Learning the Plasticity: Plasticity-Driven Learning Framework in Spiking Neural Networks

Guobin Shen[1,2,3]*, **Dongcheng Zhao**[1,2,4]*, **Yiting Dong**[3],
Yang Li[3], **Feifei Zhao** [3] **Yi Zeng**[1,2,3,4†]
[1]Beijing Institute of AI Safety and Governance (Beijing-AISI)
[2]Beijing Key Laboratory of Safe AI and Superalignment
[3]BrainCog Lab, CASIA [4]Long-term AI
{shenguobin2021, zhaodongcheng2016, dongyiting2020,
liyang2019, zhaofeifei2014, yi.zeng}@ia.ac.cn

Abstract

The evolution of the human brain has led to the development of complex synaptic plasticity, enabling dynamic adaptation to a constantly evolving world. This progress inspires our exploration into a new paradigm for Spiking Neural Networks (SNNs): a Plasticity-Driven Learning Framework (PDLF). This paradigm diverges from traditional neural network models that primarily focus on direct training of synaptic weights, leading to static connections that limit adaptability in dynamic environments. Instead, our approach delves into the heart of synaptic behavior, prioritizing the learning of plasticity rules themselves. This shift in focus from weight adjustment to mastering the intricacies of synaptic change offers a more flexible and dynamic pathway for neural networks to evolve and adapt. Our PDLF does not merely adapt existing concepts of functional and Presynaptic-Dependent Plasticity but redefines them, aligning closely with the dynamic and adaptive nature of biological learning. This reorientation enhances key cognitive abilities in artificial intelligence systems, such as working memory and multitasking capabilities, and demonstrates superior adaptability in complex, real-world scenarios. Moreover, our framework sheds light on the intricate relationships between various forms of plasticity and cognitive functions, thereby contributing to a deeper understanding of the brain's learning mechanisms. Integrating this groundbreaking plasticity-centric approach in SNNs marks a significant advancement in the fusion of neuroscience and artificial intelligence. It paves the way for developing AI systems that not only learn but also adapt in an ever-changing world, much like the human brain.

1 Introduction

Synaptic plasticity, characterized by the ability of synapses to strengthen or weaken over time, underpins learning and memory in the human brain. This adaptive mechanism allows for dynamic responses to an ever-evolving environment, manifesting across various levels from molecular to neural networks [1; 2; 3; 4; 5; 6; 7]. Its significance is widely recognized; however, the direct application of synaptic plasticity as an optimization algorithm in artificial neural networks presents notable challenges.

One of the key challenges arises from the diversity of learning rules observed in biology, such as Long-Term Depression (LTD), Long-Term Potentiation (LTP), and Spike-Timing-Dependent Plasticity

*Equal contribution.
†Corresponding Author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

(STDP). These mechanisms, although instrumental in understanding and implementing functions like learning and memory, are often challenging to directly apply in neural network optimization due to their complexity and subtlety [8; 9; 10; 11]. Traditional modeling methods, inspired by biological evidence, struggle to capture the inherent dynamism of neural systems and require meticulous hand-design or integration with deep learning optimization techniques.

Spiking Neural Networks (SNNs) adeptly emulate the discrete spike sequence information transmission found in biological nervous systems, intricately modeling the dynamics of biological neurons. The intrinsic event-driven and real-time nature of information processing in SNNs grants them an enhanced capability for managing tasks with temporal dynamics, outperforming traditional Artificial Neural Networks (ANNs) in these aspects [12]. While backpropagation has been established as a foundational technique in neural network optimization [13; 14], its precise analog in biological systems remains controversial, raising questions about the feasibility of replicating complex biological tasks using such algorithms [15]. Another way to optimize SNNs is to draw on plasticity mechanisms in biology. This approach replicates the challenges of complex tasks based on local synaptic plasticity rules observed in biological systems. However, enhancing the learning ability of SNN through various learning rules often relies on the coordination of manual presets and lacks the flexibility to adapt to changes in different environments [16; 17; 18; 19; 20; 21; 21], limiting the development of this field.

We argue that the main challenge in this field stems from the rigid application of observed biological plasticity mechanisms in neural network design. There is a lack of abstraction and deeper understanding of these mechanisms, leading to models that do not fully exploit the adaptability and learning potential of synaptic changes. To overcome these limitations, we propose an innovative approach that shifts the focus from traditional synaptic weight adjustments to learning the principles of plasticity. We advocate for an abstract and parametric modeling of plasticity, aiming to learn and adapt the rules of plasticity within the network. This paradigm shift aligns more closely with the dynamic nature of biological learning and opens avenues for developing more robust and adaptable neural network models.

Our contributions can be summarized as follows:

- We propose abstract and parametric modeling of biological plasticity, providing a higher level of generalization and summary of local plasticity, leading to more flexible forms and better generalization capabilities.
- We introduce the Plasticity-Driven Learning Framework (PDLF), emphasizing the understanding and application of plasticity rules over traditional synaptic weight adjustments, allowing neural networks to evolve and adapt in dynamic environments.
- This approach potentially enhances the generalization and multitasking abilities of SNNs in dynamic and real-world scenarios, offering a platform for continuous learning and adaptation, mirroring the extraordinary capabilities of biological nervous systems.

2 Related Work

Research on biologically plausible neural networks has advanced along several interconnected paths, with our work building upon and extending these foundations in novel directions.

Training Methods for SNNs. Recent advances in SNN training follow two primary paradigms: surrogate gradient techniques [22; 23; 24; 25; 26; 27] that approximate backpropagation despite non-differentiable spiking functions, and ANN-to-SNN conversion methods [28; 29] that leverage established deep learning techniques. Though effective, these approaches rely on gradient-based optimization whose biological plausibility remains controversial and confine SNNs within traditional DNN frameworks, limiting their adaptability. Biologically-inspired alternatives, including STDP-based learning [8], reward-modulated plasticity [30], and three-factor rules [31], better align with biological principles but implement fixed plasticity rules that fail to match the adaptive dynamics of biological systems while typically underperforming compared to gradient-based methods.

Adaptable Plasticity Mechanisms. Traditional implementations of synaptic plasticity in artificial systems rely on hand-designed rules with fixed parameters [32], limiting their flexibility across diverse tasks. Recent work has begun exploring meta-learning approaches for plasticity, including differentiable plasticity frameworks [33; 34; 35] and neuromodulated plasticity [36]. These approaches allow for some adaptation of plasticity rules but typically maintain a separation between weight optimization and plasticity rule optimization.

Our PDLF advances this line of research by fundamentally reorienting the learning process to focus on mastering the principles of plasticity themselves, enabling more dynamic adaptation across diverse conditions. Unlike previous approaches that primarily use plasticity as a mechanism for weight updates, our framework treats plasticity as the central object of learning, allowing for the emergence of task-appropriate adaptation mechanisms.

Memory and Adaptation in Neural Networks Working memory in biological systems involves complex interplay between neural activity and synaptic dynamics. Traditional models emphasize persistent neural activity [37], while recent theories highlight the role of short-term synaptic plasticity in maintaining information without continuous firing [38; 39].

In artificial systems, evolutionary strategies have been employed to optimize neural network parameters [40; 41] and, more recently, plasticity rules [42; 36]. However, these approaches typically optimize fixed rules rather than developing mechanisms that can continuously adapt throughout an agent's lifetime.

Our work advances these efforts by developing a parametric framework for plasticity that enables continuous adaptation through the synergistic interaction of Synaptic Cooperation Plasticity (SCP) and Presynaptic-Dependent Plasticity (PDP). This approach enhances generalization and adaptation capabilities, particularly in scenarios involving unexpected perturbations or tasks not encountered during training.

3 Method

3.1 Neuron and Synaptic Models

We employed leaky integrate-and-fire (LIF) neurons in our network models due to their biological plausibility and computational efficiency. The state of each LIF neuron was represented by its membrane potential, which integrated the incoming signals and generated a spike when the potential crossed a predefined threshold, as shown in Eq. 1.

$$\tau_m \frac{\partial v}{\partial t} = -(v - v_0) + I(t) \tag{1}$$

In Eq. 1, τ_m is the membrane time constant, v is the membrane potential, v_0 is the resting potential, and $I(t)$ is the total synaptic current at time t. Once the membrane potential v exceeds a certain threshold v_{th}, the neuron generates a spike, and the potential is reset. A discrete form of the LIF neuron's behavior can be described as:

$$\begin{aligned}
&u(t) = v(t-\Delta t) + \frac{\Delta t}{\tau_m}(\sum_i w_i s_i(t) - v(t-\Delta t) + v_0) \\
&s(t) = g(u(t) - v_{th}) \\
&v(t) = u(t)(1 - s(t)) + v_{reset} s(t)
\end{aligned} \tag{2}$$

In Eq. 2, Δt is the time step, v_{reset} is the reset potential, $u(t)$ and $v(t)$ represent pre- and post-spike membrane potentials, and $g(\cdot)$ is the Heaviside function modeling spiking behavior. After the loss of the reward signal, updating the network weights stops. This strategy of directly optimizing the weights is set as a control group in our experiments. Neuronal parameters are given in Tab. 3 unless otherwise specified.

Traces are the tracks produced at the pre- and post-synaptic sites by the spikes of pre- or post-synaptic neurons. Generally, these traces represent the recent activation level of pre- and post-synaptic neurons [32]. Traces can be computed by integrating spikes using a linear operator in the model and a low-pass filter in the circuit or by using non-linear operators/circuits. In the experiments, the synaptic traces were modeled as follows:

$$x(t) = \sum_{\tau=0}^{t} \lambda^{t-\tau} s(\tau) \tag{3}$$

In Eq. 3, $x(t)$ is the synaptic trace at time t, λ is the decay factor reflecting how quickly a spike's influence fades with time, and $s(\tau)$ represents the spike at time τ. In the context of our experiment, these synaptic traces maintain a short-term history of neuron activation, thereby adding an element of temporal dynamics to our network model. As shown in Eq. 4, these synaptic traces are used to maintain a short-term history of neuronal activation and, in conjunction with PDLF, to modulate synaptic weights.

3.2 Plasticity-Driven Learning Framework

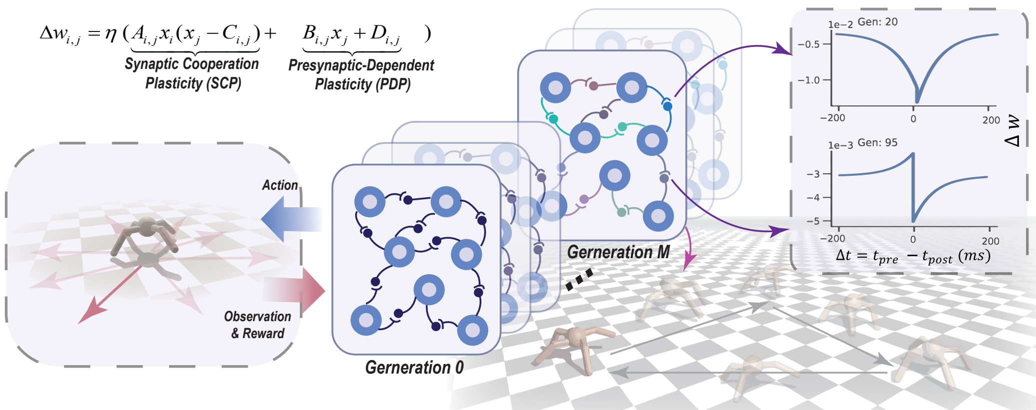

Figure 1: **Diagram of PDLF.** Top: By combining Synaptic Cooperation Plasticity (SCP) and Presynaptic-Dependent Plasticity (PDP), neurons can achieve diverse and heterogeneous plasticity. Bottom: Agents with PDLF learn plasticity rather than directly adjusting weights. Different forms of synaptic plasticity can be formed between neurons, enabling better multi-task learning. Plasticity helps the agents dynamically adjust weights and learn previously unseen scenarios during training, even without explicit reward signals.

The realm of Spiking Neural Networks (SNNs) has witnessed the identification of multiple local plasticity mechanisms, such as Spike-Timing-Dependent Plasticity (STDP) [3] and Bienenstock-Cooper-Munro (BCM) rules [43]. However, abstracting and refining these biological observations into concrete plasticity mechanisms suitable for diverse complex tasks remains a formidable challenge. To circumvent excessive manual design and simple combination of different plasticity mechanisms, we introduce a more abstract level of plasticity: a parametric plasticity framework.

Adhering to the fundamental rule of local plasticity - that synaptic strength is influenced by pre- and post-synaptic neural activity - we design our framework based on a parametric expansion of the Spiking BCM rule [44]. The PDLF comprises two key components: Synaptic Cooperation Plasticity (SCP) and Presynaptic-Dependent Plasticity (PDP). SCP dynamically adjusts synaptic strength by considering the activity of both pre- and post-synaptic neurons. In contrast, PDP adjusts based on pre-synaptic activity alone and introduces a bias to synaptic changes for stability. The synaptic weight update in our framework is formulated as follows:

$$\Delta w_{i,j} = \eta\,(\underbrace{A_{i,j}x_i(x_j - C_{i,j})}_{\text{Synaptic Cooperation Plasticity (SCP)}} + \underbrace{B_{i,j}x_j + D_{i,j}}_{\text{Presynaptic-Dependent Plasticity (PDP)}}) \tag{4}$$

In Eq. 4, $\Delta w_{i,j}$ represents the change in synaptic weight between neurons i and j. x_i and x_j represent the spike traces [32] of pre- and post-synaptic neurons respectively. The parameters $A_{i,j}$, $B_{i,j}$, $C_{i,j}$, and $D_{i,j}$ are learnable, enabling the network to form distinct and adaptable plasticity rules.

SCP, represented by the term $A_{i,j}x_i(x_j - C_{i,j})$, adjusts synaptic strength based on the temporal correlation of neural activities, with $C_{i,j}$ acting as a threshold for post-synaptic activity. PDP, denoted by $B_{i,j}x_j + D_{i,j}$, modifies synaptic weights based on pre-synaptic activity, with $D_{i,j}$ providing a stable bias for each neuron. The learning rate η scales the overall synaptic weight change.

To optimize these parameters, we employ an Evolutionary Strategy (ES) [41], inspired by the natural selection processes shaping biological organisms. In this context, the parameters of the plasticity-centric learning rule can be viewed as intrinsic priors, optimized throughout the evolutionary process to ensure survival and adaptation. The ES involves a population of agents, each with a unique set of plasticity parameters, with their fitness evaluated based on adaptability and performance in various tasks. Through this process, the parameters are optimized, enabling agents to maintain flexible and dynamic adaptation throughout their lifespan.

4 Results

4.1 PDLF Enhances Working Memory Capacity

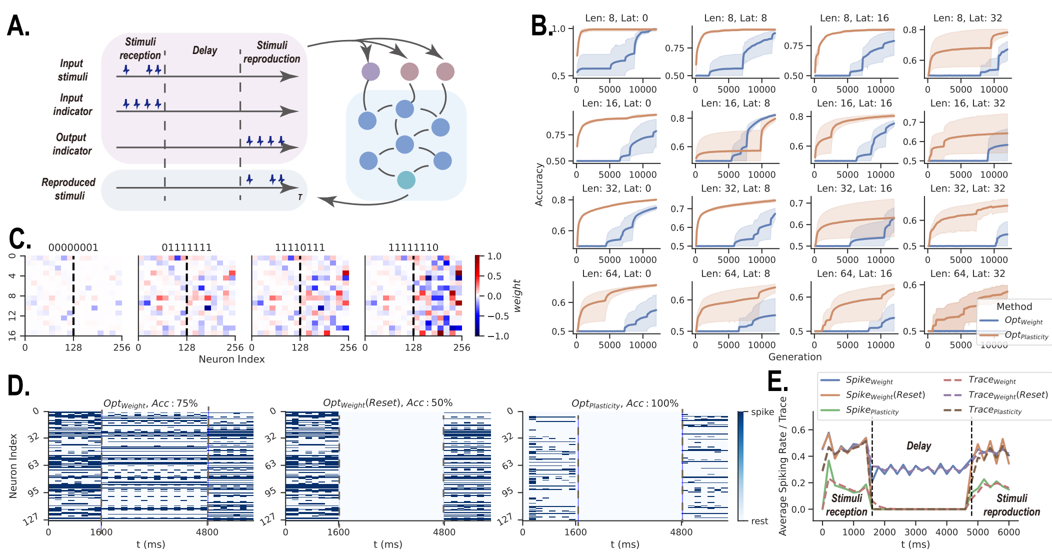

Figure 2: **PDLF impact on Working Memory. A.** Copying task schematic: SNNs receive motion stimuli (200ms each), followed by a variable delay period, then reproduce stimuli upon receiving a test signal. **B.** Performance comparison: SNNs with plasticity show faster convergence, longer memory duration, and greater capacity. 'Len'=stimulus length, 'Lat'=delay steps. **C.** Distinct synaptic weights formed for different stimuli with PDLF (input weights left of dashed line, output weights right). **D.** Neuron states comparison: directly trained SNNs require sustained activity during delay; PDLF-based SNNs encode stimuli in synaptic weights, preserving memory even after membrane potential reset. **E.** Firing rates: SNNs with PDLF maintain lower firing rates across stages.

In this section, we aim to demonstrate the effect of PDLF on Working Memory (WM). WM is the ability to maintain and process information temporarily and is the cornerstone of higher intelligence [45].

We explore the significant impact of PDLF on enhancing the WM capabilities of SNNs. We utilize a task known as the copying task [46], as illustrated in Fig. 2A. In this task, SNNs are initially presented with a sequence of stimuli, each lasting for 200 ms. This is then followed by a delay period of variable lengths, and finally, a test stimuli of equivalent duration to the sample stimuli is presented. The challenge for the SNNs lies in accurately reproducing the initial sequence of stimuli in the correct order upon receiving the test stimuli.

To thoroughly demonstrate the advantages of employing a PDLF-based approach in working memory tasks, we carry out a comparison with the strategy of directly optimizing weights utilizing the ES based on widely used three-layer SNNs. As shown in Fig. 2B, SNNs equipped with PDLF exhibit faster convergence rates, an ability to retain memory over longer durations, and an enhanced memory capacity.

To further investigate the influence of PDLF, we visualize synaptic weights following various stimuli inputs when the stimuli length is set to 8. As shown in Fig. 2C, SNNs endow with PDLF can form distinct connection weights for different stimuli, demonstrating their superior adaptive capacity.

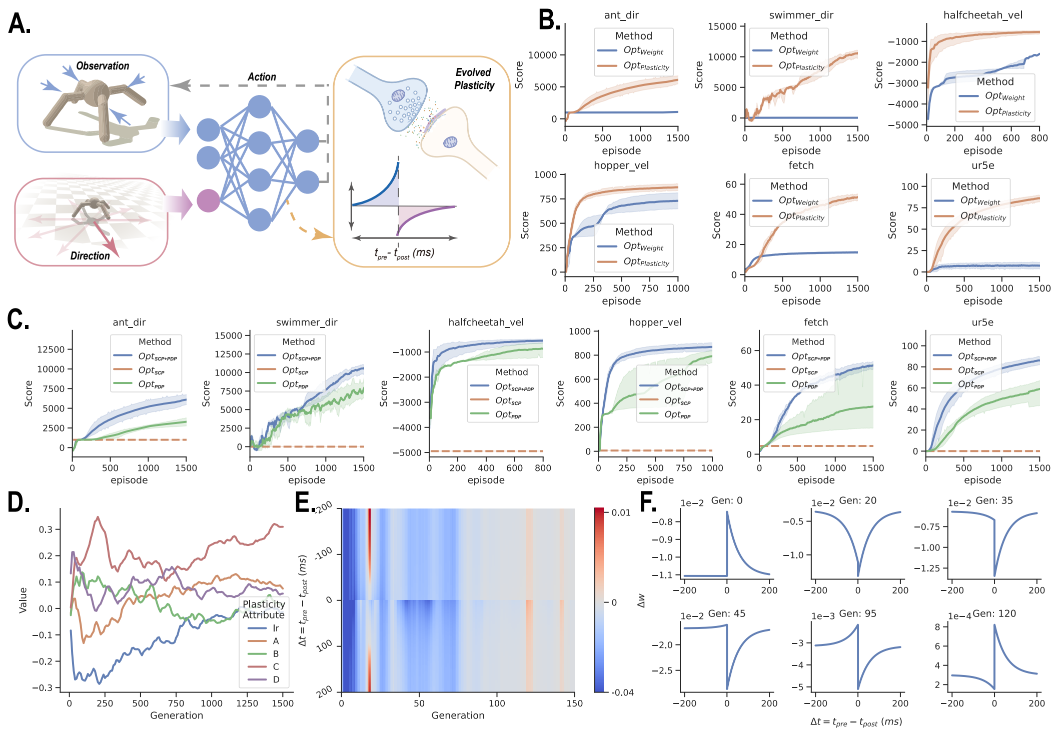

Figure 3: **PDLF in multi-task RL and plasticity analysis. A.** Multi-task RL setup: single network learns tasks with distinct/opposing objectives, with task goals as input observations. **B.** Training curves comparing PDLF versus directly trained weights across direction tasks (`ant_dir`, `swimmer_dir`), velocity tasks (`halfcheetah_vel`, `hopper_vel`), and target location tasks (`fetch`, `ur5e`). PDLF enables superior multi-task performance through dynamic weight adaptation. **C.** Ablation study showing both SCP and PDP components are essential; removing PDP causes divergence due to loss of equilibrium. **D.** Evolution of synaptic plasticity parameters during training. **E.** Impact of plasticity on weights at different inter-spike intervals across training generations. **F.** Functional characteristics of plasticity across generations shown in **E**.

We compare the neuronal states at various stages between SNNs directly trained with weights and those incorporating PDLF, as illustrated in Fig. 2D. SNNs directly trained with weights rely on neuronal activity during the delay period to maintain memory. Resetting the membrane potential to 0 after the stimulus input leads to memory loss for input stimuli. This indicates that in SNNs directly optimized with weights, memory is primarily stored in neuronal activity. In contrast, SNNs incorporating PDLF encode the input stimulus into synaptic weights, demonstrating remarkable memory capabilities. Their ability to adjust synaptic weights facilitates the enhancement of working memory and allows neurons to remain in a resting state when not receiving task-related stimuli. This contributes to network efficiency and enhances network capacity, which has been validated in other biological and computational neuroscience studies [38; 39].

Finally, we visualize the average spike traces of SNNs under different training strategies, as shown in Fig. 2E. Notably, SNNs incorporating PDLF can sustain lower firing rates, thereby enhancing their efficiency in managing computational resources.

In summary, these results highlight the significant role of PDLF in bolstering the working memory capacity of SNNs, demonstrating its potential for facilitating complex cognitive tasks in artificial intelligence systems.

4.2 PDLF Enhances Multi-task Learning

Reinforcement Learning (RL) involves an agent learning to interact with its environment to achieve a specific goal, with the quality of its actions dictated by a reward signal. Conventional RL approaches often entail training the agent on a single task with a fixed reward function. However, in complex, real-world environments, agents need to handle multiple tasks and adjust to changing reward functions. This multi-task learning scenario presents significant challenges, mainly when the tasks involved are diverse and potentially conflicting.

In the field of RL, some of the most challenging problems lie within the domain of continuous control, usually modeled using sophisticated physics engines. These tasks require agents to manipulate simulated physical entities with high precision and coordination, similar to how humans control their limbs to carry out complex tasks. We use the Brax [47] simulator to design six continuous control environments. In these settings, agents need to navigate at various speeds, directions, and destination points. As shown in Fig. 3A, different task objectives are treated as observations to guide the agent in accomplishing different tasks. These challenging tasks serve as baselines in fields such as meta-learning [48; 49; 50]. During training, they are only exposed to a limited number of task instances, such as eight specific directions or eight fixed speeds. They use a single network to learn these unrelated or conflicting tasks.

Our experiments compare two types of SNNs - one with synaptic weights that have been directly optimized and another with optimized PDLF. Both types of SNNs maintain the same scale and structure to ensure a fair comparison. Through this, we aim to highlight the advantages of optimizing PDLF over direct weight optimization in a multi-task environment. This comparison also evaluates the effectiveness and potential of our proposed model, especially in the face of the inherent challenges posed by complex continuous control tasks.

Table 1: Comparison of performance across various reinforcement learning tasks with different configurations of synaptic plasticity. Each task is evaluated using different training methods: Plasticity$_{SCP+PDP}$, Plasticity$_{SCP}$, Plasticity$_{PDP}$, and direct weight training. The values presented are the mean and standard deviation over 5 trials. The row 'Chance Level' shows the performance metrics when randomly chooses actions. * Upon the removal of Presynaptic-Dependent Plasticity (PDP), the SNNs become unstable, leading to divergence.

Training	ant_dir	swimmer_dir	halfcheetah_vel	hopper_vel	fetch	ur5e
Opt$_{SCP+PDP}$	6904 ± 801	10531 ± 827	-549 ± 95	869 ± 38	51 ± 3	86 ± 5
Opt$_{PDP}$	3284 ± 570	7831 ± 1479	-870 ± 312	792 ± 50	26 ± 26	58 ± 13
Opt$^*_{SCP}$	-	-	-	-	-	-
Opt$_{Weight}$	1069 ± 98	31 ± 8	-1598 ± 324	729 ± 116	15 ± 0.6	7 ± 5
Chance Level	995 ± 0.01	0.12 ± 0.02	-4946 ± 1.3	6.51 ± 0.3	4.74 ± 0.0	0 ± 0.0

We explore the performance of PDLF in a three-layer, fully-connected SNN model with 128 hidden spiking neurons. Both the synaptic weights and the plasticity parameters are initialized to 0. During testing, their plasticity rules are fixed for agents with plasticity, and synaptic weights are reset to 0. The trained weights are applied during testing for agents trained directly on weights. We utilize reinforcement learning tasks to thoroughly test PDLF, requiring the agents to learn to cope with different tasks simultaneously and generalize the acquired knowledge to unseen, more complex tasks.

In our experiments, agents with PDLF show superior performance in multiple tasks compared to agents with directly optimized synaptic weights. As seen in Fig. 3B and Tab. 1, agents with PDLF exhibit more effective learning curves. The agents with PDLF quickly adapted to the changes in tasks, making them more capable of tackling the multi-task challenges inherent in the designed environments. In contrast, SNNs with directly trained weights fail to adapt to different tasks and can only acquire trivial solutions, such as maintaining approximate immobility in tasks involving multiple target directions.

To further validate the effectiveness of our approach, we compare PDLF against Meta-Hebb [5], a state-of-the-art meta-learning approach for ANNs that also learns plasticity rules. As shown in Table 2, PDLF demonstrates superior performance across multiple tasks, highlighting the advantages of learning plasticity rules within the SNN framework.

Table 2: Comparison with meta-learning baselines on multi-task RL environments. Values represent mean ± standard deviation over 5 trials.

Architecture	Method	`ant_dir`	`swimmer_dir`	`halfcheetah_vel`	`hopper_vel`
ANN	Meta-Hebb	5646 ± 543	7605 ± 891	-735 ± 124	794 ± 67
SNN	**PDLF**	$\mathbf{6904 \pm 801}$	$\mathbf{10531 \pm 827}$	$\mathbf{-549 \pm 95}$	$\mathbf{869 \pm 38}$

Ablation studies, depicted in Fig. 3C and Tab. 1, provide further insights into the contributions of different plasticity mechanisms. Removing any form of plasticity results in decreased agent performance, with the removal of PDP causing a divergence due to the loss of the equilibrium mechanism. This result highlights the importance of all plasticity mechanisms in maintaining the stability and adaptability of the agents.

The changing curve of a synapse's PDLF during training (Fig. 3D), along with the impact of plasticity on weights for different generations of agents (Fig. 3E), demonstrate the effective learning of optimal parameters for the PDLF rule, facilitated by the evolutionary strategy. Interestingly, the specific functions of plasticity across different generations of agents differed (Fig. 3F), indicating the evolution and fine-tuning of plasticity mechanisms to improve agent performance across generations.

4.3 PDLF Enhances Generalization Ability

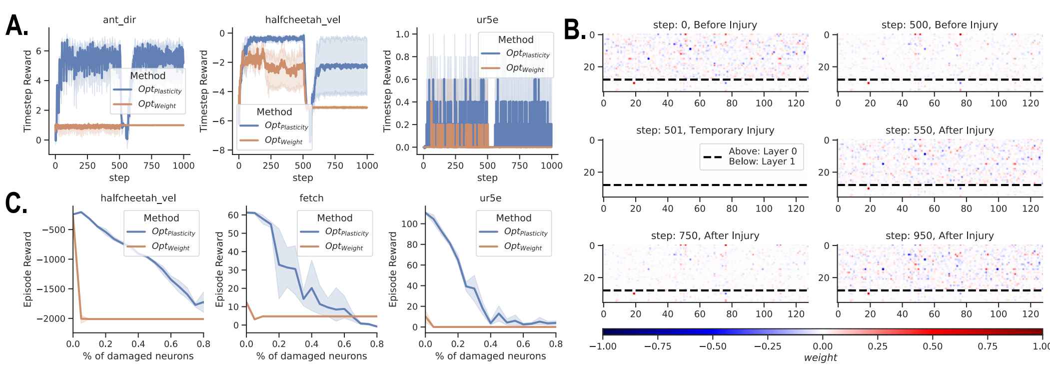

Figure 4: **Robustness against neural injuries. A.** Recovery from temporary damage: When weights reset to 0 at step 500 for 50 steps, PDLF agents recover functionality, showing superior robustness. **B.** Weight visualization before/after temporary damage: Input layer weights (above dashed line) and readout layer weights (below) show recovery based on plasticity and input stimuli despite complete weight loss. **C.** Response to permanent damage: With varying proportions of neurons blocked (weights fixed at 0), PDLF agents maintain better performance, demonstrating enhanced adaptability to irreversible neural impairment.

PDLF serves as an important mechanism for enhancing an agent's generalization abilities, enabling the agent to exhibit stronger performance when dealing with unfamiliar tasks or when facing neuronal damage.

We further investigate the performance of PDLF when the agents faced injuries. We design two different types of injuries: temporary injuries and permanent injuries. Temporary injuries refer to a scenario where all synaptic weights of the agents are reset to 0 and kept for 50 steps. Permanent injuries refer to a situation where some synapses are set to 0 initially and do not update according to plasticity. In the tests for injuries, agents with plasticity display a remarkable ability to recover from temporary neuronal damage simulated by resetting all synaptic weights to 0 for 50 steps (Fig. 4A). Fig. 4B illustrates the changes in network weights at various stages before and after the temporary damage. Remarkably, despite losing all synaptic weights due to the inflicted temporary damage, agents manage to recover these weights using their inherent plasticity and incoming input stimuli. Moreover, even under permanent neuronal damage, with a proportion of neurons blocked and their weights unable to update, the plasticity-enabled agents continue to exhibit better performance and robustness (Fig. 4C). These results suggest that PDLF can contribute to the resilience of artificial agents, much as it does in biological systems.

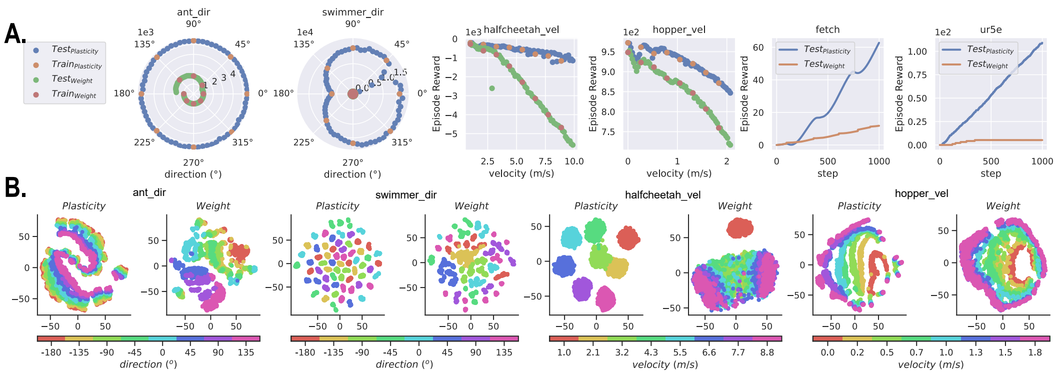

Figure 5: **A.** Performance of different agents in trained tasks and tasks not seen during training. Agents with plasticity can generalize well to unseen tasks, while agents trained directly on weights have difficulty generalizing to unseen test tasks due to their weights being fixed during testing. **B.** Low-dimensional embeddings of neuronal states during reinforcement learning tasks, differentiated by training strategy. Each point corresponds to the state of the hidden layer neurons at a specific time step. The color coding signifies distinct tasks. Agents that possess plasticity demonstrate an enhanced capability to distinguish between different tasks. Moreover, the neuronal states associated with identical tasks exhibit the intriguing property of forming a manifold within the high-dimensional space.

More strikingly, agents with PDLF show a robust ability to generalize to tasks unseen during training, as demonstrated in Fig. 5A. For tasks with various movement directions and speeds, the agents only encounter a small subset of cases during training, represented by the red and orange points in Fig. 5A. Therefore, in this experiment, the agents are required to move in directions and at speeds unseen in training, emphasizing the agents' more profound understanding and generalization capacity for the tasks. In contrast, agents trained directly on weights struggle with generalization due to their fixed weights during testing. This observation underscores the flexible adaptability offered by PDLF, enhancing the agent's ability to navigate unseen scenarios.

During the training phase, the agents are instructed to move in straight lines in eight specific directions. However, as illustrated in Fig. 5A, agents with PDLF demonstrate a degree of generalization capability. They can learn to move in straight lines toward directions not encountered during training, while agents without PDLF struggle to generalize what they have learned.

To further test the generalization capacity introduced by PDLF, we hope that agents could learn to turn or even form more complex paths simply by changing the target signal and without any additional feedback information related to posture. The results are shown in Fig. 6. Compared to agents that directly train their weights, those with PDLF demonstrate impressive generalization abilities toward this complex task. They can quickly adjust synaptic weights through PDLF and dynamically modify their state in previously unseen scenarios during different pieces of training. This allows them to progress toward varying target directions.

In Fig. 5B, we visualize the neuronal states of agents trained using different strategies during RL tasks in low-dimensional space. Each point in this representation corresponds to the state of the hidden layer neurons at a particular time step, with the varying colors indicating different tasks. The remarkable aspect of these visualizations is how agents with inherent plasticity demonstrate a pronounced ability to discern between distinct tasks. Even more intriguing is the observation that the neuronal states corresponding to the same task tend to cluster together, forming a clear manifold in the high-dimensional space. This feature of PDLF contributes to the agent's robust ability to

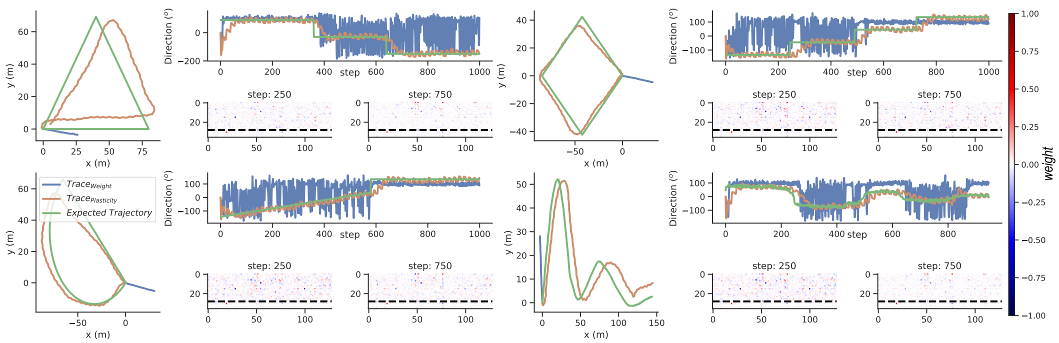

Figure 6: **Testing of the agent's generalization capabilities.** During training, the agent only learned to move in a straight line. The agent's movement trajectories are shown when the target direction is altered during the testing process. The green line represents the expected trajectory of the agent moving at a constant speed. The orange line is the actual movement trajectory of the agent with plasticity. Agents with plasticity can better understand different tasks, adjust synaptic weights according to different target directions, and thus show greater flexibility and superior generalization performance.

generalize across various tasks while maintaining distinctive task-specific patterns in neuronal states. This further illustrates the powerful capabilities of agents with PDLF and the profound impact of such plasticity on the agent's learning and adaptation abilities.

5 Discussion

Current AI algorithms excel at specific tasks but struggle when conditions deviate from training scenarios, limiting their adaptability in complex environments. In contrast, biological systems demonstrate exceptional adaptability through synaptic plasticity, the cornerstone of generalized intelligence [51; 52].

While SNNs offer brain-inspired mechanisms for building flexible intelligent systems [53; 54; 55; 56], current training methods—whether backpropagation or fixed plasticity rules like STDP—limit generalization in multi-task environments. Our PDLF addresses this through the synergy of SCP and PDP, enabling dynamic plasticity rule adjustment rather than static weight modification. This higher-order learning process fosters adaptive synaptic modifications based on neuronal activity history, narrowing the gap between artificial and biological systems.

Our results demonstrate that PDLF enhances memory capacity by encoding information in synaptic weights rather than sustained neural activity, improving energy efficiency. It enables superior multi-task learning, generalization to unseen tasks, and adaptation to novel scenarios not encountered during training. Critically, PDLF provides robustness under both temporary and permanent neural damage, mirroring biological recovery mechanisms through neural reorganization—a vital feature for deployment in dynamic, unpredictable real-world environments.

In conclusion, our study highlights PDLF as a critical feature that enhances the resilience and adaptability of artificial agents. These findings provide valuable insights for designing future artificial systems, opening up new possibilities for creating adaptive, robust, and intelligent agents capable of navigating complex and dynamic environments. Further work can explore more sophisticated forms of PDLF and study their impacts on various facets of artificial agent performance.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62406325).

References

[1] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[2] Elie L Bienenstock, Leon N Cooper, and Paul W Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32–48, 1982.

[3] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.

[4] Gina G Turrigiano. Homeostatic plasticity in neuronal networks: the more things change, the more they stay the same. *Trends in neurosciences*, 22(5):221–227, 1999.

[5] Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. *Advances in Neural Information Processing Systems*, 33:20719–20731, 2020.

[6] Fangxin Liu, Wenbo Zhao, Yongbiao Chen, Zongwu Wang, Tao Yang, and Li Jiang. Sstdp: Supervised spike timing dependent plasticity for efficient spiking neural network training. *Frontiers in Neuroscience*, 15:756876, 2021.

[7] Fangxin Liu, Wenbo Zhao, Zongwu Wang, Yongbiao Chen, Tao Yang, Zhezhi He, Xiaokang Yang, and Li Jiang. Sato: spiking neural network acceleration via temporal-oriented dataflow and architecture. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1105–1110, 2022.

[8] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[9] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, and Timothée Masquelier. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing*, 205:382–392, 2016.

[10] Priyadarshini Panda and Kaushik Roy. Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition. In *2016 international joint conference on neural networks (IJCNN)*, pages 299–306. IEEE, 2016.

[11] Yu Duan, Zhongfan Jia, Qian Li, Yi Zhong, and Kaisheng Ma. Hebbian and gradient-based plasticity enables robust memory and rapid learning in rnns. *arXiv preprint arXiv:2302.03235*, 2023.

[12] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[13] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. 323(6088):533–536.

[14] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[15] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[16] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[17] Yiting Dong, Dongcheng Zhao, Yang Li, and Yi Zeng. An unsupervised stdp-based spiking neural network inspired by biologically plausible learning rules and connections. *Neural Networks*, 2023.

[18] Dongcheng Zhao, Yi Zeng, Tielin Zhang, Mengting Shi, and Feifei Zhao. Glsnn: A multi-layer spiking neural network based on global feedback alignment and local stdp plasticity. *Frontiers in Computational Neuroscience*, 14:576841, 2020.

[19] Fangxin Liu, Zongwu Wang, Wenbo Zhao, Yongbiao Chen, Tao Yang, Xiaokang Yang, and Li Jiang. Randomize and match: Exploiting irregular sparsity for energy efficient processing in snns. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pages 451–454. IEEE, 2022.

[20] Fangxin Liu, Haomin Li, Ning Yang, Zongwu Wang, Tao Yang, and Li Jiang. Teas: Exploiting spiking activity for temporal-wise adaptive spiking neural networks. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 842–847. IEEE, 2024.

[21] Fangxin Liu, Zongwu Wang, Wenbo Zhao, Ning Yang, Yongbiao Chen, Shiyuan Huang, Haomin Li, Tao Yang, Songwen Pei, Xiaoyao Liang, et al. Exploiting temporal-unrolled parallelism for energy-efficient snn acceleration. *IEEE Transactions on Parallel and Distributed Systems*, 35(10):1749–1764, 2024.

[22] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[23] Guobin Shen, Dongcheng Zhao, and Yi Zeng. Exploiting high performance spiking neural networks with efficient spiking patterns. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2025.

[24] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[25] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[26] Guobin Shen, Dongcheng Zhao, Tenglong Li, Jindong Li, and Yi Zeng. Are conventional snns really efficient? a perspective from network quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27538–27547, 2024.

[27] Guobin Shen, Dongcheng Zhao, and Yi Zeng. Exploiting nonlinear dendritic adaptive computation in training deep spiking neural networks. *Neural Networks*, 170:190–201, 2024.

[28] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66, 2015.

[29] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

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

[31] Łukasz Kuśmierz, Takuya Isomura, and Taro Toyoizumi. Learning with three factors: modulating hebbian plasticity with errors. *Current opinion in neurobiology*, 46:170–177, 2017.

[32] Jean-Pascal Pfister and Wulfram Gerstner. Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, 26(38):9673–9682, 2006.

[33] Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018.

[34] Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *arXiv preprint arXiv:2002.10585*, 2020.

[35] Guobin Shen, Dongcheng Zhao, Yiting Dong, and Yi Zeng. Brain-inspired neural circuit evolution for spiking neural networks. *Proceedings of the National Academy of Sciences*, 120(39):e2218173120, 2023.

[36] Andrea Soltoggio, Kenneth O Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108:48–67, 2018.

[37] Xiao-Jing Wang. Synaptic reverberation underlying mnemonic persistent activity. *Trends in neurosciences*, 24(8):455–463, 2001.

[38] Gianluigi Mongillo, Omri Barak, and Misha Tsodyks. Synaptic theory of working memory. *Science*, 319(5869):1543–1546, 2008.

[39] Nicolas Y Masse, Guangyu R Yang, H Francis Song, Xiao-Jing Wang, and David J Freedman. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nature neuroscience*, 22(7):1159–1167, 2019.

[40] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[41] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[42] Stefan Schliebs and Nikola Kasabov. Evolving spiking neural network—a survey. *Evolving Systems*, 4:87–98, 2013.

[43] Julijana Gjorgjieva, Claudia Clopath, Juliette Audet, and Jean-Pascal Pfister. A triplet spike-timing–dependent plasticity model generalizes the bienenstock–cooper–munro rule to higher-order spatiotemporal correlations. *Proceedings of the National Academy of Sciences*, 108(48):19383–19388, 2011.

[44] Trevor Bekolay, Carter Kolbeck, and Chris Eliasmith. Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks. In *Proceedings of the annual meeting of the cognitive science society*, volume 35, 2013.

[45] Alan D. Baddeley and Graham Hitch. Working memory. volume 8 of *Psychology of Learning and Motivation*, pages 47–89. Academic Press.

[46] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[47] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021.

[48] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[49] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[50] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[51] Xu Liu, Steve Ramirez, Petti T Pang, Corey B Puryear, Arvind Govindarajan, Karl Deisseroth, and Susumu Tonegawa. Optogenetic stimulation of a hippocampal engram activates fear memory recall. *Nature*, 484(7394):381–385, 2012.

[52] Stephen J Martin, Paul D Grimwood, and Richard GM Morris. Synaptic plasticity and memory: an evaluation of the hypothesis. *Annual review of neuroscience*, 23(1):649–711, 2000.

[53] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[54] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[55] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[56] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2):1–35, 2019.

[57] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's contributions, including our proposed Plasticity-Driven Learning Framework (PDLF), which shifts focus from weight adjustment to learning plasticity principles. The experimental results support these claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix C.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The manuscript does not contain theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Appendices A and B we describe in detail the settings required to reproduce the experiment, and for the datasets and key code that will be released with this manuscript, we have provided it in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case

of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code will be provided in the supplementary material. The data sets involved in the experiments are all open data sets.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed training details and evaluation metrics are provided in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For all experiments, error bars are reported. These represent standard deviation over multiple runs as shown in our tables and figures.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed hardware facilities and computational overhead are provided in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: This manuscript does not involve direct human subjects; all data used are from publicly available datasets that comply with the ethical standards of NeurIPS.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Detailed social impact is discussed in the Conclusion and Appendix C

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There is no relevant risk. Our work focuses on a learning framework for spiking neural networks that does not pose risks of harmful misuse or dual-use applications.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We do cite all the existing assets in our paper as well as in our codebase.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs were not used as a component of the core methods in this research. They were only used for auxiliary tasks such as language editing and not for developing the Plasticity-Driven Learning Framework or conducting experiments.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Experimental Settings

Table 3: Parameters of the spiking neurons.

Parameter	Value		Description
	WM task	RL task	
Δt	20 ms	200 ms	Simulation time step
τ_m	40 ms	400 ms	Membrane time constant
λ	54 ms	544 ms	Decay factor
v_{th}	0.1 V	0.1 V	Membrane threshold
v_{reset}	0 mV	0 mV	Reset potential
v_0	0 mV	0 mV	Resting potential

A.1 Working Memory Task

To validate the impact of PDLF on working memory, we designed a working memory task. The agent would first receive a stimulus sequence, and after a delay of m steps, the agent is asked to reproduce the received stimulus. In each experiment, a random sequence of length n would be generated, where $r_t \sim \mathcal{B}(1, \frac{1}{2}), 1 < t \leq n$. At each time step, the input is a three-dimensional vector $\vec{a_t}$, which can be divided into three stages:

- Stimulus reception: If $1 < t \leq n$, $\vec{a_t} = (r_t, 1, 0)$. The first element is the type of input stimulus, and the second element is the indicator for the input stimulus.
- Delay: If $n + 1 < t \leq n + m$, $\vec{a_t} = (0, 0, 0)$. This phase represents a delay period where no new stimulus is presented.
- Stimulus reproduction: If $n + m + 1 < t \leq 2n + m$, $\vec{a_t} = (0, 0, 1)$. The last element indicates whether a pulse needs to be reproduced.

At each step, the model has a scalar output s_t, which is a prediction for the stimulus. The Mean Square Error over the last m steps is taken as the reward of the model:

$$R = -\frac{1}{n} \sum_{\tau=1}^{n} (r_t - s_{m+n+\tau})^2 \tag{5}$$

Eq. 5 is used as a reward function in training. To intuitively compare agents with different strategies, as shown in Fig. 2B, we utilize the average accuracy per step as the performance measure during testing, as shown in Eq. 6.

$$\text{Acc} = \frac{1}{n} \sum_{\tau=1}^{n} (r_t == s_{m+n+\tau}) \tag{6}$$

During the stimulus reception stage, each stimulus follows the distribution $\mathcal{B}(1, \frac{1}{2})$, which means that the average accuracy at the chance level is 0.5.

A.2 Multi-task Reinforcement Learning

We evaluated our method on five continuous control environments based on the Brax simulator (`ant_dir`, `swimmer_dir`, `halfcheetah_vel`, `hopper_vel`, `ur5e`, `fetch`).

- `ant_dir`: We train an ant agent to run in a target direction in this environment. The training task set includes 8 directions, uniformly sampled from $[0, 360]$ degrees. As shown in Fig. 3D, the generalization test task set includes 72 directions, uniformly sampled from $[0, 360]$ degrees. The agent's reward comprises speed along the target direction and control cost.

- `swimmer_dir`: In this environment, we train a swimmer agent to move in a fixed direction. The settings for training and testing tasks are similar to `ant_dir`.
- `halfcheetah_vel`: In the `halfCheetah_vel` environment, we train a half-cheetah agent to move forward at a specific speed. The training tasks include 8 speeds, uniformly sampled from [1, 10] m/s. The generalization test tasks include 72 different speeds, uniformly sampled from the same range as the training tasks.
- `hopper_vel`: In the `hopper_vel` environment, we train a hopper agent to advance at a specific speed. The experimental setup is the same as `halfcheetah_vel`, but the sampling interval for the speed is $[0, 2]$ m/s.
- `ur5e`: The UR5e is a common 6-DOF (degrees of freedom) robotic arm frequently used in industrial automation and robotics research. The agent receives a reward when the distance between the robotic arm's end and the target position is less than 0.02 m. The target position is then randomly reset. The agent's goal is to reach the target position as many times as possible within the stipulated time.
- `fetch`: We train a dog agent to run to a target location in this environment. The experimental setup is similar to `ur5e`.

The agent's final reward is the average reward across all tasks, which encourages the agent to learn multiple tasks simultaneously.

B Training Strategies

Algorithm 1 Parameter-Exploring Policy Gradients (PEPG)

Initialize the number of generations M, and the population size N
Initialize policy parameters θ
Initialize adaptive noise scaling parameters σ
Initialize learning rates α_θ, α_σ
Initialize Adam parameters $m_\theta, v_\theta, m_\sigma, v_\sigma, \beta_1, \beta_2, \epsilon$
Initialize noise standard deviation σ_0
for $m = 1$ to M **do**
  for $n = 1$ to N **do**
    Sample noise $\epsilon \sim \mathcal{N}(0, \sigma_0)$
    Compute offspring $\theta' = \theta + \sigma \odot \epsilon$
    Evaluate fitness $f(\theta')$
  end for
  Compute fitness baseline $b = mean(f(\theta'))$
  Compute gradients $\nabla_\theta = \frac{1}{N}\sum_{i=1}^{N} f_i \cdot \epsilon_i$
  Compute adaptive noise scaling gradient $\nabla_\sigma = \frac{1}{2N}\sum_{i=1}^{N}((f_i - b)^2 - \sigma^2)$
  Update Adam parameters for θ: $m_\theta = \beta_1 m_\theta + (1-\beta_1)\nabla_\theta$, $v_\theta = \beta_2 v_\theta + (1-\beta_2)\nabla_\theta^2$
  Update policy parameters $\theta = \theta + \alpha_\theta \cdot \frac{m_\theta}{\sqrt{v_\theta}+\epsilon}$
  Update Adam parameters for σ: $m_\sigma = \beta_1 m_\sigma + (1-\beta_1)\nabla_\sigma$, $v_\sigma = \beta_2 v_\sigma + (1-\beta_2)\nabla_\sigma^2$
  Update adaptive noise scaling $\sigma = \sigma \exp(\alpha_\sigma \cdot \frac{m_\sigma}{\sqrt{v_\sigma}+\epsilon})$
end for

We employ Parameter-Exploring Policy Gradients (PEPG) [41] to optimize SNNs. For SNNs with plasticity, the plasticity parameters in Eq. 4 are used for optimization. Evolution across generations is facilitated by modifying synaptic plasticity rules rather than directly adjusting the weights. SNNs with directly trained weights are considered a control group, where synaptic weights are the optimization parameters. The implementation of PEPG used in the experiments is provided by Algorithm 1. Unless expressly stated otherwise, the parameter settings and their explanations are shown in Table 4. The way to compute fitness $f(\theta)$ varies depending on the task. For the working memory task, fitness is provided by Eq. 5, while for multi-task reinforcement learning, fitness is the average episodic reward across different subtasks.

All experiments were conducted on a server equipped with 8 NVIDIA A100 GPUs, each with 40 GB of memory. The implementations were carried out using the JAX framework [57]. For tasks like

Table 4: Parameters in PEPG.

Parameter	**Value**	**Description**
θ	0	Initial policy parameters
σ	0.1	Initial adaptive noise scaling parameters
α_θ	0.15	Learning rate for policy parameters
α_σ	0.1	Learning rate for adaptive noise scaling
m_θ, v_θ	$0, 0$	Initial Adam parameters for policy parameters
m_σ, v_σ	$0, 0$	Initial Adam parameters for adaptive noise scaling
β_1, β_2	$0.9, 0.999$	Hyperparameters of Adam optimizer
ϵ	10^{-8}	Adam parameters
M	1500	Number of generations
N	128	Number of offspring per generation

working memory, training was performed on a single GPU, and it often took only a few minutes to complete. For multi-task robotic control, training was performed in parallel across four GPUs. The training duration depended on the dimensionality of the task inputs and outputs, as well as the convergence speed, and ranged approximately from 4 to 40 hours.

C Limitations

While our Plasticity-Driven Learning Framework (PDLF) demonstrates significant advances in the adaptability and functionality of Spiking Neural Networks (SNNs), there are several limitations and challenges that need to be addressed for further development and practical application:

- Generalization across diverse environments: While PDLF enhances generalization within the scope of tasks similar to those encountered during training, its effectiveness across highly diverse or drastically different environments remains less explored. The adaptability of PDLF in scenarios that deviate significantly from trained contexts needs further investigation to understand its limits in generalization.
- Complexity of Implementation: The implementation of PDLF involves intricate parametric modeling of plasticity rules which may increase the complexity of neural network architectures. This complexity could pose challenges in scaling the models for larger and more complex tasks due to increased computational demands and parameter tuning requirements.
- Optimization challenge: Using evolutionary strategies to optimize plasticity parameters, although inspired by biological processes, may be computationally expensive. However, in actual deployment, evolutionary strategy optimization is not required, so it has no significant impact on reasoning.

Addressing these limitations will require continued research efforts, not only to enhance the computational efficiency and applicability of PDLF but also to deepen our understanding of the interplay between artificial and biological learning systems. Further investigations into alternative optimization techniques, robust initialization methods, and comprehensive testing across varied environments will be key to advancing the PDLF approach in the field of artificial intelligence.