Exploring Plasticity-Driven SNNs for Enhanced ML

S. Labady, T. Manos, C. Jimenez-Romero, M. Quoy
ETIS UMR8051, CY Cergy Paris University / ENSEA / CNRS
6 avenue du Ponceau, 95014 Cergy-Pontoise Cedex, France

***Abstract*—This study explores the optimization of SNNs using GAs to address the MC problem, a benchmark challenge in RL. The focus is on how GAs can navigate the high-dimensional parameter space of SNNs, enhancing their learning efficiency and adaptability. Employing a meta-optimization approach, L2L, the study systematically adjusts SNN parameters, demonstrating the potential of combining GA with biologically-inspired neural models. Additionally, the incorporation of synaptic plasticity into SNNs represents a significant advancement, aiming to capture more accurately the dynamic nature of biological learning processes. Through experiments, this study showcases the effectiveness of optimized SNNs, both static and plastic, in solving the MC problem, emphasizing the importance of synaptic plasticity. The results highlight the feasibility and benefits of using SNNs and GAs in complex ML tasks, opening new avenues for the development of adaptive and efficient computational models.**

***Keywords*—Spiking Neural Networks (SNNs), Genetic Algorithms (GAs), Reinforcement Learning (RL), Spike-Timing-Dependent Plasticity (STDP), Learning to learn (L2L), Mountain Car (MC), Machine Learning (ML), Artificial Neural Networks (ANNs), Leaky Integrate-and-Fire (LIF), Neural Simulation Tool (NEST)**

I. Introduction

The intersection of computational neuroscience and ML offers fertile ground for exploring models that not only mimic the brain's functionality but also harness its computational paradigms for solving complex problems. Among these models, SNNs stand out for their ability to replicate the temporal dynamics of biological neural networks, offering a promising avenue for the development of more efficient and adaptive computational algorithms. This study focuses on the optimization of SNNs using GAs to address a well-known benchmark problem in RL : the MC problem.

The MC problem, despite its apparent simplicity, poses significant challenges in the domain of RL due to its requirement for strategic planning and efficient exploration of the environment. It serves as a crucial testbed for evaluating the efficacy of learning algorithms in navigating complex state spaces with minimal a priori knowledge.

Recent advancements in GAs, characterized by their robustness in exploring complex and high-dimensional search spaces, present a novel approach to optimizing the parameters of SNNs. GAs, inspired by the principles of natural selection and genetics, offer a powerful tool for systematically adjusting the parameters and structures of SNNs to enhance their learning capabilities. This study leverages these algorithms to explore the potential of SNNs in solving the MC challenge, highlighting the efficiency and adaptability of this approach.

Furthermore, the integration of synaptic plasticity into SNNs represents a significant step forward in modeling the dynamic learning processes observed in biological neural networks. By enabling synapses to adapt their strengths in response to environmental stimuli, plasticity allows for more flexible and potent learning mechanisms.

In summary, this study contributes to the ongoing discourse in computational neuroscience and machine learning by:

1) Demonstrating the applicability of GAs for optimizing SNNs in solving RL problems.
2) Exploring the role of synaptic plasticity in enhancing the learning efficiency and adaptability of SNNs.
3) Evaluating the potential of SNNs, optimized through GAs, in addressing the challenges posed by the MC.

II. State of the art

SNNs, distinct from traditional ANNs, mimic the biological processes of the human brain. The dynamic and temporal nature of SNNs allows for a more nuanced processing of information, which is particularly beneficial for tasks that involve real-time decision-making and complex temporal dynamics. SNNs use spikes, similar to how real brain neurons communicate. They can learn through a process called STDP, which changes the strength of connections based on the timing of spikes. This helps SNNs adapt and learn from patterns in activity over time.

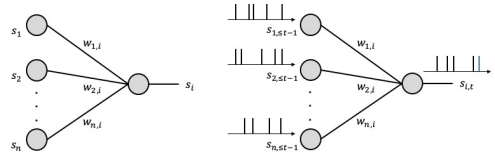

Fig. 1. Illustration of neural networks: (left) an ANN, where each neuron processes real numbers; and (right) an SNN, where dynamic spiking neurons process and communicate binary sparse spiking signals over time.

A. Neuronal Dynamics Key Concepts

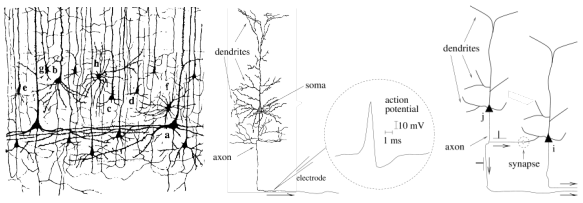

Fig. 2. This reproduction of a drawing of Ramón y Cajal shows a few neurons in the mammalian cortex that he observed under the microscope. [1]

The exploration of fundamental neuronal concepts such as action potentials, synaptic activities, and firing thresholds,by

Gerstner et al., 2014 [1], is foundational. These concepts provide a basis for understanding the intricate workings of neurons, crucial for the development of our neural network models. The detailed descriptions of how neurons process and transmit information, along with the mathematical modeling of these processes, offer a solid groundwork.

The theoretical insights and methodologies presented in the book "Neuronal Dynamics" 2014 [1] are instrumental. They do not only enhance our understanding of the intricate details of neuronal systems but also provide a robust framework for developing sophisticated and biologically inspired neural network models. The integration of these concepts into our project paves the way for innovative advancements in neural network modeling and simulation, propelling our research into new realms of computational neuroscience.

B. Evolutionary and spike-timing-dependent RL train spiking neuronal network motor control

In Haşegan et al. 2022 [2], the authors presented a novel approach to enhancing the performance of SNNs models. The focus was on comparing and analyzing the effectiveness of two biologically inspired learning mechanisms: Spike-Timing-Dependent Reinforcement Learning (STDP-RL) and Evolutionary strategy (EVOL). The study addressed the gap between ANNs and SNNs by exploring the potential of STDP-RL and EVOL in optimizing SNNs. The researchers employed these mechanisms on SNNs to solve the RL control problem, a task where a pole must be balanced on a moving cart. This study aimed to recreate accurate models that capture tinteraction. One advantage of EVOL is its ability to potentially bypass the need to capture all interacting components of synaptic plasticity, making it a viable alternative to STDP-RL. The researchers compared the performance of each algorithm after training. The results revealed that EVOL emerged as a powerful method for training SNNs to perform sensory-motor behaviors. This comparison opens new capabilities for SNNs in RL and serves as a testbed for neurobiologists aiming to understand multi-timescale learning mechanisms and dynamics in neuronal circuits.

The CartPole environment requires applying force to either side of a cart to maintain the balance of a vertically positioned pole. The model's task was to keep the pole balanced, with the environment fully described by four observations: cart position, cart velocity, pole angle, and pole angular velocity. For simulations, the NEURON environment [3] with the NetPyNE package was used, allowing the integration of the CartPole environment with the network simulation. The findings from this research highlight the potential of using evolutionary strategies in training biologically realistic SNNs for complex tasks. This approach not only enhanced the performance of these models but also contributes to a deeper understanding of the learning processes in biological neuronal networks.

The EVOL method has shown its effectiveness in the CartPole environment, it could significantly benefit our project. EVOL's success in optimizing SNNs for dynamic tasks aligns well with the challenges of the MC scenario. This environment requires strategic timing and decision-making to navigate a car up a steep hill, similar to the precision needed in CartPole. By applying EVOL, we can enhance our SNNs' ability to learn and adapt in real-time, crucial for mastering other RL tasks. EVOL's approach to exploring diverse network architectures will allow us to identify optimal strategies for acceleration and momentum-building, key elements in the MC challenge.

C. Exploring Parameter and Hyper-Parameter Spaces of Neuroscience Models on High Performance Computers With Learning to Learn

Yegenoglu et al. 2022 [4] proposed a feed-forward network SNN of LIF neurons to solve the MC Challenge. The inputs to the SNN are the position and velocity of the car, discretized and encoded using 30 input neurons for each variable. The SNN is implemented in NEST [5] and consists of an encoding layer, an intermediate layer, and an output layer, with the output layer neurons corresponding to the possible actions in the MC task. The action sent to the environment is determined by the neuron with the highest spiking activity in the output layer. The fitness metric for optimization is defined as the maximum horizontal position reached by the car during an episode of 110 simulation steps. The optimization process used a GA. This algorithm optimized the weights of the SNN, and the optimized parameters are then used to initialize the next generation of individuals. The analysis of the SNN over 400 generations showed that the fitness becomes positive after 50 generations, indicating that the car is moving towards the goal position. The best solution (goal position of 0.5) was first achieved around generation 160. In subsequent generations, while the mean fitness saturated at around 0.3, the best fitness reached the maximum of 0.5. Upon completion of 400 generations, the best individual fitness being 0.5 was recorded, confirming the successful application of the SNN in solving the MC problem. It was found that the network, on average, required 101 simulation steps to reach the goal position, thus effectively solving the task. This performance compares favorably with other machine learning techniques applied to this problem, demonstrating the effectiveness of SNNs combined with GAs in RL task. It offers a viable alternative to traditional RL and gradient descent methods. Despite the promising outcomes demonstrated, the complexity and computational demands of training SNNs, particularly when integrated with evolutionary strategies like GA, pose significant challenges. Furthermore, the scalability of this approach to more complex or varied tasks remains an open question. While the study successfully applied the method to the MC problem, the generalizability of this approach to other scenarios with different dynamics and complexity is not yet clear.

D. Supervised Learning in SNN via Reward-Modulated Spike-Timing-Dependent Plasticity for a Target Reaching Vehicle

In Bing et al. 2019 [6], the authors delved into SNNs and their application in robotic control. The significance of this study is an innovative approach to training SNNs using

a method called Reward-Modulated Spike-Timing-Dependent Plasticity (R-STDP). This approach marked a departure from traditional neural network training methods, offering a more biologically realistic model that mirrors the functioning of the human brain. The implementation of R-STDP in SNNs is crucial as it enables the network to learn from a reward mechanism, which is critical for tasks that involve decision-making or adaptive responses to the environment.

R-STDP extends the STDP learning rule by incorporating a reward signal into the learning process. This means that the synaptic adjustments in response to the spike timing are scaled by how well the network's output aligns with a desired outcome or goal. R-STDP essentially combines the temporal learning ability of STDP with a RL aspect, allowing the network to not only learn patterns but also to optimize its behavior towards achieving specific goals or tasks.

The end-to-end learning approach adopted in their study involved training SNNs using pre-generated datasets to guide the robot's behavior. This method of training can be particularly beneficial for our project as it provided a framework for designing SNNs that can learn and adapt to perform specific tasks efficiently. It addressed some of the challenges associated with implementing R-STDP in robotic applications. These challenges include the design of a unified learning paradigm that can be applied to various tasks and the complexity of defining appropriate rewards. The research not only proposed solutions to these challenges but also demonstrates the feasibility of the approach through practical experiments.

E. Reinforcement Learning With Reward-Modulated STDP

A study conducted by Wunderlich et al. 2019 [7] presented a comprehensive examination of the benefits offered by neuromorphic computing, specifically through a pilot study utilizing a single-chip prototype of the BrainScaleS 2 neuromorphic system. This research underscored the potential of neuromorphic devices in mimicking the brain's architecture and dynamics, aiming to replicate its computational power, robust learning capabilities, and energy efficiency.

In their exploration, the authors employed a SNN that learned to play a simplified version of the Pong video game, achieving this through RL with R-STDP. This approach not only demonstrated the neuromorphic system's ability to facilitate complex learning tasks but also highlighted the efficiency gains in terms of speed and power consumption. Compared to equivalent software simulations, the neuromorphic emulation showcased at least an order of magnitude faster performance and three orders of magnitude higher energy efficiency.

In their innovative approach, R-STDP is used as the learning mechanism for training a SNN to play a simplified version of Pong. In the context of the Pong game, the SNN controlled the paddle, with the objective to hit the ball by predicting its trajectory. The reward signal was structured to reflect the paddle's success in hitting the ball, effectively guiding the network towards improving its performance over time. This study offers promising insights into addressing RL challenges.

F. Structure of the Paper

Following this, we delve into the methodology, where we describe our approach to applying GAs for optimizing SNNs within the context of the MC problem. We then present our experimental results, showcasing the effectiveness of our methods. Finally, we conclude with a discussion on the implications of our findings and propose directions for future research.

III. Methodology

L2L [8] for SNNs is a meta-optimization approach that leverages high-performance computing to explore high-dimensional parameter spaces efficiently. It utilizes an other learning method, for example the GAs [9] , to find optimal parameters for the SNNs.

NEST plays a pivotal role in our study as the chosen simulation software for SNNs. It is a highly regarded simulator in computational neuroscience for its precise handling of neural dynamics and large-scale network architectures. With its rich library of neuron models and synapse dynamics, NEST enables the detailed simulation of neuronal activity and synaptic transmission, crucial for investigating the complex interplay of neural components in SNNs.

The MC environment [10], a benchmark in RL, involves a car in a valley, requiring strategic accelerations to reach a goal atop a hill. This deterministic environment, crucial in evaluating learning algorithms, features a car initially at the valley's bottom, where the agent must learn to navigate to the goal efficiently. Gym's discrete action version is used here.

To accommodate the increased computational demands of this more sophisticated neural networks, we leveraged the robust capabilities of the AWS EC2 m5ad.24xlarge instance, which boasts 96 vCPUs, 384 GiB of memory. This choice of hardware facilitated a more efficient exploration of the parameter space and allowed for the parallel execution of multiple simulations of the environment, significantly accelerating the optimization process.

We employ the *iaf_psc_alpha* [5] model, which is a leaky integrate-and-fire (LIF) neuron with alpha-shaped postsynaptic currents. This model is well-suited for simulating the dynamics of biological neurons, with its simplicity allowing for efficient computation while still capturing key aspects of neuronal behavior. It is described by the following differential equation, which governs the evolution of the membrane potential V_m:

$$\frac{dV_m}{dt} = \frac{V_m - E_L}{\tau_m} + \frac{I_{syn} + I_e}{C_m} \tag{1}$$

Here, E_L represents the resting membrane potential, τ_m the membrane time constant, C_m the membrane capacitance, I_{syn} the synaptic input current, and I_e a constant external input current.

The neuron model includes various parameters, such as the membrane capacitance C_m, membrane time constant τ_m, and others listed in the Table I:

The synaptic input current I_{syn} is composed of both excitatory and inhibitory components, given by the sum of alpha functions, which are modeled as Equation 2.

Parameter	Unit	Description	Value
V_{th}	mV	Spike threshold	-55
E_L	mV	Resting membrane potential	-70
C_m	pF	Membrane capacitance	250
τ_m	ms	Membrane time constant	10.0
τ_{ref}	ms	Refractory period duration	1.0
V_{reset}	mV	Reset potential	-70
τ_{syn_ex}	ms	Excitatory synaptic time constant	2.0
τ_{syn_in}	ms	Inhibitory synaptic time constant	2.0
I_e	pA	Constant input current	—

TABLE I
PARAMETERS OF THE *iaf_psc_alpha* [5] NEURON MODEL.

$$i_{syn,X}(t) = \sum_j W_{ij}\alpha_X(t - t_j^d) \tag{2}$$

where $\alpha_X(t)$ is an alpha function representing the postsynaptic current induced by a spike at time t, W_{ij} is the weight of the connection from neuron j to i, and t_j^d is the time of the spike from neuron j after a delay d.

A spike is emitted once the membrane potential crosses the spike threshold V_{th}, and after the spike, the membrane potential is reset to V_{reset} for the duration of the refractory period τ_{ref}.

The *iaf_psc_alpha* [5] model captures the essence of neuronal spiking and allows for the integration of diverse synaptic inputs. Its parameterization provides flexibility, enabling the model to be tailored to match specific neuronal properties for various simulation scenarios.

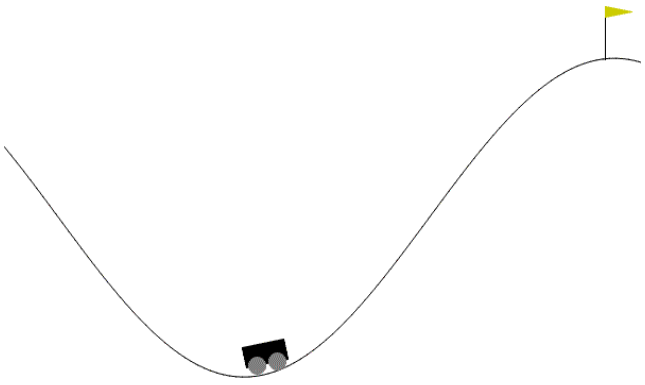

Fig. 3. The MC Environment used for demonstrating the capabilities of the enhanced SNN models.Figure Edited From [10]

Parameter	Value
Action Space	Discrete(3)
Observation Shape	(2,)
Observation High	[0.6 0.07]
Observation Low	[-1.2 -0.07]
Import Command	gym.make("MountainCar-v0")

TABLE II
ENVIRONMENT SETUP.

Num	Observation	Min	Max
0	Position (m)	-Inf	Inf
1	Velocity (m/s)	-Inf	Inf

TABLE III
THE OBSERVATION SPACE IS A 2D ARRAY, REPRESENTING THE CAR'S POSITION AND VELOCITY.

Num	Action	Value
0	Accelerate Left	Inf
1	No Acceleration	Inf
2	Accelerate Right	Inf

TABLE IV
THE AGENT HAS THREE DISCRETE ACTIONS: ACCELERATE LEFT, NO ACCELERATION, ACCELERATE RIGHT.

The MC follows this transition dynamics:

$$\text{vel}_{t+1} = \text{vel}_t + (\text{act} - 1) \times \text{force} - \cos(3 \times \text{pos}_t) \times \text{grav} \tag{3}$$

$$\text{position}_{t+1} = \text{position}_t + \text{velocity}_{t+1} \tag{4}$$

Where force = 0.001 and gravity = 0.0025. The collisions at either end are inelastic with the velocity set to 0 upon collision with the wall. The position is clipped to the range [-1.2, 0.6] and velocity is clipped to the range [-0.07, 0.07].

Reward: The goal is to reach the flag placed on top of the right hill as quickly as possible, as such the agent is penalised with a reward of -1 for each timestep.

Starting State: The position of the car is assigned a uniform random value in [-0.6 , -0.4]. The starting velocity of the car is always assigned to 0.

Episode End: The episode ends if either of the following happens: 1- Termination: The position of the car is greater than or equal to 0.5 (the goal position on top of the right hill). 2- Truncation: The length of the episode is 200.

A. Optimizing Static-synapses SNNs with GAs for the MC challenge

Using the Yegenoglu et al. 2022 [4]'s approach, GAs [9] with SNNs is explored, specifically in the context of the MC. The study focuses on how GAs can effectively navigate the high-dimensional parameter space of SNNs, enhancing learning efficiency and adaptability. This approach not only addresses the specific challenges of the MC problem but also sheds light on the broader potential of combining evolutionary algorithms with biologically-inspired neural models in ML tasks. We use The L2L [8] tool, which is rooted in ML, and adapted to optimize various computational models in neuroscience, ranging from single-neuron models to comprehensive simulations of brain dynamics. Here we present an experimental study undertaken to explore the efficacy of the L2L [8] method in optimizing neuroscience models.

In our architecture in Figure 4, The SNN consisted of three layers: an input layer is divided into two segments, with 30 neurons each, dedicated to encoding the car's velocity and

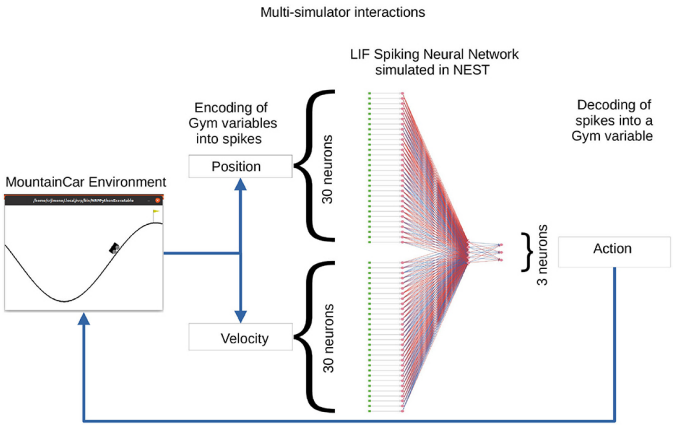

Fig. 4. A feed-forward SNN to solve the MC task. In the MC environment (left) the agent must steer the car to reach the goal position (flag on top of the hill). The position and velocity of the car reported by the MC are encoded into spikes by the encoding layer (DC generators depicted as green dots) of the three-layer SNN (neurons depicted as pink dots and excitatory/inhibitory connections as red/blue lines) running in the NEST (right). The activity from the three output neurons (accelerate left, accelerate right, do nothing) is decoded into actions for MC. The set of weights between the sixty encoding neurons and the three output neurons is the object of optimization. Figure Edited From [4]

position, respectively, an intermediate layer with 5 neurons, and an output layer representing the possible actions with 3 neurons. This decoding process involves monitoring three separate output neurons, each corresponding to a possible action: push left, push none, or push right. The network simulates for a fixed time, and the neuron with the highest spike count determines the action to be taken. The neurons were modeled as leaky integrate-and-fire units, a common choice in SNNs due to their biological plausibility and computational efficiency.

The GA was employed to optimize the synaptic weights of the SNN. The key parameters of the GA included:

- Population size (`32`): Determines the number of individuals in each generation. A larger population size allows for more diverse solutions but increases computational load.
- Crossover probability (`0.7`): The likelihood of two individuals exchanging genetic information, promoting diversity in the population. (`Exploration`)
- Mutation probability (`0.7`): Governs the chance of random alterations in an individual's genetic, introducing variability.($\Delta\text{Mut} \sim \mathcal{N}(\mu = 0, \sigma = 1)$) (`Exploration`)
- Number of generations (`1000`): The total number of iterations for the GA, affecting the convergence of the algorithm.
- Tournament size (`4`): Used in the selection process to choose individuals for reproduction. (`Exploitation`)
- Mating Bend (`0.5`)): Half of each parent genetics is used for crossover operation.

The chosen values for these parameters were based on preliminary trials and literature benchmarks [4]. Specifically, the population size and number of generations were set to ensure sufficient exploration of the solution space, while the crossover and mutation probabilities were balanced to maintain diversity without excessive randomness. During the learning process, each individual in the population represented a unique set of SNN weights. At each generation, the individuals' fitnesses were evaluated based on the performance of their corresponding SNN in the MC task. The GA then applied selection, crossover, and mutation operators to evolve the population towards better solutions. The optimization process led to the emergence of effective synaptic weight configurations, enabling the SNN to successfully solve the challenge.

$$\text{Fitness}(I) = f(\text{paramètres}(I)) = \max_{T}(PET) \qquad (5)$$

The fitness function for the problem is defined as the maximum horizontal position reached by the car during an episode, which spans 110 simulation steps. In the formula $f = \max_T(PET)$, $\max_T$ selects the highest value in a vector, where PET represents the car's position at each simulation step up to $T = 110$. This fitness function evaluates the effectiveness of the car in reaching its farthest forward position, accounting for all simulation steps, thereby providing an overall assessment of performance throughout the episode.

B. Enhancing SNNs with Synaptic Plasticity

Building upon the foundation laid by our initial experiments with static-synapse SNNs and GAs, we furthered our exploration by integrating synaptic plasticity into the network. This addition marks a significant evolution in our approach, aiming to capture the dynamic nature of biological learning processes more accurately.

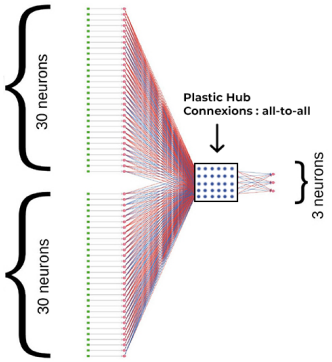

Fig. 5. Schematic representation of the plastic SNN architecture. The network consists of two input layers, each with 30 neurons encoding specific environmental variables. These layers connect to a plastic hub with all-to-all connectivity, symbolized by the square, which processes the inputs and relays information to the 3 output neurons responsible for the decision-making actions.

In this experiment, the optimizer is tasked with determining not only the synaptic weights but also a set of plasticity flags, indicating whether each synapse is plastic (capable of changing its strength) or static. The inclusion of plasticity flags introduces a novel layer of complexity and realism into the SNN model. Synaptic plasticity allows the network to

modify its connections adaptively in response to environmental stimuli. This feature is anticipated to enhance the SNN's ability to solve the MC problem by enabling more nuanced and flexible learning behaviors.

The network in the Figure 5 has an input layer, an intermediate (or hidden) layer, and an output layer. The input layer is divided into two segments, with 30 neurons each, dedicated to encoding the car's velocity and position, respectively. The intermediate layer, also consisting of 30 neurons, serves as the computational hub where the input signals are processed and integrated. The output layer has 3 neurons, each corresponding to one of the possible actions: accelerate left (0), no acceleration (1), or accelerate right (2). All the neurons are modeled using the `iaf_psc_alpha` [5] neuron model in NEST, which is a common choice for simulating the dynamics of biological neurons.

Each neuron in the input layers is connected to all neurons in the intermediate layer. Similarly, the intermediate layer neurons are fully connected to the output neurons. Additionally, our new network has an all-to-all connectivity pattern within the intermediate layer itself. This means that each neuron in the intermediate layer is connected to every other neuron in the same layer. However, we avoid self-connections. This means that while the neurons are extensively interconnected, no neuron is connected to itself. This setup helps to prevent feedback loops that could destabilize the network's dynamics, ensuring that the signal processing remains effective and efficient.

Synapses connecting neurons can be either static or plastic, which is determined by the `plasticity_flag` associated with each synapse. For synapses where this flag is 1, the network employs a plasticity model known as `stdp_synapse` [5] from the NEST simulation library. This STDP model is a form of Hebbian learning where synaptic weights are adjusted based on the relative timing of pre and postsynaptic spikes. This mechanism strengthens synapses if a presynaptic spike precedes a postsynaptic one (potentiation) and weakens them if the order is reversed (depression).

The synapse parameters were carefully chosen to reflect realistic synaptic dynamics and to ensure robust learning behavior. The following are the parameters set in our simulations:

- **Weight** (w): The strength of the synaptic connection, set by the plasticity mechanism.
- **Transmission delay** (d): The time taken for the signal to traverse the synapse, set to 1.0 ms.
- **Learning rate** (λ): Determines the rate at which the synapse can change, set to 0.01.
- **Pre-trace and post-trace time constants** (τ_{plus}, τ_{minus}): Governs the decay of spike traces, both set to 20.0 ms.
- **Potentiation factor** (μ_{plus}): Modulates the weight change during potentiation, set to 1.0.
- **Depression factor** (μ_{minus}): Modulates the weight change during depression, set to 1.0.
- **Maximum synaptic weight** (W_{max}): Defines the upper limit within which the synaptic weight can vary, set to 3000.0.

The STDP model is based on the interaction between the pre- and postsynaptic trace values which decay over time according to their respective time constants. The synaptic weight updates are calculated at each time step using the following Equations 6, 7, 8, 9, 10 :

Pre-synaptic trace update:

$$pre\_trace \leftarrow pre\_trace \times e^{-\frac{dt}{\tau_{tr\_pre}}} \tag{6}$$

Post-synaptic trace update:

$$post\_trace \leftarrow post\_trace \times e^{-\frac{dt}{\tau_{tr\_post}}} \tag{7}$$

Synaptic weight potentiation:

$$w_{pot} = W_{max} \left(\frac{w}{W_{max}} + \lambda \left(1 - \frac{w}{W_{max}} \right)^{\mu_+} pre\_trace \right) \tag{8}$$

Synaptic weight depression:

$$w_{dep} = W_{max} \left(\frac{w}{W_{max}} - \alpha\lambda \left(\frac{w}{W_{max}} \right)^{\mu_-} post\_trace \right) \tag{9}$$

Synaptic weight update:

$$w = \min(\max(W_{min}, w_{dep}), w_{pot}) \tag{10}$$

These equations govern how the synaptic weight is updated in response to pre- and postsynaptic spikes.

For the environment simulation, we allow up to 600 steps, providing the neural network sufficient time to adjust its synapses for optimal learning. For measuring success, we use the car's maximum position as the fitness indicator, directly correlating network performance with its progress in the MC problem.

From there, we have set the population size parameter *pop_size* to 96. This configuration aligns with the hardware's capacity, ensuring that each of the 96 CPUs available can simultaneously process a different individual from the population during the simulation. This efficient use of computational resources means that the time taken for each generation to complete its simulation in the environment is significantly optimized, allowing for a more rapid and effective exploration of the solution space.

IV. Experiments and Results

A. Optimizing Static-synapses SNNs

The Figure 6 presents the evolution of the fitness values across generations of individuals throughout the learning process. The x-axis represents the Generation Index (times 10), while the y-axis denotes the best individual fitness value of each generation. Notably, from the 110th generation, we observe that the fitness value begins to plateau at around 0.5. This indicates a convergence of the GA where further iterations yield minimal improvement in fitness. The plateau suggests that the population has reached a point where the individuals

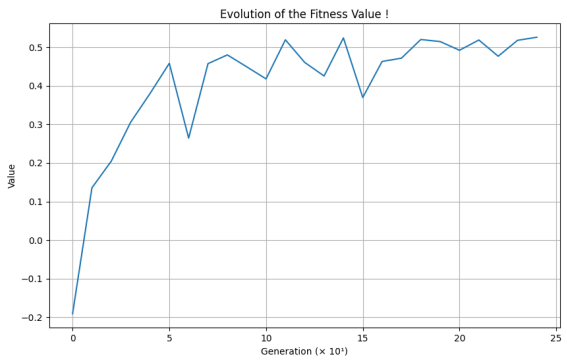

Fig. 6. Evolution of the fitness values. The x-axis represents the individual instances. The y-axis denotes the corresponding fitness values achieved by these individuals.

are sufficiently adapted to the optimization problem. Such a plateau often signals that the algorithm has either found an optimal solution.

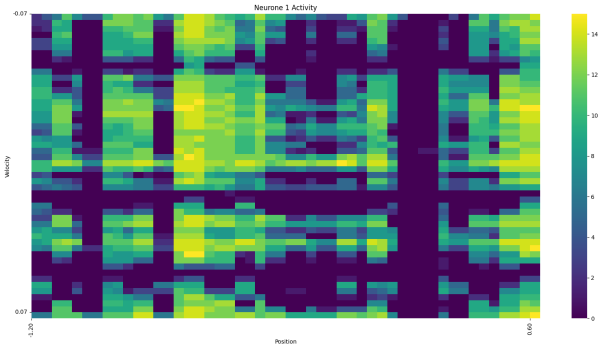

Fig. 7. Activity of Neuron 1. The x-axis is the position space. The y-axis is the velocity space. The intensity of the color is the spike count.

Figure 7 represents the activity of Neuron 1 of the output layer, which is associated with the action 'go to the left' in a simulated environment. The number of spikes generated by this neuron within a 20ms simulation window is plotted against the state space defined by the velocity (y-axis) and position (x-axis) of the agent. The intensity of the color indicates the spike count, with warmer colors representing higher activity. From the heatmap, we can observe a correlation between the neuron's activity and the vehicle's position. Specifically, there is heightened activity in the regions that likely correspond to the valley positions [-0.6, -0.4] in the state space. This suggests that Neuron 1 is particularly stimulated to drive the vehicle to the left when it is located in these valley regions, to gain momentum and gain energy by moving uphill. This pattern of activity is an essential component of the learning process, as it reflects the neural adaptation to specific environmental states that necessitate a leftward action to achieve a goal, such as reaching the top of a hill.

Figure 8 displays captures the activity of Neuron 2 of the

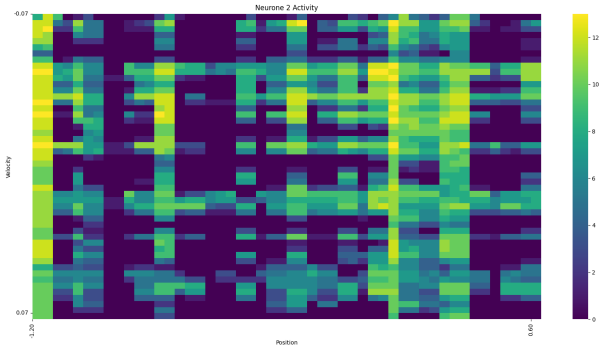

Fig. 8. Activity of Neuron 2. The x-axis is the position space. The y-axis is the velocity space. The intensity of the color is the spike count.

output layer. A closer examination of the heatmap reveals that Neuron 2 is most active around the position 0.5, which is presumably the target position in the simulation environment. This suggests that the neuron is stimulated to activate more intensely as the vehicle approaches its goal, contributing to decelerating and stopping the vehicle at the crucial moment. The heightened activity around the target position indicates an adaptive response that has likely been reinforced throughout the learning process to achieve the desired outcome of halting the vehicle precisely at the goal. This neural behavior is critical for precise control of the vehicle's movement, ensuring it does not overshoot the target.

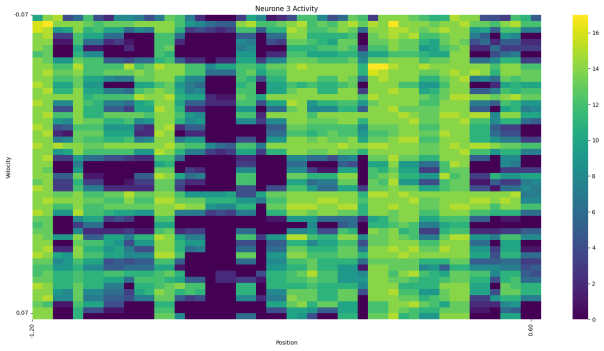

Fig. 9. Activity of Neuron 3. The x-axis is the position space. The y-axis is the velocity space. The intensity of the color is the spike count.

Figure 9 illustrates the activity of Neuron 3 of the output layer. From the heatmap, it is evident that Neuron 3 exhibits increased activity at the beginning of the first slope [-1.2, -0.8]. This pattern suggests that the neuron has been conditioned to propel the vehicle forward, giving it enough momentum to ascend the upcoming incline as the vehicle navigates through the valley. Further along the course, as the vehicle encounters the mountain slope [-0.3, -0.5], Neuron 3's activity peaks. This surge in neural stimulation is necessary to maintain the vehicle's velocity, enabling it to overcome the forces of gravity

and inertia that impede its uphill movement. The strategic activation of Neuron 3 demonstrates the network's learned ability to adaptively modulate action commands to navigate complex terrains effectively.

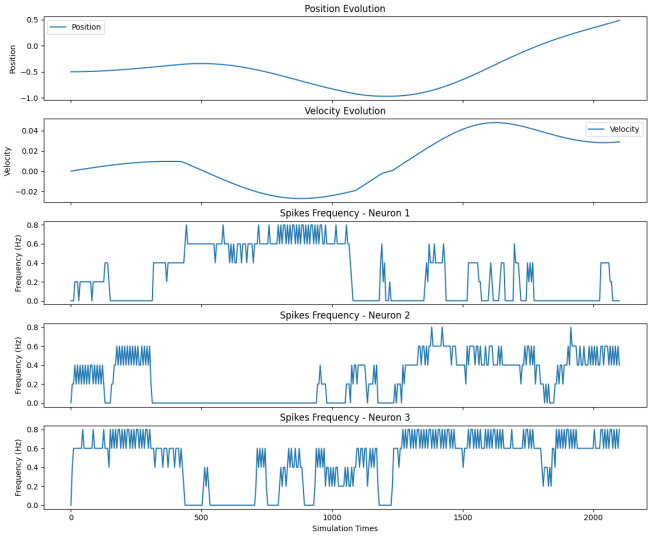

Fig. 10. Simulation Recording. The first curve shows the evolution of position over simulation time. The 2nd curve shows the evolution of velocity over simulation time. The 3rd is the activity frequency of neuron 1 over time. The 4th is the activity frequency of neuron 2 over time. The 5th is the activity frequency of neuron 3 over time.

Figure 10 presents a comprehensive view of a simulation over 110 (20ms/step) steps, detailing the position and velocity of the agent alongside the spike frequency of three distinct neurons, measured in Hertz. Initially placed at a position of -0.5 with zero velocity, the agent operates under the optimal found policy to navigate the simulation environment.

In the early phase, Neuron 3, responsible for moving right, shows a high spike frequency, initiating the agent's forward movement up a rightward slope. This activity is correlated with the initial increase in position to -0.35. Following this, Neuron 3's activity decreases, and Neuron 1, which triggers leftward movement, becomes active. This shift in neuronal control propels the agent backward up a leftward slope to a position of -1. The purpose of this maneuver is to gain momentum, akin to a slingshot effect, allowing the agent to subsequently surge forward with greater velocity to scale the mountainous terrain ahead.

As the simulation progresses, Neuron 3 reasserts dominance, its increasing activity corresponding with the agent's forward assault on the mountain at high speed. Upon nearing the goal, the middle neuron, Neuron 2, exhibits a significant spike in frequency, signaling a halt to the agent's progress near the flag, representation of the simulation's endpoint.

The spike frequencies of the neurons indicate a strategic interplay of neural commands, where each neuron's role is dynamically adjusted according to the agent's position and the environmental demands. Neuron 1's heightened activity during the early and mid-phases indicates a calculated withdrawal for a kinetic buildup, while Neuron 3's consistent involvement throughout the simulation underscores its role in forward propulsion. Finally, Neuron 2's late-stage activation suggests a braking mechanism, ensuring the agent stops at the desired target. This orchestration of neural inputs illustrates the learned policy's efficacy in achieving the simulation's objectives by modulating the agent's movements in response to the evolving terrain.

B. Enhancing SNNs with Synaptic Plasticity

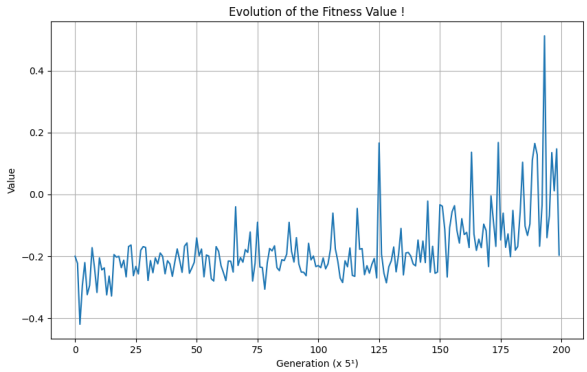

Fig. 11. Evolution of the fitness value for the best individual of each generation.

Integrating synaptic plasticity into the network, Figure 11 showcases the evolution of fitness values for the best individual across generations. The x-axis indicates the generation index (factor of 5), and the y-axis represents the fitness value. The plot reveals a gradual improvement in fitness over generations, with notable increases and some fluctuations, culminating in a peak at the 965th generation. This peak signifies the emergence of an individual that successfully solved the challenge.

The best-performing individual, produced at the 965th generation, activated approximately $\frac{1389}{2700} \approx 51.44\%$ of the synapses as plastic (plasticity_flag = 1). This individual resolved the MC challenge in 585 steps. Figures 14, 13, and 12 illustrate the evolution of synaptic weights during the simulation of the MC problem, depicting the adaptive changes in the network's synaptic strengths at various stages.

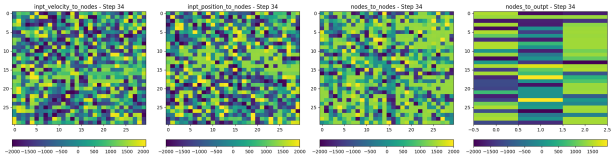

Fig. 12. Synaptic weights distribution at step 34. The first heatmap displays the synaptic weights from the input layer that encodes velocity to the nodes of the subsequent layer. The second heatmap illustrates the synaptic weights from the input layer that encodes the position to the nodes of the intermediate layer. The third heatmap represents the connections within the intermediate layer itself, showing the weights between nodes in this layer. The final heatmap in the sequence represents the synaptic weights from the intermediate layer to the output layer.

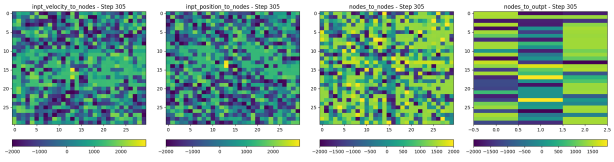

Fig. 13. Synaptic weights distribution at step 305. Same description as figure[12]

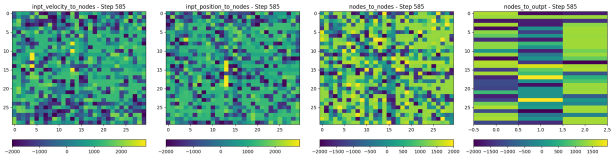

Fig. 14. Synaptic weights distribution at step 585. Same description as figure[12]

Through the sequence of the synaptic weight distribution maps at steps 34, 305, and 585 (Figures 12, 13, and 14), we can observe the dynamic adaptation process of the spiking neural network. As the agent explores the environment, the weights begin to show more organized patterns, reflecting the network's adaptation to the MC environment to enhance its performance. Finally, at step 585, we see that the weights have adapted further, aligning with the successful strategy that enabled the individual to solve the challenge (Figure 14).

After the initial simulation, which has been detailed in the previous sections, a second simulation was conducted, starting from the final state of the network from the last step of the first simulation. In this subsequent run, the network, while retaining its plasticity, successfully completed the challenge in 382 steps. This result was further solidified by a third simulation, again initiating from the network's state at the last step of the second simulation. The network continued to exhibit plasticity and accomplished the task in 431 steps, demonstrating consistent performance even with ongoing synaptic modifications. These simulations highlight the robustness of the SNNs ability to maintain its task-solving efficiency across successive runs.

From the fourth simulation onwards, there was a noticeable decrease in the network's efficiency, which can be attributed to a well-known phenomenon in synaptic plasticity, particularly in STDP, known as Synaptic Saturation (SS) [11]. SS occurs when the synapses in a neural network have been strengthened to the point that they can no longer increase in efficiency with further learning. This happens because, in STDP, the synaptic weights are adjusted according to the timing of spikes between neurons. If a presynaptic neuron consistently fires just before a postsynaptic neuron, the synapse between them strengthens. Over time, as the network learns, many synapses can be saturated. This means that those synapses lose the ability to adapt and fine-tune the response to new or nuanced stimuli, which can lead to a decline in the overall learning performance of the network. SS is a critical challenge in designing neural networks with plasticity, as it can limit the long-term adaptability and efficiency of the network.

V. Conclusion and Discussion

The integration of synaptic plasticity, alongside the optimization of SNN parameters through GAs, opens new avenues for the development of more adaptive and efficient computational models. Our findings not only demonstrate the feasibility of employing SNNs and GAs in solving complex RL tasks but also underscore the potential benefits of further research in this area, particularly in terms of exploring more intricate models of synaptic behavior and learning processes. Moreover, investigating the scalability of the proposed approach to more complex environments and tasks could further validate the robustness and versatility of optimized SNNs in the broader context of ML and AI.

Furthermore, we posit that an increase in computational resources could allow for extending the simulation steps from 600 to 1000. As a plastic network requires a higher number of simulation steps compared to a static one. The network's need for more simulation steps is expected to adapt and learn the best approach to solve the problem, instead of using a static solution. Such an extension would afford the network a greater opportunity for synaptic convergence, thereby enhancing the learning and adaptation process. The first individual to solve the challenge did it in 585 steps, which means that there could be other individuals who did not have enough time to converge before him.

One of the next step we can proposed is the implementation of a novel fitness function designed to address the challenge of the extreme number of steps required by the network to solve the problem. Specifically, we introduced a two-tiered fitness evaluation strategy:

- If the maximum position (`MaxPosition`) of the car is less than 0.5, the fitness value is set equal to `MaxPosition`.
- Conversely, if `MaxPosition` is greater than or equal to 0.5, the fitness value is enhanced by a bonus calculated as `MaxPosition + 100/sim_step`, thus incentivizing solutions that reach the goal in fewer steps.

This revised fitness function effectively encourages the algorithm to challenge the individual who solves the challenge to force them to solve it in less time. And we will find solutions that not only reach the target position but do so with an optimized number of steps, addressing a crucial efficiency aspect of SNN performance in solving the MC problem.

In conclusion, our study contributes to the ongoing discourse in the field of computational neuroscience and ML by highlighting the potential of using plastic SNNs. The advancements proposed herein not only address specific challenges inherent in the MC problem but also lay the groundwork for further exploration into the capabilities and applications of biologically-inspired computational models.

Appendix A
Source Code and Experiment Replication

The full source code for the simulations and analyses discussed in this paper can be found on GitHub at the following link:

https://github.com/Sterley/Plasticity-Driven-SNN

The repository includes a `README.md` file which provides detailed instructions on how to replicate the experiments and use the code.

REFERENCES

1 Neuronal dynamics - a neuroscience textbook, cambridge university press in july 2014. [Online]. Available: https://neuronaldynamics.epfl.ch/index.html

2 D. Haşegan, M. Deible, C. Earl, D. D'Onofrio, H. Hazan, H. Anwar, and S. A. Neymotin, "Evolutionary and spike-timing-dependent reinforcement learning train spiking neuronal network motor control," 2022-08-11. [Online]. Available: https://www.biorxiv.org/content/10.1101/2021.11.20.469405v3

3 NEURON | empirically-based simulations of neurons and networks of neurons. 2022, duke, yale and the blue brain project. [Online]. Available: https://www.neuron.yale.edu/neuron/

4 A. Yegenoglu, A. Subramoney, T. Hater, C. Jimenez-Romero, W. Klijn, A. Pérez Martín, M. van der Vlag, M. Herty, A. Morrison, and S. Diaz-Pier, "Exploring Parameter and Hyper-Parameter Spaces of Neuroscience Models on High Performance Computers With Learning to Learn," *Frontiers in Computational Neuroscience*, vol. 16, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2022.885207

5 M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

6 Z. Bing, I. Baumann, Z. Jiang, K. Huang, C. Cai, and A. Knoll, "Supervised learning in SNN via reward-modulated spike-timing-dependent plasticity for a target reaching vehicle," *Frontiers in Neurorobotics*, vol. 13, 2019. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2019.00018

7 T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel, C. Pehle, S. Billaudelle, G. Kiene, C. Mauch, J. Schemmel, K. Meier, and M. A. Petrovici, "Demonstrating advantages of neuromorphic computation: A pilot study," *Frontiers in Neuroscience*, vol. 13, p. 260, 2019.

8 Y. Ruan, Y. Xiong, S. Reddi, S. Kumar, and C.-J. Hsieh, "Learning to learn by zeroth-order oracle," 2020-02-07. [Online]. Available: http://arxiv.org/abs/1910.09464

9 S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks: Theory and Applications*, ser. Studies in Computational Intelligence, S. Mirjalili, Ed. Springer International Publishing - 2019, 2023, pp. 43–55. [Online]. Available: https://doi.org/10.1007/978-3-319-93025-1_4

10 Mountain car v1.0.0 - gym documentation. [Online]. Available: https://www.gymlibrary.dev/environments/classic_control/mountain_car/

11 T. D. B. Nguyen-Vu, G. Q. Zhao, S. Lahiri, R. R. Kimpo, H. Lee, S. Ganguli, C. J. Shatz, and J. L. Raymond, "A saturation hypothesis to explain both enhanced and impaired learning with enhanced plasticity," *eLife*, vol. 6, 2017.

12 M. Akl, D. Ergene, F. Walter, and A. Knoll, "Toward robust and scalable deep spiking reinforcement learning," *Frontiers in Neurorobotics*, vol. 16. 2023, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2022.1075647

13 Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12. 2018, 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2018.00331

14 R. Zhao, Z. Yang, H. Zheng, Y. Wu, F. Liu, Z. Wu, L. Li, F. Chen, S. Song, J. Zhu, W. Zhang, H. Huang, M. Xu, K. Sheng, Q. Yin, J. Pei, G. Li, Y. Zhang, M. Zhao, and L. Shi, "A framework for the general design and computation of hybrid neural networks," *Nat Commun*, vol. 13, pp. 3427 – 2022–06–14, 2022. [Online]. Available: https://www.nature.com/articles/s41467-022-30964-7

15 F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, no. 70, pp. 2171–2175, 2012. [Online]. Available: http://jmlr.org/papers/v13/fortin12a.html

16 L2l — l2l 1.0.0 documentation 2017, anand subramoney. created using sphinx 3.3.0. [Online]. Available: https://meta-optimization.github.io/L2L/

17 Introduction — neuromatch academy: Computational neuroscience. [Online]. Available: https://compneuro.neuromatch.io/tutorials/intro.html

18 G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review,," *Neural Networks*, vol. 113, pp. 54–71, 2019-05. [Online]. Available: http://arxiv.org/abs/1802.07569

19 D. Chen, P. Peng, T. Huang, and Y. Tian, "Deep reinforcement learning with spiking q-learning," 2022-01-21. [Online]. Available: http://arxiv.org/abs/2201.09754

20 D. Patel, H. Hazan, D. J. Saunders, H. T. Siegelmann, and R. Kozma, "Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game," *Neural Networks*, vol. 120, pp. 108–115, 2019-12-01. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019302266

21 X. Jiang, Q. Zhang, J. Sun, and R. Xu, "Fully spiking neural network for legged robots," 2023-10-08. [Online]. Available: http://arxiv.org/abs/2310.05022

22 G. Tang, N. Kumar, R. Yoo, and K. Michmizos, "Deep reinforcement learning with population-coded spiking neural network for continuous control," in *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 2021-10-04, pp. 2016–2029. [Online]. Available: https://proceedings.mlr.press/v155/tang21a.html

23 D. Zhang, T. Zhang, S. Jia, X. Cheng, and B. Xu, "Population-coding and dynamic-neurons improved spiking actor network for reinforcement learning," 2022-09-22. [Online]. Available: http://arxiv.org/abs/2106.07854

24 S. F. Chevtchenko and T. B. Ludermir, "Combining STDP and binary networks for reinforcement learning from images and sparse rewards," *Neural Networks*, vol. 144, pp. 496–506, 2021-12-01. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608021003609