Research project

Progress report

Exploring Plasticity-Driven Spiking Neural Networks for Enhanced Machine Learning

---

BY

Sterley Gilbert LABADY

Master 2 Artificial Intelligence & Robotics

ETIS UMR8051, CY Cergy Paris University / ENSEA / CNRS
6 avenue du Ponceau, 95014 Cergy-Pontoise Cedex, France

Delivered December 21, 2023

Directed by :

Pr.	Thanos MANOS	ETIS Laboratory, CNRS UMR 8051	Tutor 1
Pr.	Cristian JIMENEZ-ROMERO	ETIS Laboratory, CNRS UMR 8051	Tutor 2
Pr.	Mathias QUOY	ETIS Laboratory, CNRS UMR 8051	Tutor 3

Contents

1 Introduction

1.1 Project Description and Context

This research project explores the frontiers of Spiking Neural Networks (SNNs) in the context of machine learning, with a particular focus on their application within reinforcement learning environments. SNNs, distinct from traditional Artificial Neural Networks (ANNs), mimic the biological processes of the human brain, making them an intriguing area of study in computational neuroscience and artificial intelligence. 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.

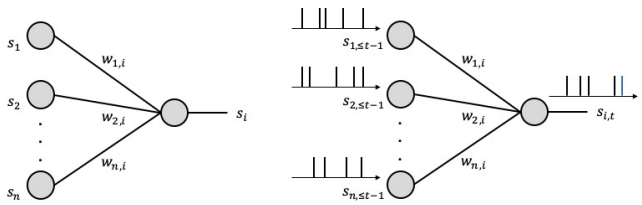

Figure 1 – Illustration of neural networks: (left) an ANN, where each neuron processes real numbers; and (right) an SNN, where dynamic spiking neurons process and communicate binary sparse spiking signals over time. [1]

The application of SNNs in the realm of machine learning, especially in environments characterized by uncertainty and the need for quick adaptation, presents a unique set of challenges and opportunities. This project aims to contribute to this evolving field by developing optimized SNN architectures that integrate local plasticity rules, thereby enhancing adaptability and computational efficiency. The overarching goal is to push the boundaries of what is currently possible with SNNs, thereby advancing the state of the art in neural network research.

1.2 Objectives

The primary objective of this research is to investigate the structure and functioning of SNNs to enhance their applicability in machine learning tasks. This involves a deep dive into the mechanisms that underlie the adaptability and efficiency of SNNs, particularly in how they process and learn from temporal information. Key to this investigation is the integration of brain-inspired mechanisms, specifically local plasticity rules, which are hypothesized to significantly improve the learning capabilities of SNNs.

To demonstrate the practical applicability and effectiveness of these enhanced SNNs, the research utilizes the Mountain Car environment, a standard testbed in reinforcement learning. This environment poses a simple yet challenging problem: a car must be driven up a steep hill, which can only be accomplished by learning and applying the optimal strategy over time. This choice of environment is strategic, as it provides a clear and controlled scenario to test the learning capabilities and efficiency of the SNN models being developed.

1.3 Problems and Challenges

Training Spiking Neural Networks (SNNs) for specific machine learning tasks presents unique challenges, primarily due to the infeasibility of using gradient-based methods as in Artificial Neural Networks (ANNs). This is a result of the discontinuous nature of spikes in SNNs, which complicates the application of traditional backpropagation. Furthermore, incorporating local plasticity rules into SNNs, essential for autonomous learning and adaptation, adds complexity to the network architecture and training process. In Section 2, we will conduct a state-of-the-art review of various training methods for SNNs, analyzing their efficiency, learning effectiveness, adaptability, and biological relevance.

1.4 Structure of the Report

The report is organized into several chapters, each focusing on a different aspect of the research:

— **Chapter 1: Introduction** - This chapter sets the stage for the research, outlining the project's context, objectives, and challenges.

— **Chapter 2: State of the Art** - A review of the current literature and developments in the field of SNNs Training, particularly in the context of machine learning applications.

— **Chapter 3: Experiments** - Details of some experiments, methodologies employed, and results obtained from applying SNNs within the Mountain Car environment.
— **Chapter 4: Future Work** - Discussion on potential avenues for further research, including exploration of more complex environments.
— **Chapter 5: Conclusion** - Summarizing the key findings, implications for the field, and final thoughts on the future trajectory of SNNs in machine learning.

Each chapter is designed to build upon the previous ones, culminating in a comprehensive understanding of the role and potential of SNNs in advancing machine learning capabilities.

2 State of the Art

2.1 Neuronal Dynamics Key Concepts

The exploration of fundamental neuronal concepts such as action potentials, synaptic activities, and firing thresholds,by Wulfram Gerstner et al.[2], 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.

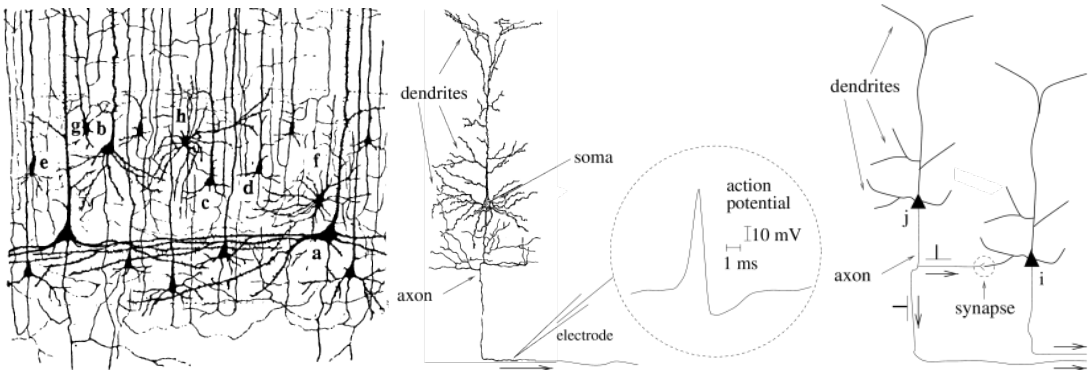

Figure 2 – This reproduction of a drawing of Ramón y Cajal shows a few neurons in the mammalian cortex that he observed under the microscope. [2]

$$\tau_m \frac{du}{dt} = -[u(t) - u_{\text{rest}}] + RI(t).[2] \tag{1}$$

The Leaky Integrate-and-Fire model, as presented [2], serves as a starting point for modeling neuronal dynamics. Its simplicity yet effectiveness in representing neuronal behavior is particularly relevant. This model aids in understanding how neurons accumulate signals and the conditions under which they generate action potentials, which is pivotal in constructing realistic neural network. Understanding the mechanisms of synaptic change and adaptation, including the principles of Long-Term Potentiation and Long-Term Depression, informs our approach to modeling how neural networks learn and adapt over time. This is especially significant in our endeavor to develop neural networks that can mimic learning processes akin to those in biological systems. The discussion about the spatial structure of neurons, particularly dendrites and synapses, is beneficial for understanding how neuronal networks are interconnected and how information flows within these networks. This understanding is critical for our project, as it guides the design of network architectures that reflect the complex structure of biological neural networks, thereby enhancing the functionality and efficiency of our simulations. By understanding how neuronal populations function and interact, we can better simulate large-scale neural networks, mirroring the collective behavior of neurons in biological systems. This perspective is essential for analyzing network-level phenomena and understanding how individual neuronal behaviors culminate in complex cognitive functions.

The theoretical insights and methodologies presented in "Neuronal Dynamics" [2] are instrumental. They 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.

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

Yegenoglu et al. [3] has proposed a feed-forward network SNN of Leaky Integrate-and-Fire (LIF) neurons to solve the Mountain Car Challenge. The inputs to the SNN are the position and velocity of the car, discretized and encoded using 30 input neurons for each variable. This discretization process is mathematically expressed as:

$$w = \frac{\min + \max}{n}[3]$$

where w is the width of the bins, min and max are the minimum and maximum values of the interval, and n is the number of input neurons. The SNN is implemented in NEST 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 Mountain Car task. The action sent to the OpenAI Gym 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, mathematically represented as:

$$f = \max_T(P_T)[3]$$

where P_T contains the position of the car at each simulation step up to $T = 110$. The optimization process uses a Genetic Algorithm. This algorithm optimizes 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 shows 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) is first achieved around generation 160. In subsequent generations, while the mean fitness saturates at around 0.3, the best fitness reaches 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 Mountain Car 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 genetic algorithms in reinforcement learning task.

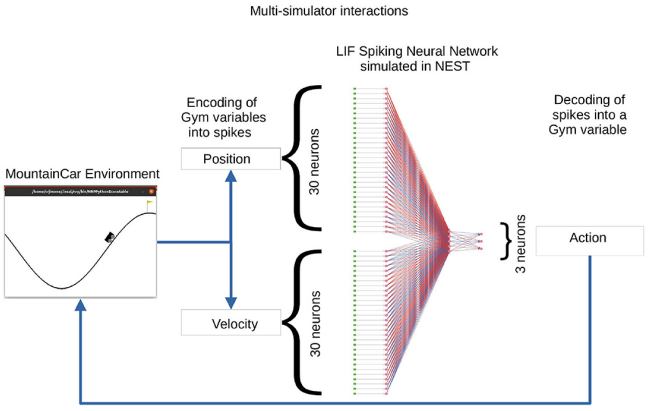

Figure 3 – A feed-forward spiking network to solve the Mountain Car task. [3]

This study provides a significant contribution to the understanding and application of SNNs in complex machine learning environments. It demonstrates the potential of combining SNNs with evolutionary strategies, like Genetic Algorithms, to efficiently navigate and optimize in high-dimensional parameter spaces. Is offers a viable alternative to traditional reinforcement learning and gradient descent methods. Despite the promising outcomes demonstrated by Yegenoglu et al.[3], the complexity and computational demands of training SNNs, particularly when integrated with evolutionary strategies like Genetic Algorithms, 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 Mountain Car problem, the generalizability of this approach to other scenarios with different dynamics and complexity is not yet clear.

2.3 Evolutionary and spike-timing-dependent reinforcement learning train spiking neuronal network motor control

Daniel Haşegan et al.[4] presents a novel approach to enhancing the performance of Spiking Neuronal Network (SNN) models. The focus is on comparing and analyzing the effectiveness of two biologically inspired learning

mechanisms: spike-timing-dependent reinforcement learning (STDP-RL) and evolutionary strategy (EVOL). The central premise is that while artificial neural networks (ANNs) have succeeded in various domains, the performance of more biologically realistic SNN models in similar tasks has been relatively suboptimal. The study al.[4] addresses this gap by exploring the potential of STDP-RL and EVOL in optimizing SNNs. The researchers employed these mechanisms on SNNs to solve the CartPole reinforcement learning (RL) control problem, a task where a pole must be balanced on a moving cart. This study aims 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.

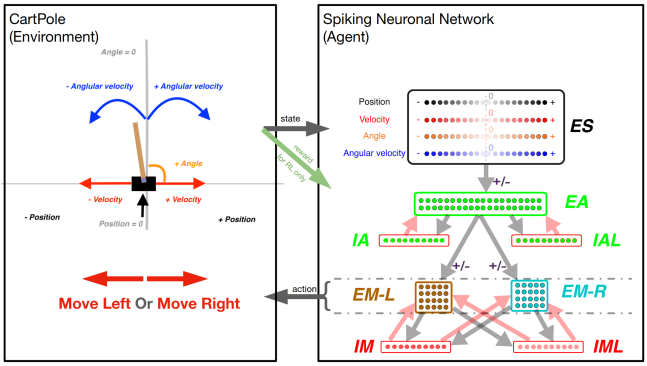

Figure 4 – The CartPole game environment (left) interfacing with the SNN model (right). [4]

Daniel Haşegan et al.[4] utilized the OpenAI Gym platform for simulations. 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 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 enhances 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 Mountain Car project. EVOL's success in optimizing Spiking Neural Networks (SNNs) for dynamic tasks aligns well with the challenges of the Mountain Car 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 Reinforcement Learning tasks. EVOL's evolutionary approach to exploring diverse network architectures will allow us to identify optimal strategies for acceleration and momentum-building, key elements in the Mountain Car challenge. Adapting EVOL to our specific needs could lead to significant advancements in SNN capabilities, particularly in complex, time-sensitive decision-making tasks in machine learning.

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

Zhenshan Bing et al.[5] delves into the realm of Spiking Neural Networks (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 marks 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. [5]

$$w_{ij}(t) = w_{ij}(t - \Delta t) + \Delta w_{ij}(t) \tag{2}$$

$w_{ij}(t)$: Weight of the synaptic connection between pre-synaptic neuron i and post-synaptic neuron j at time window t.
Δt : Length of the simulation time window T.
$\Delta w_{ij}(t)$: Change in synaptic weight during the current time window.

$$\Delta w_{ij}(t) = \eta \times r_{ij}(t) \times \text{STDP}_{ij}(t) \times g_{ij}(t) \tag{3}$$

$\Delta w_{ij}(t)$: Change in synaptic weight during the current time window.
η : Learning rate, controlling the speed of learning in the SNN (Spiking Neural Network).
$r_{ij}(t)$: Reward signal calculated for the synaptic connection between neurons i and j.
$STDP_{ij}(t)$: Spike-Timing-Dependent Plasticity function for the synaptic connection.
$g_{ij}(t)$: Additional factor or function influencing the weight change.

STDP is a biological learning rule based on the relative timing of spikes between neurons. It posits that the strength of synaptic connections is adjusted depending on the timing of the action potentials (spikes) of the connected neurons. Specifically, if a presynaptic neuron's spike precedes a postsynaptic neuron's spike within a short time window, the synaptic strength increases, facilitating the synaptic connection (known as Long-Term Potentiation or LTP). If the postsynaptic neuron spikes before the presynaptic neuron, the synaptic strength decreases (Long-Term Depression or LTD). R-STDP, on the other hand, extends the STDP learning rule by incorporating a reward signal into the learning process. This means that the synaptic adjustments (LTP or LTD) 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 reinforcement learning aspect, allowing the network to not only learn patterns but also to optimize its behavior towards achieving specific goals or tasks.

Bing et al.[5] demonstrates the application of SNNs in controlling a simulated mobile robot to reach a target while avoiding obstacles. This scenario is very much in line with the kind of complex tasks that our project might be addressing. The end-to-end learning approach adopted in their study involves training SNNs using pre-generated datasets to guide the robot's behavior. This method of training can be particularly beneficial for your project as it provides a framework for designing SNNs that can learn and adapt to perform specific tasks efficiently. It addresses 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 proposes solutions to these challenges but also demonstrates the feasibility of the approach through practical experiments. This aspect of the study can offer valuable insights, especially if it involves designing neural networks that can adapt and learn from their environment.

2.5 BPTT in an SNN context

Mahmoud Akl et al.[6] sheds light on the innovative use of backpropagation through time (BPTT) in training SNNs. This method is particularly significant, especially if it involves complex temporal dynamics or seeks to model systems that evolve over time. BPTT, traditionally used in training recurrent neural networks, is adapted in to train SNNs, which inherently process information over time due to their spiking nature. BPTT is use to effectively propagate gradients backward through time, accounting for the temporal dependencies in the data. This approach is pivotal for our project as it suggests a pathway to harness the computational advantages of temporal dynamics in SNNs, bringing them closer to the capabilities of traditional deep learning models while retaining their biological plausibility.

When an SNN processes input data, the response is not just a single action but a series of spikes over time. BPTT takes into account these temporal sequences, effectively treating the SNN as a recurrent network. During the training process, the algorithm calculates gradients not only based on the current state but also considering

the network's behavior at previous time steps. This backward flow of gradients through time allows the network to learn from the temporal sequence of events, which is critical for tasks where timing of inputs and outputs is paramount.

Mahmoud Akl et al.[6], test of the deep spiking reinforcement learning method is through its application to various control problems from the OpenAI Gym. This approach allows for a practical evaluation of the method in diverse, complex environments. Incorporating BPTT in training SNNs for your project could lead to models that are not only more adept at handling temporal data but also more efficient in terms of computational resources.

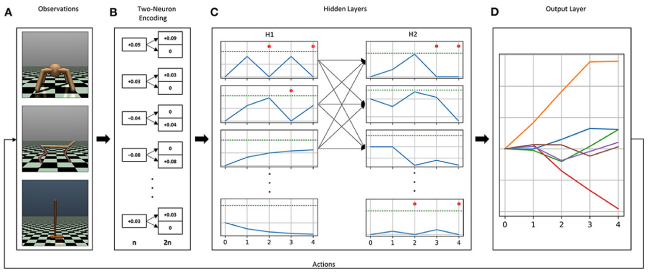

Figure 5 – he continuous control OpenAI Gym environments Ant-v3. [6]

Furthermore, the study's exploration of BPTT in SNNs [6], coupled with the management of unique hyper-parameters such as decay factors and firing thresholds, offers a comprehensive approach to modeling complex systems. It highlights the potential of SNNs to be trained effectively for a variety of tasks, ranging from simple control problems to more intricate tasks that involve continuous decision-making. BPTT in training SNNs opens up new avenues for your project. It provides a framework for effectively leveraging the temporal processing capabilities of SNNs, paving the way for developing models that are both biologically inspired and computationally powerful. This approach could be a key to manage complex, time-dependent data processing.

2.6 Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks

The Spatio-Temporal Backpropagation (STBP) algorithm introduced in the research by Yujie Wu et al.[7] offers a groundbreaking approach to training Spiking Neural Networks (SNNs). Traditional training methods for SNNs, including unsupervised learning and indirect supervised learning, often fall short in achieving high performance, primarily due to their limited ability to simultaneously address the spatial and temporal characteristics of neural activities. The STBP algorithm addresses these challenges by integrating both the spatial and temporal domains in the training phase. It does this through an innovative approach that involves approximating the derivative of spike activity, making it compatible with gradient descent training. This approximation is critical as it allows the algorithm to effectively navigate the non-differentiable points of spike activity, which are a fundamental aspect of SNNs.

Theoretically, $g(u)$ is a non-differentiable Dirac function of $\delta(u)$ which greatly challenges the effective learning of SNNs. $g(u)$ has zero value everywhere except an infinity value at zero, which causes the gradient vanishing or exploding issue that disables the error propagation. For the Derivative Approximation of the Non-Differentiable Spike Activity, Yujie Wu et al.[7] introduce four curves to approximate the derivative of spike activity denoted by h_1, h_2, h_3, and h_4:

$$h_1(u) = \frac{1}{a_1}\text{sign}(|u - V_{\text{th}}| < \frac{a_1}{2}), [7] \tag{4}$$

$$h_2(u) = (a_2\sqrt{2} - \frac{a_2^2}{4}|u - V_{\text{th}}|)\text{sign}(2a_2\sqrt{-|u - V_{\text{th}}|}), [7] \tag{5}$$

$$h_3(u) = \frac{1}{a_3}\frac{e^{\frac{V_{\text{th}}-u}{a_3}}}{(1 + e^{\frac{V_{\text{th}}-u}{a_3}})^2}, [7] \tag{6}$$

$$h_4(u) = \frac{1}{2\pi a_4}\sqrt{e^{-\frac{(u-V_{\text{th}})^2}{2a_4}}}, [7] \tag{7}$$

where $a_i (i = 1, 2, 3, 4)$ determines the curve steepness, i.e., the peak width. In fact, h_1, h_2, h_3, and h_4 are the derivative of the rectangular function, polynomial function, sigmoid function and Gaussian cumulative distribution function, respectively.

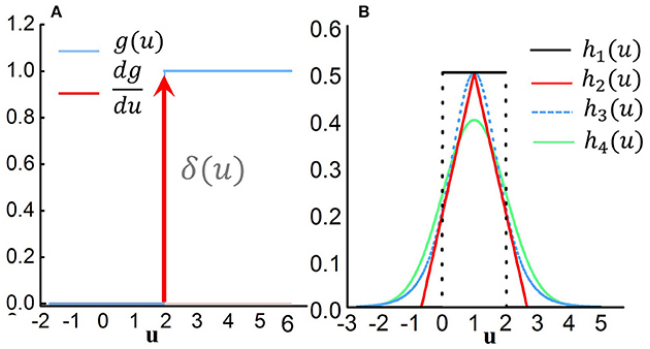

Figure 6 – Derivative approximation of the non-differentiable spike activity. (A) Step activation function of the spike activity and its original derivative function. (B) Several typical curves to approximate the derivative of spike activity. [7]

The STBP has been tested across various architectures and datasets, demonstrating its versatility and effectiveness. They employed both fully connected and convolutional architectures on static datasets like MNIST and a custom object detection dataset, as well as on dynamic datasets like N-MNIST. The results were impressive, indicating that the STBP method could achieve superior accuracy compared to existing state-of-the-art algorithms for SNNs. This suggests that the STBP approach is not only effective in handling the spatial-temporal dynamics inherent in SNNs but also in enhancing their performance across different types of neural network architectures and applications.The STBP method provides a new perspective in the field of computational neuroscience, particularly in the development and training of SNNs. Its ability to efficiently and accurately train SNNs using both spatial and temporal information opens up new avenues for research and application in areas where simulating brain-like behaviors and cognitive functions is crucial. For our project, exploring the potential of STBP could lead to significant advancements in neural network modeling, contributing to the development of more sophisticated and biologically-inspired computational models.

2.7 A framework for the general design and computation of hybrid neural networks

Rong Zhao et al. (2022) [8] delves into the realm of hybrid neural networks (HNNs), an innovative approach that blends the functionalities of spiking neural networks (SNNs) and artificial neural networks (ANNs). The cornerstone of this framework is the introduction of hybrid units (HUs), serving as interfaces that seamlessly integrate the characteristics of both SNNs and ANNs. This integration is pivotal in leveraging the strengths of each network type, thereby enhancing flexibility and efficiency in computational tasks.

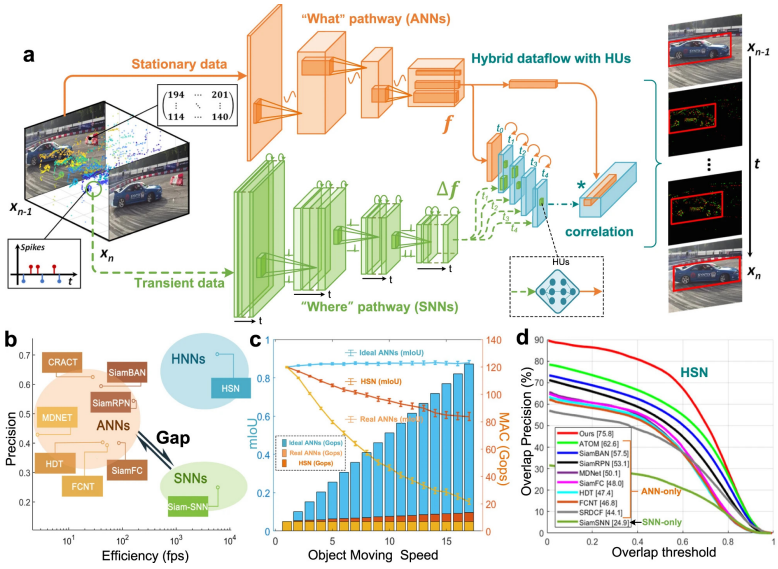

Figure 7 – The architecture of HSN. The orange part represents the "what" pathway processing the static information from APS, whereas the green part represents the "where" pathway processing the dynamic information from DVS. [8]

The concept of HNNs emerges from the need to address the limitations inherent in SNNs and ANNs when operating independently. ANNs, renowned for their proficiency in learning nonlinear relationships and handling large datasets, fall short in terms of energy efficiency and real-time data processing. On the other hand, SNNs excel in event-driven computation and temporal data processing but lag in precision and complex data handling. The HNN framework proposed in this study aims to amalgamate the advantages of both networks. The integration is facilitated by hybrid units (HUs), which are novel constructs designed to transform information between the two network types. These HUs are versatile, allowing for both manual design based on deterministic relationships and automatic learning for more complex or unknown relationships. This flexibility is crucial for adapting the HNN framework to a wide array of applications. Rong Zhao et al. (2022)[8] delineates three core implementations of HNNs:

Hybrid Sensing Network (HSN): This network exemplifies the divide-and-conquer strategy. It separates visual information into static and dynamic parts, processes them through respective pathways (ANNs for static and SNNs for dynamic), and combines the outputs using learnable HUs. The result is a system that significantly enhances tracking accuracy and efficiency, demonstrating superior performance compared to singular paradigm models.

Hybrid Modulation Network (HMN): The HMN utilizes an ANN-based backbone for continuous signal generation and an SNN-based branch network for task execution. Here, the HUs adjust neuron thresholds in the branch network based on task-related information from the backbone network, showcasing an efficient approach to meta-continual learning across multiple tasks.

Hybrid Reasoning Network (HRN): Targeting complex reasoning tasks, the HRN integrates multimodal information using ANN-based parsers and executes reasoning with an SNN-based symbolic analyzer. The HRN stands out in its ability to perform logical reasoning in a dynamic, interpretable, and robust manner, backed by both designable and learnable HUs.

These implementations indicate significant strides in accuracy. Particularly, the HRN's performance in visual question answering tasks, as evidenced by its high accuracy rates across various question types, marks a notable advancement in neural network applications. We should analysis of the computational overhead introduced by HUs. This is particularly pertinent given the inherently disparate computational models of SNNs and ANNs, where the asynchronous, event-driven nature of SNNs could conflict with the batch-processing efficiency of ANNs. The resultant computational complexity and its impact on training and inference times are critical factors for practical applications.

3 Experiments

3.1 Understanding the Fundamentals of Neuronal Dynamics

To comprehend the complexities of neuronal dynamics, a series of simulations were conducted to observe the behavior of neurons under different stimuli. The primary focus was on understanding the membrane potential fluctuations, the generation of action potentials, and the role of synaptic plasticity in learning and memory. The results of these simulations provide insights into the mechanisms by which neurons process information and adapt their synaptic strengths accordingly.

We experiemented the dynamics of an excitatory neuron under the influence of a direct current (DC) stimulus. The aim is to investigate the effects of this stimulation on the membrane potential of both an excitatory neuron and a target neuron.

The simulation was conducted using the NEST simulation tool, a prominent simulator for spiking neural network architectures. The parameters employed in the simulation are as follows:

- Neuron model: `iaf_psc_alpha` [9], an integrate-and-fire model with alpha-function shaped postsynaptic currents.
- Stimulation: A DC generator with an amplitude of 500 pA was connected to the excitatory neuron.
- Synaptic weight: The synaptic connection from the excitatory neuron to the target neuron was set to a weight of 1000.0 pA.
- Recording devices: A multimeter was used to record the membrane potential (V_m) at an interval of 0.1 ms. A spike recorder was also connected to both neurons to capture their spiking events.

The neuron model used in this simulation is based on the standard leaky integrate-and-fire (LIF) model[9].

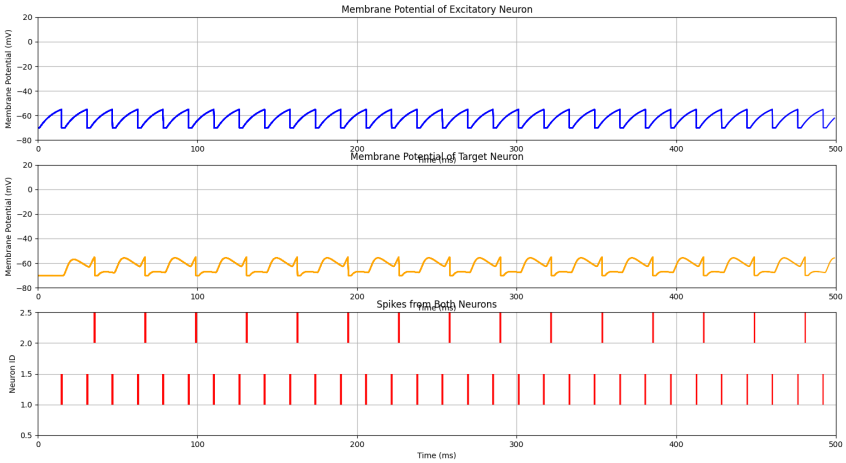

Figure 8 – Membrane potential traces of the excitatory and target neurons under DC stimulus.

3.2 Spike-Timing-Dependent Plasticity (STDP) Experimentation

We analysed of the STDP mechanism, a type of synaptic plasticity dependent on the timing of spikes between pre- and postsynaptic neurons. The simulation investigates the relationship between spike timing and synaptic weight adjustments.

The experiment was conducted using a custom simulation environment to model the behavior of synaptic weights under STDP rules. The main parameters for the simulation were as follows:

— Maximum synaptic weight (W_{max}): 1000.0 units.
— Minimum synaptic weight (W_{min}): 0.0 units.
— Initial synaptic weight (w): 500.0 units.
— STDP rate (λ): 0.01, controlling the rate of weight changes.
— STDP scaling factors (α): -1.0, (μ_+), (μ_-): both set to 1.0, these determine the shape of the STDP update function.
— Time constants for trace decay: τ_{tr_pre} and τ_{tr_post} set to 20.0 ms.
— Simulation time: 1000 ms with a time step (dt) of 1.0 ms.

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:

$$\text{Pre-synaptic trace update: } pre\_trace \leftarrow pre\_trace \times e^{-\frac{dt}{\tau_{tr\_pre}}} \tag{8}$$

$$\text{Post-synaptic trace update: } post\_trace \leftarrow post\_trace \times e^{-\frac{dt}{\tau_{tr\_post}}} \tag{9}$$

$$\text{Synaptic weight potentiation: } w_{pot} = W_{max}\left(\frac{w}{W_{max}} + \lambda\left(1 - \frac{w}{W_{max}}\right)^{\mu_+} pre\_trace\right) \tag{10}$$

$$\text{Synaptic weight depression: } w_{dep} = W_{max}\left(\frac{w}{W_{max}} - \alpha\lambda\left(\frac{w}{W_{max}}\right)^{\mu_-} post\_trace\right) \tag{11}$$

$$\text{Synaptic weight update: } w = \min(\max(W_{min}, w_{dep}), w_{pot}) \tag{12}$$

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

The following figure displays the temporal evolution of the pre- and postsynaptic traces along with the changes in synaptic weight over the course of the simulation, providing a visual insight into the STDP process.

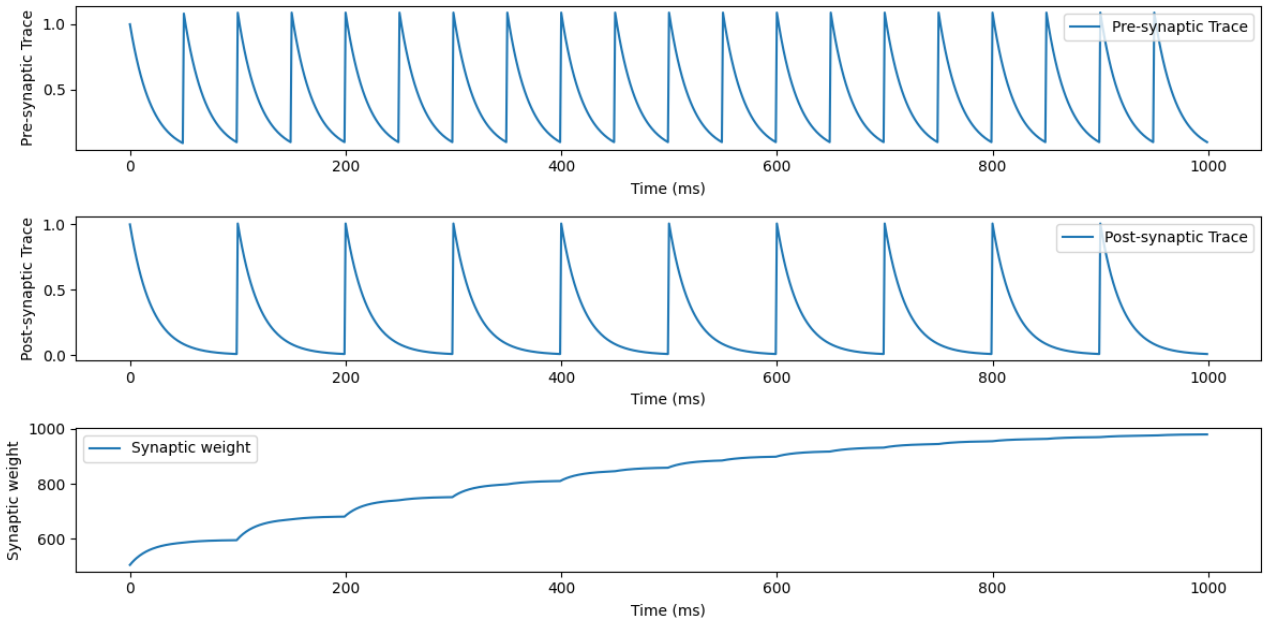

Figure 9 – Top: Pre-synaptic trace over time. Middle: Post-synaptic trace over time. Bottom: Evolution of the synaptic weight showcasing the STDP dynamics.

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

3.3.1 Introduction

Using the Yegenoglu et al.[3]'s approach, Genetic Algorithms (GAs)[10] with Spiking Neural Networks (SNNs) was explored, specifically in the context of the Mountain Car problem, a classical test in reinforcement learning. This combination presents a novel approach to harnessing the unique strengths of both GAs and SNNs. GAs are employed for their powerful optimization capabilities, systematically adjusting the parameters of SNNs to optimize performance in complex tasks. 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 Mountain Car problem but also sheds light on the broader potential of combining evolutionary algorithms with biologically-inspired neural models in machine learning tasks. Yegenoglu et al.[3] use The LearningToLearn[11] methodology, which is rooted in machine learning, 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 LearningToLearn[11] Method in optimizing neuroscience models.

Learning to Learn(L2L)[11] for Spiking Neural Networks (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 Genetic Algorithm[10] , to find optimal parameters for the Spiking Neural Networks.

The Mountain Car environment[12], a benchmark in reinforcement learning, 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.

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 1 – Mountain Car Environment Specifications[12]

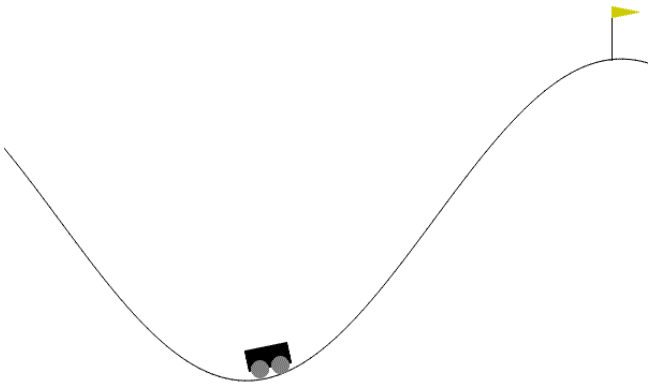

Figure 10 – The Mountain Car Environment used for demonstrating the capabilities of the enhanced SNN models.[12]

Observation Space The observation space is a 2D array, representing the car's position and velocity. Key details:

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

Table 2 – Observation Space[12]

Action Space The agent has three discrete actions: accelerate left, no acceleration, accelerate right. Table details:

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

Table 3 – Action Space

Transition Dynamics Given an action, the mountain car follows the following transition dynamics:

$$\text{velocity}_{t+1} = \text{velocity}_t + (\text{action} - 1) \times \text{force} - \cos(3 \times \text{position}_t) \times \text{gravity} \text{ [12]} \tag{13}$$

$$\text{position}_{t+1} = \text{position}_t + \text{velocity}_{t+1} \text{ [12]} \tag{14}$$

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.

3.3.2 Learning Setup

In our experiment, we utilized a spiking neural network (SNN) for solving the Mountain Car task. The SNN was optimized using a genetic algorithm (GA)[3].

[3]The SNN consisted of three layers: an input layer encoding the state variables (position and velocity), an intermediate layer, and an output layer representing the possible actions. The neurons were modeled as leaky integrate-and-fire units, a common choice in SNNs due to their biological plausibility and computational efficiency. The action is decoded from the network's output using spike event counts from the spiking neural network (SNN). 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.

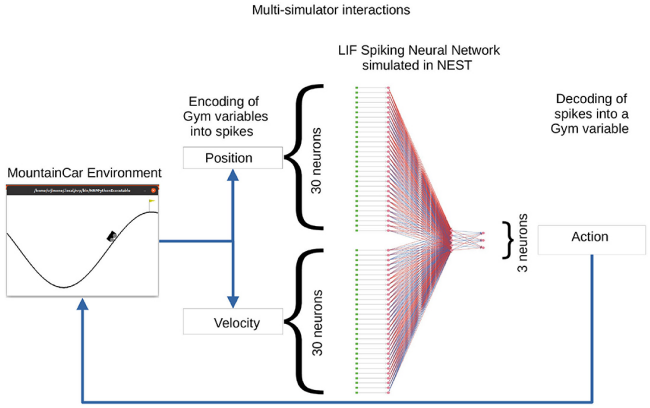

Figure 11 – A feed-forward spiking network to solve the Mountain Car task. [3]

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. 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 Mountain Car 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 Mountain Car task.

$$\text{Fitness}(I) = f(\text{paramètres}(I)) = \max_{T}(PET) \tag{15}$$

The fitness function for the Mountain Car (MC) optimization 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.

3.3.3 Results and Analysis

Figure 12 – Evolution of the fitness values.

The graph presents the evolution of the fitness values across generations of individuals throughout the learning process. The x-axis represents the individual instances, spanning multiple generations, while the y-axis denotes the corresponding fitness values achieved by these individuals. Notably, from the 600th individual onwards, which corresponds to the 18th generation (given that there are 32 individuals per generation), we observe that the fitness value begins to plateau at around 0.5. This indicates a convergence of the genetic algorithm where further iterations yield minimal improvement in fitness. The plateau suggests that the population has reached a point where the individuals are sufficiently adapted to the optimization problem. Such a plateau often signals that the algorithm has either found an optimal solution.

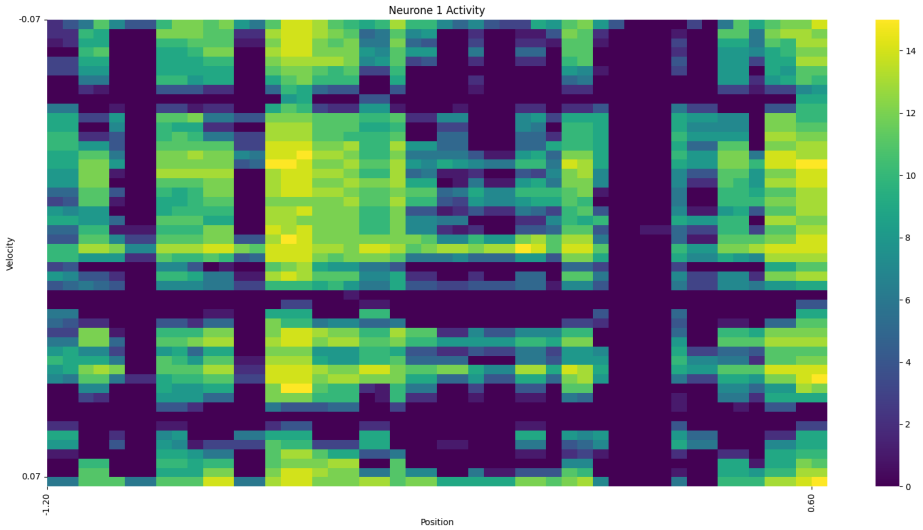

Figure 13 – Activity of Neuron 1.

This heatmap represents the activity of Neuron 1, 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.

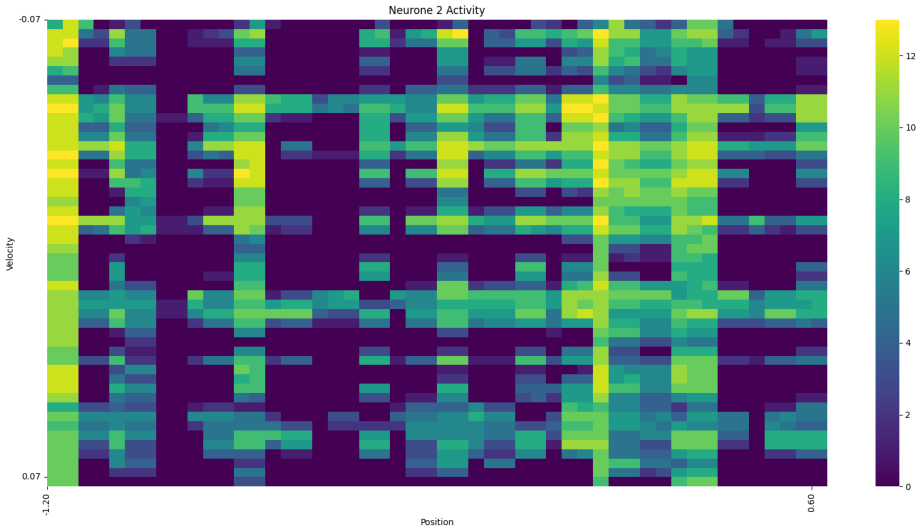

Figure 14 – Activity of Neuron 1.

The heatmap displayed captures the activity of Neuron 2. 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.

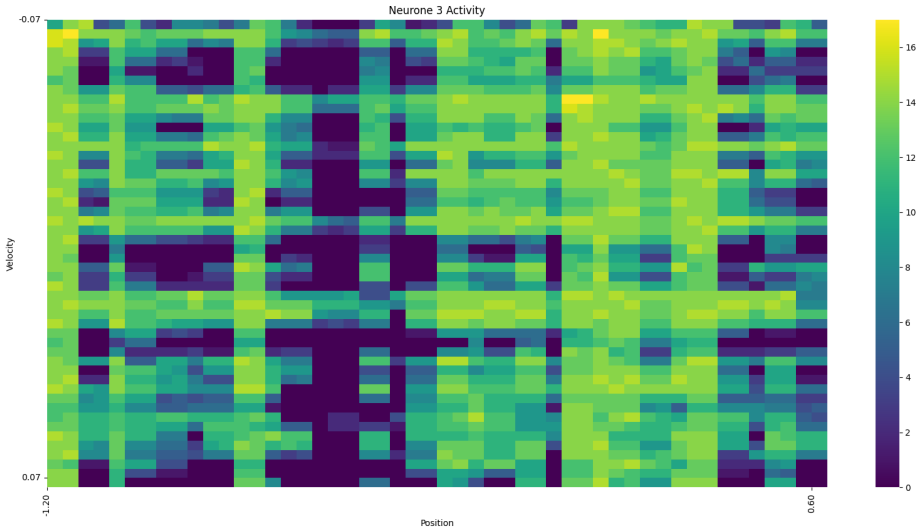

Figure 15 – Activity of Neuron 1.

The heatmap illustrates the activity of Neuron 3. 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.

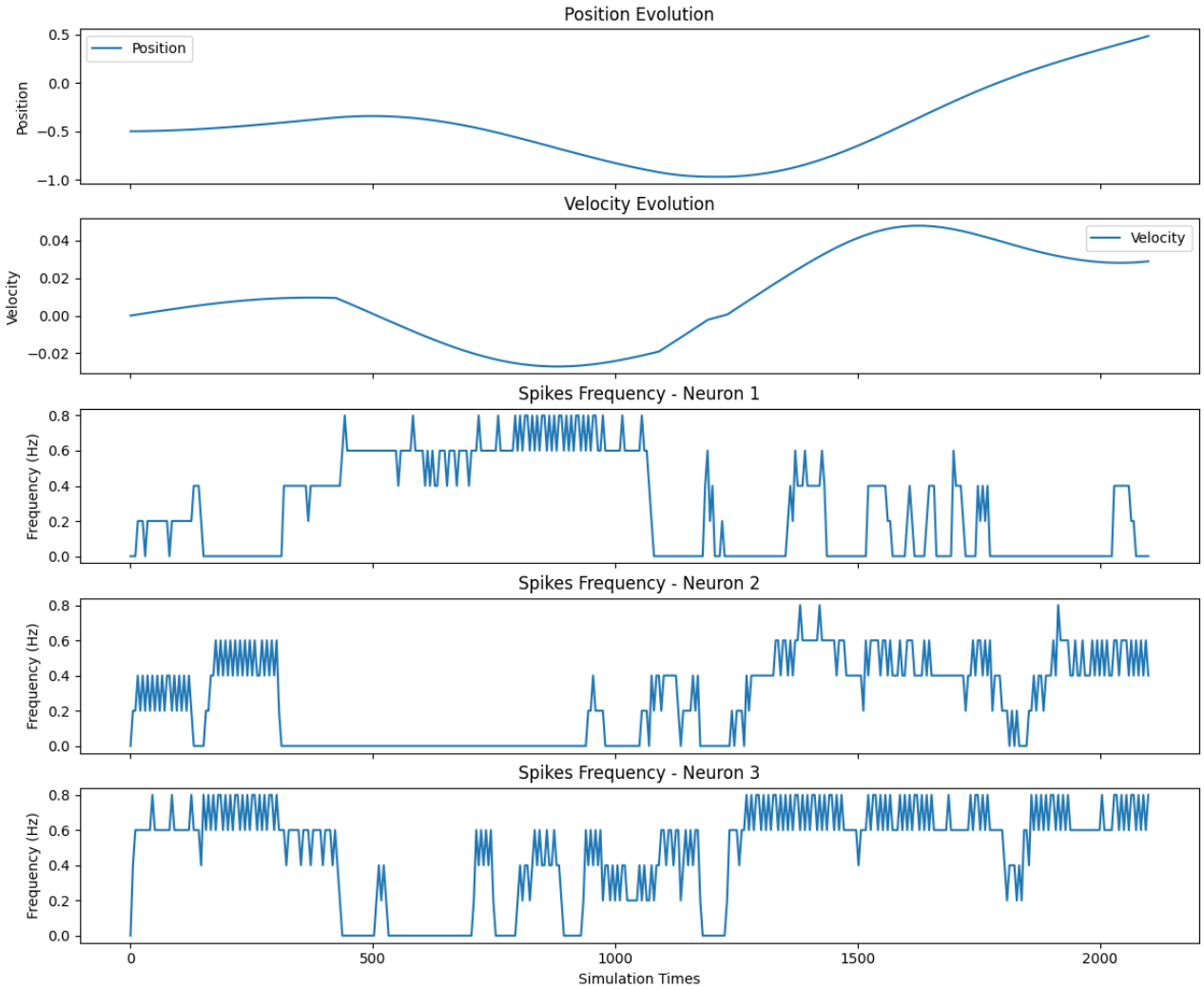

Figure 16 – Simulation Recording

The graph 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.

4 Future Work and Conclusion

The following points outline the prospective work that aims to extend the current understanding and application of SNNs:

- **Optimization Techniques:** The current work primarily utilized a Genetic Algorithm for the learning-to-learn (L2L) framework in optimizing SNNs. Future studies should consider the implementation of alternative optimization strategies, such as the Kalman filter.
- **Complex Challenge Application:** Going forward, the application scope should be broadened to encompass more complex challenges, such as controlling a pendulum (pendrum) or managing the dynamics of a cart-pole system (carpol). These complex tasks often require a sophisticated understanding of the environment and can benefit from the temporal processing advantages of SNNs.
- **Reinforcement Learning Signals:** Another promising direction is to integrate reinforcement learning (RL) signals to the SNN training. By leveraging the RL paradigm, the network could potentially self-optimize through trial and error, adjusting synaptic weights in response to received rewards. This approach may enable the SNNs to develop more nuanced behaviors and adapt to changing environments more effectively.
- **ANN vs SNN:** As an essential next step, we plan to tackle the Reinforcement Learning Challenges using an Artificial Neural Network (ANN) and conduct a comparative analysis with the Spiking Neural Network (SNN) approach. This comparison will provide valuable insights into the relative strengths and limitations of each model in terms of learning efficacy, computational efficiency, and temporal dynamics management.

The state-of-the-art review illuminated various approaches, such as the integration of the Leaky Integrate-and-Fire model, the application of evolutionary strategies and spike-timing-dependent reinforcement learning, and the use of reward-modulated STDP. We have made a significant strides in advancing our understanding and application of Spiking Neural Networks (SNNs) within machine learning, particularly in reinforcement learning environments. Through the implementation of Genetic Algorithm to optimize SNNs, we have observed the fundamental behaviors and potential of SNNs. The experiments conducted have laid a solid foundation for further exploration.

References

[1] Compute with time, not over it: An introduction to spiking neural networks – king's communications, learning & information processing lab.

[2] Neuronal dynamics - a neuroscience textbook by wulfram gerstner, werner m. kistler, richard naud and liam paninski.

[3] Alper Yegenoglu, Anand Subramoney, Thorsten Hater, Cristian Jimenez-Romero, Wouter Klijn, Aarón Pérez Martín, Michiel van der Vlag, Michael Herty, Abigail Morrison, and Sandra Diaz-Pier. Exploring Parameter and Hyper-Parameter Spaces of Neuroscience Models on High Performance Computers With Learning to Learn. *Frontiers in Computational Neuroscience*, 16, 2022.

[4] Daniel Haşegan, Matt Deible, Christopher Earl, David D'Onofrio, Hananel Hazan, Haroon Anwar, and Samuel A. Neymotin. Evolutionary and spike-timing-dependent reinforcement learning train spiking neuronal network motor control.

[5] Zhenshan Bing, Ivan Baumann, Zhuangyi Jiang, Kai Huang, Caixia Cai, and Alois Knoll. Supervised learning in SNN via reward-modulated spike-timing-dependent plasticity for a target reaching vehicle. 13.

[6] Mahmoud Akl, Deniz Ergene, Florian Walter, and Alois Knoll. Toward robust and scalable deep spiking reinforcement learning. 16.

[7] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. 12.

[8] Rong Zhao, Zheyu Yang, Hao Zheng, Yujie Wu, Faqiang Liu, Zhenzhi Wu, Lukai Li, Feng Chen, Seng Song, Jun Zhu, Wenli Zhang, Haoyu Huang, Mingkun Xu, Kaifeng Sheng, Qianbo Yin, Jing Pei, Guoqi Li, Youhui Zhang, Mingguo Zhao, and Luping Shi. A framework for the general design and computation of hybrid neural networks. 13(1):3427.

[9] NEST Development Team. *iaf_psc_alpha – Leaky integrate-and-fire model with alpha-shaped input currents*. NEST Initiative, 2023. Accessed: 2023-12-20.

[10] Seyedali Mirjalili. Genetic algorithm. In Seyedali Mirjalili, editor, *Evolutionary Algorithms and Neural Networks: Theory and Applications*, Studies in Computational Intelligence, pages 43–55. Springer International Publishing.

[11] Yangjun Ruan, Yuanhao Xiong, Sashank Reddi, Sanjiv Kumar, and Cho-Jui Hsieh. Learning to learn by zeroth-order oracle.

[12] Mountain car - gym documentation.

[13] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. 13(70):2171–2175.

[14] L2l — l2l 1.0.0-beta documentation.

[15] Introduction — neuromatch academy: Computational neuroscience.

[16] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. 113:54–71.

[17] NEST simulator.

[18] Ding Chen, Peixi Peng, Tiejun Huang, and Yonghong Tian. Deep reinforcement learning with spiking q-learning.

[19] Devdhar Patel, Hananel Hazan, Daniel J. Saunders, Hava T. Siegelmann, and Robert Kozma. Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game. 120:108–115.

[20] Xiaoyang Jiang, Qiang Zhang, Jingkai Sun, and Renjing Xu. Fully spiking neural network for legged robots.

[21] Guangzhi Tang, Neelesh Kumar, Raymond Yoo, and Konstantinos Michmizos. Deep reinforcement learning with population-coded spiking neural network for continuous control. In *Proceedings of the 2020 Conference on Robot Learning*, pages 2016–2029. PMLR.

[22] Duzhen Zhang, Tielin Zhang, Shuncheng Jia, Xiang Cheng, and Bo Xu. Population-coding and dynamic-neurons improved spiking actor network for reinforcement learning.

[23] Sérgio F. Chevtchenko and Teresa B. Ludermir. Combining STDP and binary networks for reinforcement learning from images and sparse rewards. 144:496–506.