

Future directions of this work may lead to the integration of evolutionary approaches with in-lifetime learning models like reinforcement learning. Specifically, evolutionary approaches could be used to generate the reward mechanism for reinforcement learning models or learning models that use Spike Timing Dependent Plasticity. Furthermore, it may be interesting to explore the network topology as a function of evolutionary generations.

REFERENCES

[1] Wyeth Bair and Christof Koch. 1996. Temporal Precision of Spike Trains in Extrastriate Cortex of the Behaving Macaque Monkey. *Neural Computation* 8, 6 (1996), 1185–1202. https://doi.org/10.1162/neco.1996.8.6.1185

[2] Sander Bohte, Joost Kok, and Han Poutré. 2001. Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. *Neurocomputing* 48 (02 2001), 17–37. https://doi.org/10.1016/S0925-2312(01)00658-0

[3] Stephane Doncieux, Nicolas Bredeche, Jean Baptiste Mouret, and Agoston E. (Gusz) Eiben. 2015. Evolutionary robotics: What, why, and where to. *Frontiers Robotics AI* 2, MAR (2015), 1–18. https://doi.org/10.3389/frobt.2015.00004

[4] Maria Jose Escobar, Guillaume S. Masson, Thierry Vieville, and Pierre Kornprobst. 2009. Action recognition using a bio-inspired feedforward spiking network. *International Journal of Computer Vision* 82, 3 (2009), 284–301. https://doi.org/10.1007/s11263-008-0201-1

[5] Samanwoy Ghosh-Dastidar and Hojjat Adeli. 2007. Improved spiking neural networks for EEG classification and epilepsy and seizure detection. *Integrated Computer-Aided Engineering* 14 (2007), 187–212. https://doi.org/10.3233/ICA-2007-14301

[6] Ankur Gupta and Lyle N. Long. 2007. Character recognition using spiking neural networks. *IEEE International Conference on Neural Networks - Conference Proceedings* (2007), 53–58. https://doi.org/10.1109/IJCNN.2007.4370930

[7] Matthew Hartley, Neill Taylor, and John Taylor. 2006. Understanding spike-time-dependent plasticity: A biologically motivated computational model. *Neurocomputing* 69, 16-18 (2006), 2005–2016. https://doi.org/10.1016/j.neucom.2005.11.021

[8] Roger Herikstad, Jonathan Baker, Jean Philippe Lachaux, Charles M. Gray, and Shih Cheng Yen. 2011. Natural movies evoke spike trains with low spike time variability in cat primary visual cortex. *Journal of Neuroscience* 31, 44 (2011), 15844–15860. https://doi.org/10.1523/JNEUROSCI.5153-10.2011

[9] Nikola Kasabov, Valery Feigin, Zeng Guang Hou, Yixiong Chen, Linda Liang, Rita Krishnamurthi, Muhaini Othman, and Priya Parmar. 2014. Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke. *Neurocomputing* 134 (2014), 269–279. https://doi.org/10.1016/j.neucom.2013.09.049

[10] Nikola K Kasabov. 2018. *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence (Springer Series on Bio- and Neurosystems).* 1–742 pages. http://www.springer.com/series/15821

[11] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. 2020. Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures. *Frontiers in Neuroscience* 14, February (2020), 1–22. https://doi.org/10.3389/fnins.2020.00119

[12] Wolfgang Maass. 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Networks* 10, 9 (1997), 1659–1671. https://doi.org/10.1016/S0893-6080(97)00011-7

[13] Zachary F. Mainen and Terrence J. Seinowski. 1995. Reliability of spike timing in neocortical neurons. *Science* 268, 5216 (1995), 1503–1506. https://doi.org/10.1126/science.7770778

[14] B. Meftah, O. Lezoray, and A. Benyettou. 2010. Segmentation and edge detection based on spiking neural network model. *Neural Processing Letters* 32, 2 (2010), 131–146. https://doi.org/10.1007/s11063-010-9149-6

[15] Ammar Mohemmed, Stefan Schliebs, Satoshi Matsuda, and Nikola Kasabov. 2012. Span: Spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems* 22, 4 (2012). https://doi.org/10.1142/S0129065712500128

[16] Filip Ponulak and Andrzej Kasiński. 2010. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. (2 2010), 467–510 pages. https://doi.org/10.1162/neco.2009.11-08-901

[17] Filip Ponulak and Andrzej Kasiński. 2011. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiologiae Experimentalis* 71, 4 (2011), 409–433.

[18] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2017. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv* (2017).

[19] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. 2019. Deep learning in spiking neural networks. *Neural Networks* 111 (2019), 47–63. https://doi.org/10.1016/j.neunet.2018.12.002

[20] Ming Wen Tsai, Tzung Pei Hong, and Woo Tsong Lin. 2015. A two-dimensional genetic algorithm and its application to aircraft scheduling problem. *Mathematical Problems in Engineering* 2015 (2015). https://doi.org/10.1155/2015/906305

[21] Simei Gomes Wysoski, Lubica Benuskova, and Nikola Kasabov. 2010. Evolving spiking neural networks for audiovisual information processing. *Neural Networks* 23, 7 (2010), 819–835. https://doi.org/10.1016/j.neunet.2010.04.009