Proxy Target: Bridging the Gap Between Discrete Spiking Neural Networks and Continuous Control

Zijie Xu
Peking University
Beijing, China 100871
zjxu25@stu.pku.edu.cn

Tong Bu
Peking University
Beijing, China 100871
putong30@pku.edu.cn

Zecheng Hao
Peking University
Beijing, China 100871
haozecheng@pku.edu.cn

Jianhao Ding
Peking University
Beijing, China 100871
djh01998@alumni.pku.edu.cn

Zhaofei Yu*
Peking University
Beijing, China 100871
yuzf12@pku.edu.cn

Abstract

Spiking Neural Networks (SNNs) offer low-latency and energy-efficient decision making on neuromorphic hardware, making them attractive for Reinforcement Learning (RL) in resource-constrained edge devices. However, most RL algorithms for continuous control are designed for Artificial Neural Networks (ANNs), particularly the target network soft update mechanism, which conflicts with the discrete and non-differentiable dynamics of spiking neurons. We show that this mismatch destabilizes SNN training and degrades performance. To bridge the gap between discrete SNNs and continuous-control algorithms, we propose a novel proxy target framework. The proxy network introduces continuous and differentiable dynamics that enable smooth target updates, stabilizing the learning process. Since the proxy operates only during training, the deployed SNN remains fully energy-efficient with no additional inference overhead. Extensive experiments on continuous control benchmarks demonstrate that our framework consistently improves stability and achieves up to 32% higher performance across various spiking neuron models. Notably, to the best of our knowledge, this is the first approach that enables SNNs with simple Leaky Integrate and Fire (LIF) neurons to surpass their ANN counterparts in continuous control. This work highlights the importance of SNN-tailored RL algorithms and paves the way for neuromorphic agents that combine high performance with low power consumption. Code is available at https://github.com/xuzijie32/Proxy-Target.

1 Introduction

Reinforcement Learning (RL), combined with Artificial Neural Networks (ANNs), has become a cornerstone of modern artificial intelligence, achieving remarkable success in diverse domains such as game playing [Mnih, 2013, Silver et al., 2016, Mnih et al., 2015], autonomous driving [Kiran et al., 2021, Sallab et al., 2017, Shalev-Shwartz et al., 2016], and large language model training [Ouyang et al., 2022, Bai et al., 2022, Shao et al., 2024]. Among these, continuous control problems have drawn particular attention due to their close alignment with real-world robotic and embodied AI applications [Kober et al., 2013, Gu et al., 2017, Brunke et al., 2022]. However, the high computational cost and power demands of ANN-based RL algorithms limit their deployment on edge devices such as drones, wearables, and IoT sensors [Abadía et al., 2021, Tang et al., 2020, Yamazaki et al., 2022].

*Corresponding author

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

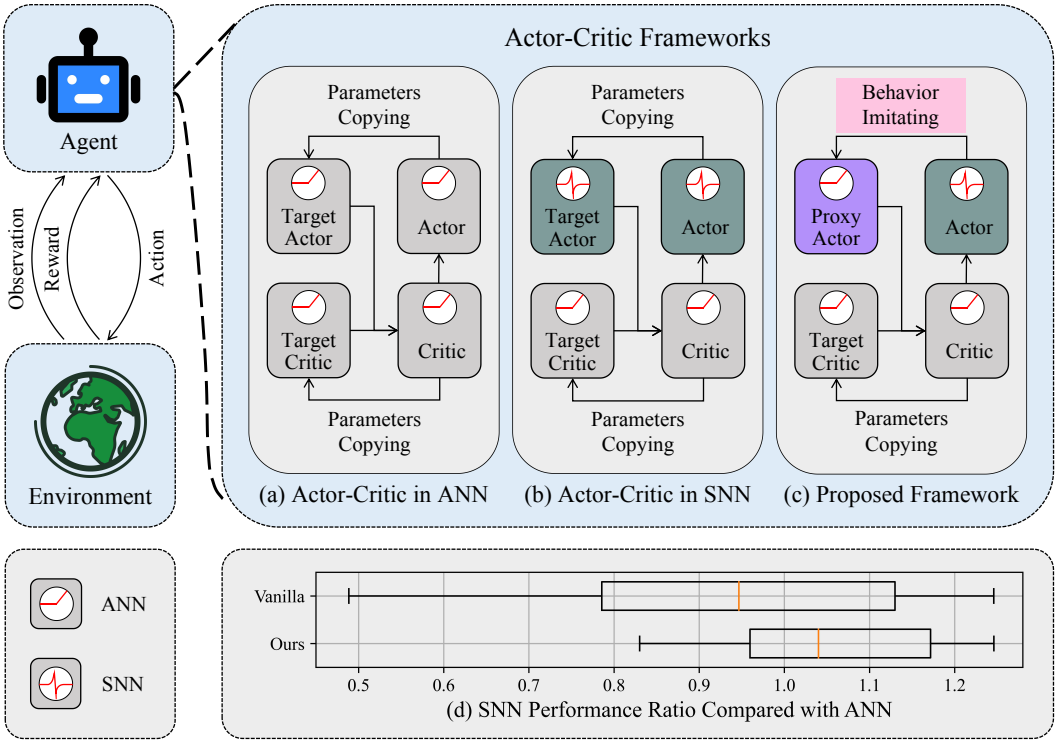


Figure 1: Overview of the training framework and performance comparison. (a)-(c) are different training paradigms. (a) Actor-Critic framework in ANNs, (b) The Actor-Critic framework for SNNs, (c) the proposed proxy target framework for SNNs. (d) Performance ratio of SNNs relative to ANNs across five random seeds and five environments. The middle orange line denotes the median, the box spans from the first to the third quartile, and the whiskers extend to the farthest data within 1.5 inter-quartile range from the box.

Inspired by biological neural systems, Spiking Neural Networks (SNNs) offer sparse, event-driven computation with ultra-low latency and energy consumption on neuromorphic hardware [DeBole et al., 2019, Davies et al., 2018]. These properties make SNNs attractive for RL applications on resource-constrained edge devices [Yamazaki et al., 2022]. Recent works have attempted to integrate SNNs into continuous-control RL algorithms via hybrid frameworks [Tang et al., 2020, 2021, Zhang et al., 2024, Chen et al., 2024, Zhang et al., 2022], where a spiking actor network (SAN) is co-trained with an ANN critic using Spatio-Temporal Backpropagation (STBP) [Wu et al., 2018, Fang et al., 2021], as illustrated in Fig. 1(b). With well-chosen hyperparameters, such frameworks have shown that SNNs can approach or even surpass the performance of ANNs in some tasks.

However, most of these studies simply retrofit SNNs into existing ANN-centric RL frameworks without adapting the algorithms to SNN dynamics. Since ANNs and SNNs exhibit fundamentally different computational characteristics, it remains unclear **whether RL algorithms designed for continuous, differentiable activations are well-suited for discrete, event-driven networks**.

A key issue arises from the *target network soft update mechanism*, a core component widely used in off-policy RL algorithms to stabilize training by gradually updating target networks [Sutton and Barto, 2018, Lillicrap, 2015, Fujimoto et al., 2018, Haarnoja et al., 2017]. This mechanism relies on continuous, smooth output changes—a property violated by the non-differentiable, binary nature of SNN spikes. This can cause abrupt output shifts, leading to unstable optimization objective, and oscillatory updates. Such instability not only makes the model highly sensitive to random seed initialization but also hampers convergence and undermines reliability in real-world deployment.

To address this mismatch between discrete spikes and continuous-control updates, we propose a proxy target framework for SNN-based RL (Fig. 1(c)). Instead of using an SNN target actor, we introduce a differentiable proxy actor network that imitates the behavior of the online spiking actor network.

The proxy target network can alter its output smoothly and continuously, stabilizing the learning process and improving performance, as demonstrated in Fig. 1(d). Since the proxy target network is only used for auxiliary training, the proposed approach retains SNN's advantages of low-latency and energy efficiency during inference in real world applications. Our main contributions are summarized as follows:

- We identify a critical mismatch between discrete SNN outputs and the continuous target network soft update mechanism used in off-policy RL, showing how this conflict destabilizes training and degrades performance.
- We propose a proxy target framework that replaces the spiking target network with a continuous, differentiable proxy, enabling smooth target updates and stable optimization.
- We introduce an implicit gradient-based update rule that aligns the proxy with the online SNN, mitigating target output gaps and giving precise optimization goals.
- Extensive experiments across multiple neuron models and continuous control benchmarks demonstrate consistent stability improvements and up to $\mathbf{32}\%$ higher average performance. To the best of our knowledge, this is the first approach where SNNs with simple Leaky Integrate-and-Fire (LIF) neurons surpass ANN performance in continuous control.

2 Related works

2.1 Learning rules of SNN-based RL

Synaptic plasticity. Inspired by the plasticity of biological synapses, several works have integrated SNNs into reinforcement learning via reward-modulated spike-timing-dependent plasticity (R-STDP) [Florian, 2007, Frémaux and Gerstner, 2016, Gerstner et al., 2018, Frémaux et al., 2013, Yang et al., 2024]. These approaches are biologically plausible and energy-efficient, but have limited performance on complex tasks.

ANN-SNN conversion. With the progress of ANN-based deep RL and ANN-SNN conversion algorithms [Cao et al., 2015, Bu et al., 2022a,b], some studies [Patel et al., 2019, Tan et al., 2021, Kumar et al., 2025] convert well-trained Deep Q-Networks (DQNs) [Mnih, 2013, Mnih et al., 2015] into SNNs. Such conversion-based methods achieve lower energy consumption during inference, but require ANN pre-training.

Gradient-based direct training. To avoid ANN pre-training, several works [Liu et al., 2022, Chen et al., 2022, Qin et al., 2022, Sun et al., 2022, 2025] directly train SNNs for RL using STBP [Wu et al., 2018], while Bellec et al. [2020] introduced e-prop with eligibility traces to learn policies through the policy gradient algorithm [Sutton et al., 1999]. These approaches achieve competitive results in discrete action spaces, but they cannot be extended to continuous control tasks.

2.2 Hybrid framework of spiking actor network.

In continuous-control problems where the action space is continuous, hybrid frameworks have been extensively explored. Tang et al. [2020] first proposed an SNN-based actor co-trained with an ANN critic in the Actor-Critic framework [Konda and Tsitsiklis, 1999]. Tang et al. [2021] demonstrated that population encoding improves the performance of spiking actor networks. Subsequent works enhanced these frameworks through various mechanisms, such as utilizing dynamic neurons [Zhang et al., 2022], incorporating lateral connections [Chen et al., 2024], adding bio-plausible topologies [Zhang et al., 2024], and integrating dynamic thresholds [Ding et al., 2022].

While these hybrid approaches report performance comparable to or exceeding their ANN counterparts, two key limitations remain. First, they often rely on complex neuron models (e.g., current-based LIF or second-order dynamic neurons), increasing computational cost and training difficulty. Second, the RL algorithms themselves are not modified to account for SNN-specific dynamics, which may cause instability and suboptimal convergence. In contrast, our proxy target framework is tailored to the discrete, event-driven nature of SNNs, achieving superior stability and performance with only simple LIF neurons.

3 Preliminaries

To avoid ambiguity, we use *training steps* to denote RL time steps and *simulation steps* to denote internal SNN simulation time steps.

3.1 Reinforcement Learning

Reinforcement Learning (RL) involves an agent interacting with an environment. The agent observes the current state s, performs an action a and receives a reward r, while the environment transitions to the next state s'. The agent's objective is to learn a policy π_ϕ, parameterized by ϕ, that maximizes the expected return.

In continuous control settings, the action space is a continuous vector (e.g., torque values). Most continuous control algorithms adopt the Actor–Critic framework with a deterministic policy [Sutton and Barto, 2018], where the actor π_ϕ outputs actions $a = \pi_\phi(s)$ and the critic Q_θ evaluates them with parameters θ [Konda and Tsitsiklis, 1999]. The actor is updated by the deterministic policy gradient [Silver et al., 2014]:

$$\nabla_\phi J(\phi) = \mathbb{E}\left[\nabla_a Q_\theta(s,a)\mid_{a=\pi(s)} \nabla_\phi \pi_\phi(s)\right]. \tag{1}$$

The critic is updated via temporal-difference (TD) learning [Sutton, 1988] using the Bellman equation [Bellman, 1966]:

$$Q_\theta(s,a) \leftarrow y, \quad y = r + \gamma Q_{\theta'}(s', a'), a' = \pi_{\phi'}(s'), \tag{2}$$

where γ is the discount factor and $(\pi_{\phi'}, Q_{\theta'})$ denote target networks.

3.2 Target network soft update

The target networks $(\pi_{\phi'}, Q_{\theta'})$ share the same architecture as their online counterparts (π_ϕ, Q_θ) but are updated more slowly to provide stable learning targets. Their parameters are updated by the Polyak function with smoothing factor τ:

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi', \quad \theta' \leftarrow \tau\theta + (1-\tau)\theta'. \tag{3}$$

These soft updates play a crucial role in off-policy continuous-control algorithms. As shown in Eqs. (1, 2), the actor and critic are jointly optimized through bootstrapping, which can cause oscillatory updates due to their mutual dependence. The target networks mitigate this by producing slowly changing targets, thereby stabilizing training and preventing divergence.

3.3 Spiking Neural Networks

Spiking neuron model. In an SNN, each neuron integrates presynaptic spikes into its membrane potential and emits a spike when the potential exceeds a threshold. The Leaky Integrate and Fire (LIF) neuron [Gerstner and Kistler, 2002] is one of the most widely used models, governed by the following dynamics:

$$I_t^l = W^l S_t^{l-1} + b^l, \quad H_t^l = \lambda V_{t-1}^l + I_t^l, \tag{4}$$

$$S_t^l = \Theta(H_t^l - V_{th}), \quad V_t^l = (1 - S_t^l)H_t^l + S_t^l \cdot V_{\text{reset}}, \tag{5}$$

where I is the input current, H is the accumulated membrane potential, S is the binary output spike, V is the membrane potential after the firing process. W and b are the weights and the biases, V_{th}, V_{reset}, and λ are the threshold voltage, the reset voltage and the membrane leakage parameter, respectively. All subscripts $(\cdot)_t$ and all superscripts $(\cdot)^l$ denote simulation step t and layer l respectively. $\Theta(\cdot)$ is the Heaviside function.

Spiking actor network. The spiking actor network (SAN) consists of a population encoder with Gaussian receptive fields [Tang et al., 2021], a multi-layer SNN, and a decoder that uses the membrane potentials of non-firing neurons as continuous outputs [Chen et al., 2024]. The SAN is trained using STBP with a surrogate gradient function. Detailed forward and backward formulations are provided in Appendix A.2.

4 Methodology

In this section, we propose a novel proxy target framework to address the incompatibility between the discrete dynamics of spiking neurons and the continuous target network soft update mechanism in RL. Section 4.1 analyzes the instability caused by discrete target outputs and introduces a proxy target network with continuous dynamics. Section 4.2 presents an implicit imitation mechanism that aligns the proxy network with the online SNN through gradient-based optimization. Section 4.3 summarizes the overall training procedure.

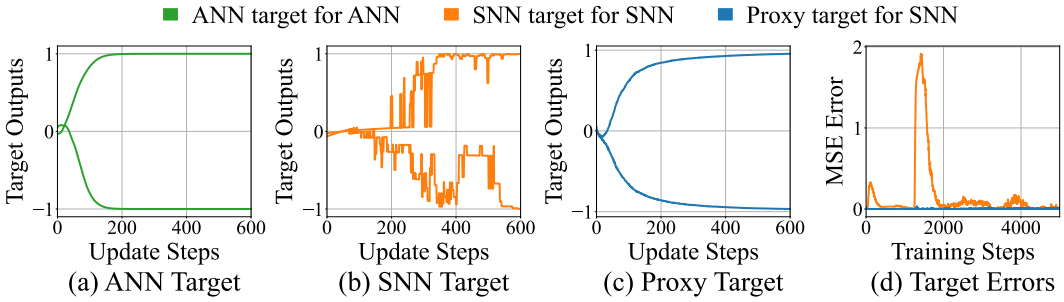


Figure 2: Effects of different target network update mechanisms. (a)-(c) show output trajectories of different target networks during updates, where each line denotes a normalized output dimension within $(-1, 1)$. (a) ANN target network exhibits smooth transitions; (b) SNN target network produces discrete and irregular output jumps; (c) the proposed proxy target achieves continuous and stable transitions. (d) Mean squared error between target and online networks during training in the InvertedDoublePendulum-v4 environment.

4.1 Addressing discrete targets by proxy network

Performance degradation due to discrete target outputs. In the standard Actor–Critic framework, the target network is updated using the Polyak function (Eq. 3), which assumes that small parameter updates lead to smooth output transitions. This assumption holds for ANNs with continuous activation functions but fails for SNNs, whose firing function is binary and non-differentiable. To illustrate this effect, we construct target networks corresponding to trained online networks using identical architectures and neuron models. The target parameters are updated according to Eq. 3 with $\tau = 0.005$ (the most commonly used setting), while the online network is frozen. Figures 2(a)–(b) show the target outputs during updates: the ANN target evolves smoothly, whereas the SNN target exhibits frequent discontinuous jumps. Although both targets eventually converge to their online counterparts, the discrete shifts in the SNN target (Fig. 2(b)) cause erratic transitions that propagate instability to the critic's optimization objectives, resulting in oscillatory and unreliable learning dynamics [Fujimoto et al., 2018].

Smoothing target outputs by proxy network. As illustrated in Fig. 1(c), to restore smoothness, we introduce a proxy target network that replaces the discrete spiking neurons of SNNs with continuous activation functions of ANNs. As shown in Fig. 2(c), the proxy network produces gradual output transitions during updates, effectively eliminating the discrete jumps observed in SNN targets. This design enables stable soft updates and prevents abrupt shifts in the target outputs, thereby improving the stability of the overall Actor–Critic learning process.

4.2 Addressing target output gaps by implicit updates

Performance degradation due to target output gaps. Although the proxy network achieves smooth updates, directly substituting spiking neurons with continuous activations (e.g., ReLU) introduces an output gap between the proxy and the online SNN. This discrepancy prevents the proxy target from accurately reproducing the output of the online SNN, distorting the critic's learning targets and reducing overall policy performance.

Aligning proxy network by implicit updates. Since the approximation errors cannot be eliminated by explicitly copying the weights of the online SNN, we propose an implicit proxy update method.

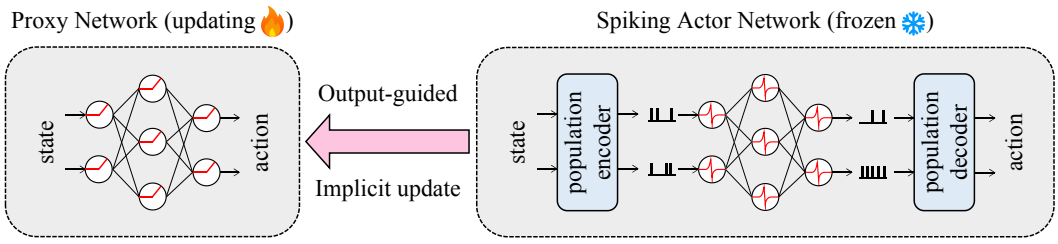


Figure 3: Architecture of the proposed proxy network and the spiking actor network. The proxy actor is updated implicitly by imitating the behavior of the online spiking actor network, ensuring stable and accurate target updates.

As shown in Fig. 3, unlike the explicit soft update that directly averages parameters, our approach computes updates in the output space, which gradually reduces the gap between the online SNN and the proxy target. Let the proxy actor $\pi^{\text{Proxy}}_{\phi'}$ have parameters ϕ' and the online spiking actor π^{SNN}_{ϕ} have parameters ϕ. For each input state s, the proxy output is implicitly updated toward the SNN actor output as:

$$\pi^{\text{Proxy}}_{\phi'}(s) \leftarrow (1-\tau') \cdot \pi^{\text{Proxy}}_{\phi'}(s) + \tau' \cdot \pi^{\text{SNN}}_{\phi}(s). \tag{6}$$

where τ' is a smoothing coefficient similar to τ in Eq. 3. Since it is difficult to directly update the corresponding parameter according to Eq. 6, we instead perform a gradient-based optimization that achieves a similar effect:

$$\phi' \leftarrow \phi' + \tau \left(\pi^{\text{SNN}}_{\phi}(s) - \pi^{\text{Proxy}}_{\phi'}(s)\right) \nabla_{\phi'} \pi^{\text{Proxy}}_{\phi'}(s) = \phi' - \frac{\tau}{2} \nabla_{\phi'} \left\| \pi^{\text{Proxy}}_{\phi'}(s)) - \pi^{\text{SNN}}_{\phi}(s)) \right\|_2^2, \tag{7}$$

where $\| \cdot \|_2^2$ denotes the squared ℓ_2 norm. Thus, the proxy network can be updated by gradient descent that minimizes the proxy loss:

$$L_{proxy} = \frac{1}{N} \sum_{i=1}^{N} \left\| \pi^{\text{Proxy}}_{\phi'}(s_i)) - \pi^{\text{SNN}}_{\phi}(s_i)) \right\|_2^2, \tag{8}$$

where N denotes the batch size, s_i are the states sampled from the replay buffer in RL algorithm. This proxy update mechanism acts as a form of implicit imitation learning, aligning the proxy network with the SNN actor while maintaining smooth output transitions, as demonstrated in Theorem 1.

Theorem 1 *Let the proxy network $\pi^{Proxy}_{\phi'}$ be updated by minimizing the loss L_{proxy} in Eq. 8. During each update, as the proxy learning rate $lr_{proxy} \to 0$, the output change satisfies*

$$\|\pi^{Proxy}_{\phi'_{new}}(s) - \pi^{Proxy}_{\phi'_{old}}(s)\| \to 0,$$

where ϕ'_{old} and ϕ'_{new} denote parameters before and after the update, respectively. Hence, minimizing L_{proxy} ensures sufficiently small and smooth policy updates, promoting stable optimization.

Since the proxy network is a multi-layer feedforward model, a universal approximator [Hornik et al., 1989], it can asymptotically match the SNN actor's output by minimizing Eq. 8. To further demonstrate this empirically, Fig. 2(d) shows the mean-squared output gap between the proxy and the SNN actor during training. While the SNN target occasionally diverges from the online SNN, the proxy network remains well-aligned throughout, validating that the proposed approach effectively mitigates target output gaps and provides precise and stable optimization goals for RL training.

4.3 Overall training framework

The proposed proxy target framework is shown in Fig. 1(c). The proxy actor network contains continuous activations of ANN that replace the discontinuous SNN target actor network. Instead of explicitly updating network parameters, the proxy actor is implicitly optimized to imitate the behavior of the online SNN actor by minimizing the loss in Eq. 8. During each update episode, the proxy actor is optimized for K iterations to reliably approximate the discrete SNN outputs, compensating for the greater representational difficulty. Meanwhile, the target critic is updated explicitly by copying

Algorithm 1 Proxy Target framework

Initialize SNN actor network $\pi_{\phi}^{\text{SNN}}(s)$, ANN critic network $Q_{\theta}^{\text{ANN}}(s,a)$ with parameters ϕ, θ
Initialize proxy actor $\pi_{\phi'}^{\text{Proxy}}(s)$ and ANN target critic $Q_{\theta'}^{\text{ANN}}(s,a)$ with parameters ϕ' and θ'
Initialize replay buffer $\mathcal{D}$
for each iteration **do**
  Execute action a according to $\pi_{\phi}^{\text{SNN}}(s)$ and store the transition (s,a,r,s') in $\mathcal{D}$
  if proxy target update **then**
    for $k=1$ to K **do**
      Sample a minibatch of N transitions (s_i,a_i,r_i,s_i') from $\mathcal{D}$
      Update proxy actor parameters ϕ' by minimizing:

$$L_{proxy} = \frac{1}{N}\sum_i \left\| \pi_{\phi'}^{\text{Proxy}}(s_i) - \pi_{\phi}^{\text{SNN}}(s_i) \right\|_2^2$$

    end for
  end if
  if ANN target update **then**
    Update ANN target critic parameters θ' by the Polyak function: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
  end if
  if ANN critic update **then**
    Compute target values y_i using proxy actor $\pi_{\phi'}^{\text{Proxy}}$ and target critic $Q_{\theta'}^{\text{ANN}}$
    Update ANN critic by minimizing: $L_{critic} = \frac{1}{N}\sum_i \left(Q_{\theta}^{\text{ANN}}(s_i,a_i) - y_i\right)^2$
  end if
  if SNN actor update **then**
    Update SNN actor by maximizing: $J = \frac{1}{N}\sum_i Q_{\theta}^{\text{ANN}}\left(s_i, \pi_{\phi}^{\text{SNN}}(s_i)\right)$
  end if
end for

weights using the Polyak function, as both the critic and target critic are conventional ANNs. The complete training procedure is summarized in Algorithm 1.

As demonstrated in Theorem 1 and Fig. 2(c)-(d), the proxy actor not only produces smooth output transitions but also closely tracks the SNN actor's behavior[1]. The proxy target framework effectively alleviates the instability caused by discrete and imprecise targets in traditional SNN–RL frameworks, resulting in a more stable training process within the Actor–Critic framework.

It is worth noting that the proposed mechanism preserves the energy efficiency of SNNs, as the proxy network and the critic network are used exclusively during training and are discarded during deployment, introducing no additional computational overhead during deployment.

5 Experiments

5.1 Experimental setup

The proposed proxy target framework (PT) was evaluated across multiple continuous-control tasks in the MuJoCo simulator [Todorov et al., 2012, Todorov, 2014b] using the OpenAI Gymnasium benchmark suite [Brockman, 2016, Towers et al., 2024], including `InvertedDoublePendulum-v4` (IDP) [Todorov, 2014a], `Ant-v4` [Schulman et al., 2015], `HalfCheetah-v4` [Wawrzyński, 2009], `Hopper-v4` [Erez et al., 2012], and `Walker2d-v4`. All environments follow the default configurations without modifications.

The experiments were carried out with different spiking neuron models, such as the LIF neuron, the current-based LIF neuron (CLIF) Tang et al. [2021], and the dynamic neuron (DN) Zhang et al.

[1]Minor fluctuations in the proxy network output (Fig. 2(c)) resemble the stochasticity introduced by soft target updates with noise injection in DRL [Fujimoto et al., 2018], which can further reduce overfitting in value estimation.

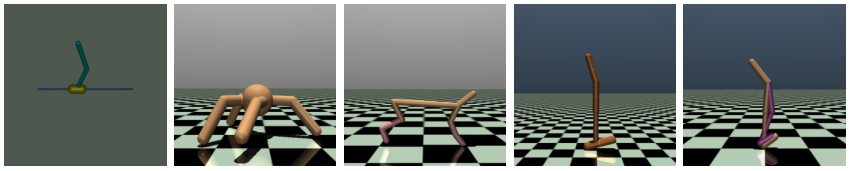


Figure 4: Continuous control tasks of the MuJoCo environments on OpenAI Gymnasium. (a) InvertedDoublePendulum-v4, (b) Ant-v4, (c) HalfCheetah-v4, (d) Hopper-v4, (e) Walker2d-v4.

[2022]. The LIF and CLIF neuron parameters follow Tang et al. [2021], while the DN parameters are initialized as in Zhang et al. [2022].

We tested the proposed algorithm in conjunction with the TD3 algorithm [Fujimoto et al., 2018], all detailed parameter settings are provided in Appendix A.4. For a fair comparison, all spiking actor networks share the same architecture, encoding, and decoding schemes provided in Appendix A.2. All SNNs have a simulation step of **5** unless otherwise noted. All reported data in this section are reproduced results across five random seeds.

5.2 Results across different spiking neurons

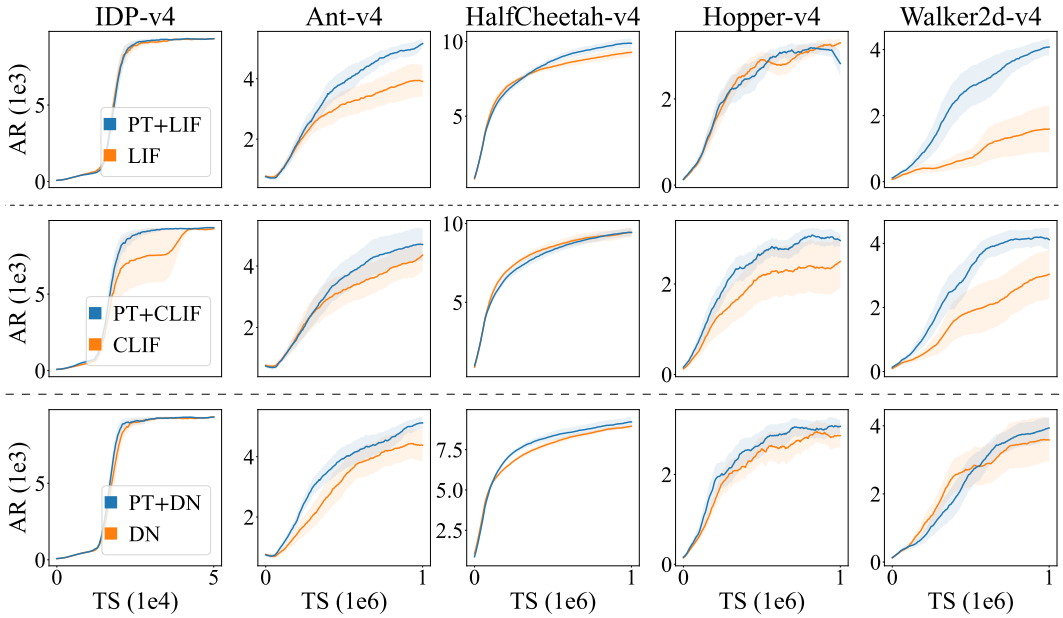
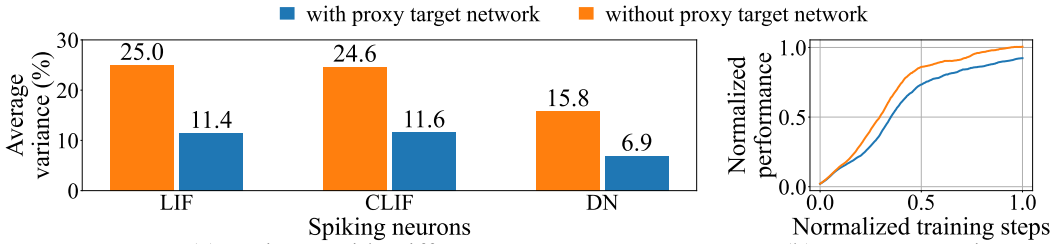


Figure 5: Learning curves of the proxy target framework (PT) and the vanilla Actor–Critic framework with the LIF neuron, the CLIF neuron and the DN. AR denotes average returns, and TS denotes training steps. The shaded region represents half a standard deviation over 5 different seeds. Curves are uniformly smoothed for visual clarity.

Increasing performance. Fig. 5 shows the learning curves of the proposed proxy target framework and the vanilla Actor-Critic framework with different spiking neurons. The proxy target framework improves the performance of different spiking neurons, demonstrating its general applicability in delivering both faster convergence and higher final returns across different neuron types and environments.

Improving stability. Fig. 6(a) shows the performance variance (after training) of the proxy target framework and the vanilla Actor-Critic framework with different spiking neurons. The proxy target framework reduces the variance of different spiking neurons, demonstrating its capability to stabilize



(a) Variance with Different Neurons (b) Proxy Target in ANN

Figure 6: (a) Average variance of different neurons after training. The average variance is computed by averaging the standard deviation ratio with 5 seeds, across all environments. (b) Normalized learning curves across all environments of the ANN integrated with the proposed proxy network across all environments. The performance and training steps are normalized linearly to $(0, 1)$. Curves are uniformly smoothed for visual clarity.

training. This is crucial for real-world deployments, where retraining costs are high and consistent behavior is required.

5.3 Exceeding state-of-the-art

To quantify relative improvements, we define the average performance gain (APG) as:

$$APG = \left(\frac{1}{|\text{envs}|} \sum_{\text{env} \in \text{envs}} \frac{\text{performance}(\text{env})}{\text{baseline}(\text{env})} - 1 \right) \cdot 100\%, \tag{9}$$

where $|\text{envs}|$ denotes the total number of environments, $\text{performance}(\text{env})$ and $\text{baseline}(\text{env})$ are the performance of the algorithm and the baseline in that particular environment. Tab.1 compares our proxy target framework with ANN-based RL, the ANN–SNN conversion method [Bu et al., 2025] (100 simulation steps), and other state-of-the-art SNN-based RL algorithms, including pop-SAN [Tang et al., 2021], MDC-SAN [Zhang et al., 2022], and ILC-SAN [Chen et al., 2024]. With the proxy network, a simple LIF-based SNN surpasses all baselines, including those using complex neuron dynamics or connection structures, and achieves higher average returns than standard ANNs. Although performance varies across tasks, the average gain across all environments and neurons indicates the general applicability of the proxy target framework.

Table 1: Max average returns over 5 random seeds with different spiking neurons, and the average performance gain against the ANN baseline, where $\pm$ denotes one standard deviation.

Method	IDP-v4	Ant-v4	HalfCheetah-v4	Hopper-v4	Walker2d-v4	APG
ANN (TD3)	7503 ± 3713	4770 ± 1014	10857 ± 475	3410 ± 164	4340 ± 383	0.00%
ANN-SNN	3859 ± 4440	3550 ± 963	8703 ± 658	3098 ± 281	4235 ± 354	-21.11%
Vanilla LIF	9347 ± 1	4294 ± 1170	9404 ± 625	3520 ± 94	1862 ± 1450	-10.54%
pop-SAN	9351 ± 1	4590 ± 1006	9594 ± 689	2772 ± 1263	3307 ± 1514	-6.66%
MDC-SAN	9350 ± 1	4800 ± 994	9147 ± 231	3446 ± 131	3964 ± 1353	0.37%
ILC-SAN	9352 ± 1	5584 ± 272	9222 ± 615	3403 ± 148	4200 ± 717	4.64%
PT-CLIF	9351 ± 1	5014 ± 1074	9663 ± 426	3526 ± 112	4564 ± 555	5.46%
PT-DN	9350 ± 1	5400 ± 277	9347 ± 666	3507 ± 144	4277 ± 650	5.06%
PT-LIF	9348 ± 1	5383 ± 250	10103 ± 607	3385 ± 157	4314 ± 423	$\mathbf{5.84}\%$

5.4 Simple neurons perform best

Interestingly, the simplest LIF neuron achieves the highest overall performance under the proxy target framework. This contrasts with previous findings where complex neuron models generally perform better. Once the SNN surpasses its ANN counterpart, the primary performance bottleneck shifts from the neuron model to the RL algorithm itself. Hence, introducing more complex spiking dynamics may unnecessarily increase training difficulty and even degrade performance.

5.5 SNN-friendly design

Fig. 6(b) shows the normalized performance of ANN with and without the proxy network. The proxy target framework cannot improve the performance in ANNs, confirming that the observed benefits arise from addressing SNN-specific challenges rather than providing a stronger RL algorithm. This validates the SNN-friendly design of our framework.

5.6 Energy efficiency

Finally, we evaluate inference energy consumption across models. The comparison includes a traditional ANN-based TD3 model, a baseline spiking actor network using vanilla LIF neurons, and our PT-LIF model. The consumption is estimated as the same way as Merolla et al. [2014], where multiply-accumulate (MAC) operation costs 3.97pJ on modern NPUs[2] [Millar et al., 2025] and synaptic operation (SOP) costs 77fJ [Hu et al., 2021].

Table 2: Energy consumptions of different tasks per inference for the spiking actor network with LIF neurons, where the energy unit is nano-joule (nJ).

Method	IDP-v4	Ant-v4	HalfCheetah-v4	Hopper-v4	Walker2d-v4	Average
ANN (TD3)	72.33	295.60	283.41	274.26	283.41	281.78 (71014.4 MACs)
Vanilla LIF	8.14	11.78	15.13	7.21	18.82	12.21 (158.6×10^3 SOPs)
PT-LIF	9.01	12.18	13.46	6.86	13.93	$\mathbf{11.09}$ (144.0×10^3 SOPs)

As shown in Tab.2, the ANN (TD3) model consumes significantly more energy, while both spiking models demonstrate dramatically lower energy consumption. Specifically, our proposed PT-LIF model achieves the lowest average consumption while maintaining better stability and performance. Moreover, PT-LIF's average firing rate (32%) is slightly lower than that of the vanilla LIF model (33%), further improving energy efficiency. These results highlight the superior energy efficiency of the proposed method, making it compelling for deployment on energy-constrained platforms.

6 Conclusion

In this work, we identified a critical mismatch between the discrete dynamics of SNNs and the continuous requirement of the target network soft update mechanism in the Actor-Critic framework. To address this, we proposed a novel proxy target framework that enables smooth target updates and faster convergence. Experimental results demonstrate that the proxy network can stabilize training and improve performance, enabling simple LIF neurons to surpass ANN performance in continuous control.

In contrast to previous works which retrofit SNNs into ANN-centric RL frameworks, this work opens a door to investigate and design SNN-friendly RL algorithms which is tailored to SNN's specific dynamics. In the future, more SNN-specific adjustments could be applied to SNN-based RL algorithms to improve performance and energy efficiency in real-world, resource-constrained RL applications.

Limitation. While this work designs a proxy target framework that is suitable for SNN-based RL, it still remains at the simulation level. The next step may involve implementing it on edge devices and enabling decisions-making in the real world.

7 Acknowledgement

This work is funded by National Natural Science Foundation of China (62422601, U24B20140, and 62088102), Beijing Municipal Science and Technology Program (Z241100004224004) and Beijing Nova Program (20230484362, 20240484703), and National Key Laboratory for Multimedia Information Processing.

[2]Energy per MAC is estimated from the maximum inference throughput (mJ^{-1}) across seven NPUs: MAX78k (C/R), GAP8, NXP-MCXN947, HX-WE2 (S/P), and MILK-V [Millar et al., 2025]. NPU initialization energy is excluded.

References

Ignacio Abadía, Francisco Naveros, Eduardo Ros, Richard R Carrillo, and Niceto R Luque. A cerebellar-based solution to the nondeterministic time delay problem in robotic control. *Science Robotics*, 6(58):eabf2756, 2021.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, 2020.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

G Brockman. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, 2022a.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022b.

Tong Bu, Maohua Li, and Zhaofei Yu. Inference-scale complexity in ANN-SNN conversion for high-performance and low-power applications. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24387–24397, 2025.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 2015.

Ding Chen, Peixi Peng, Tiejun Huang, and Yonghong Tian. Deep reinforcement learning with spiking q-learning. *arXiv preprint arXiv:2201.09754*, 2022.

Ding Chen, Peixi Peng, Tiejun Huang, and Yonghong Tian. Fully spiking actor network with intralayer connections for reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2):2881–2893, 2024.

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*, 2018.

Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, et al. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 2019.

Jianchuan Ding, Bo Dong, Felix Heide, Yufei Ding, Yunduo Zhou, Baocai Yin, and Xin Yang. Biologically inspired dynamic thresholds for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:6090–6103, 2022.

Tom Erez, Yuval Tassa, and Emanuel Todorov. Infinite-horizon model predictive control for periodic tasks with contacts. *Robotics: Science and Systems VII*, 2012.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021.

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

Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9:85, 2016.

Nicolas Frémaux, Henning Sprekeler, and Wulfram Gerstner. Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS Computational Biology*, 9(4): e1003024, 2013.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press, 2002.

Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in Neural Circuits*, 12:53, 2018.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation*, pages 3389–3396. IEEE, 2017.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):5200–5205, 2021.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12, 1999.

Aakash Kumar, Lei Zhang, Hazrat Bilal, Shifeng Wang, Ali Muhammad Shaikh, Lu Bo, Avinash Rohra, and Alisha Khalid. Dsqn: Robust path planning of mobile robot based on deep spiking q-network. *Neurocomputing*, 634:129916, 2025.

TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Guisong Liu, Wenjie Deng, Xiurui Xie, Li Huang, and Huajin Tang. Human-level control through directly trained deep spiking q-networks. *IEEE Transactions on Cybernetics*, 53(11):7187–7198, 2022.

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.

Josh Millar, Yushan Huang, Sarab Sethi, Hamed Haddadi, and Anil Madhavapeddy. Benchmarking ultra-low-power μ npus. *arXiv preprint arXiv:2503.22567*, 2025.

Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

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. *Neural Networks*, 120:108–115, 2019.

Lang Qin, Rui Yan, and Huajin Tang. A low latency adaptive coding spiking framework for deep reinforcement learning. *arXiv preprint arXiv:2211.11760*, 2022.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395. Pmlr, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Yinqian Sun, Yi Zeng, and Yang Li. Solving the spike feature information vanishing problem in spiking deep q network with potential based normalization. *Frontiers in Neuroscience*, 16:953368, 2022.

Yinqian Sun, Feifei Zhao, Zhuoya Zhao, and Yi Zeng. Multi-compartment neuron and population encoding powered spiking neural network for deep distributional reinforcement learning. *Neural Networks*, 182:106898, 2025.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

Weihao Tan, Devdhar Patel, and Robert Kozma. Strategy and benchmark for converting deep q-networks to event-driven spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9816–9824, 2021.

Guangzhi Tang, Neelesh Kumar, and Konstantinos P Michmizos. Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6090–6097. IEEE, 2020.

Guangzhi Tang, Neelesh Kumar, Raymond Yoo, and Konstantinos Michmizos. Deep reinforcement learning with population-coded spiking neural network for continuous control. In *Conference on Robot Learning*, pages 2016–2029. PMLR, 2021.

Emanuel Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *2014 IEEE International Conference on Robotics and Automation*, pages 6054–6061. IEEE, 2014a.

Emanuel Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *2014 IEEE International Conference on Robotics and Automation*, pages 6054–6061. IEEE, 2014b.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Mark Towers, Ariel Kwiatkowski, Jordan K Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, KG Arjun, et al. Gymnasium: A standard interface for reinforcement learning environments. *CoRR*, 2024.

Paweł Wawrzyński. A cat-like robot real-time learning to run. In *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers 9*, pages 380–390. Springer, 2009.

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.

Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7):863, 2022.

Zhile Yang, Shangqi Guo, Ying Fang, Zhaofei Yu, and Jian K Liu. Spiking variational policy gradient for brain inspired reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Duzhen Zhang, Tielin Zhang, Shuncheng Jia, and Bo Xu. Multi-sacle dynamic coding improved spiking actor network for reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 59–67, 2022.

Duzhen Zhang, Qingyu Wang, Tielin Zhang, and Bo Xu. Biologically-plausible topology improved spiking actor network for efficient deep reinforcement learning. *arXiv preprint arXiv:2403.20163*, 2024.

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: In the abstract and introduction sections, we clearly point out the contributions of this work.

   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: We point out the limitations in the last section.

   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: This work does not include 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: We provide detailed experimental setups in the Appendix.

   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: We provide codes with sufficient instructions in the supplementary materials.

   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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We show all setups and hyper-parameters in the Appendix.

   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: The learning curves figure and the main result table show the standard deviations.

   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: We give the experiments compute resources in the Appendix.

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: Our work conform with the NeurIPS Code of Ethics.

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: [NA]

Justification: We think there is no societal impact to be emphasized.

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: We think there is no such risks.

    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 cite the original papers properly.

    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: This work 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: This work dose 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: This work 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: All important and original component dose not involve LLM.

    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 Appendix

A.1 Proof of Theorem 1

Theorem 1 *Let the proxy network $\pi_{\phi'}^{Proxy}$ be updated by minimizing the loss L_{proxy} in Eq. 8. During each update, as the proxy learning rate $lr_{proxy} \to 0$, the output change satisfies*

$$\|\pi_{\phi'_{new}}^{Proxy}(s) - \pi_{\phi'_{old}}^{Proxy}(s)\| \to 0,$$

where ϕ'_{old} and ϕ'_{new} denote parameters before and after the update, respectively. Hence, minimizing L_{proxy} ensures sufficiently small and smooth policy updates, promoting stable optimization.

Proof 1 *By standard gradient descent, we have:*

$$\lim_{lr_{proxy}\to 0} \|(\phi'_{new} - \phi'_{old})\| = \lim_{lr_{proxy}\to 0} \left\| lr_{proxy} \cdot \nabla_{\phi'_{old}} L_{proxy} \right\| = 0.$$

Under the assumption that $\pi_{\phi'}^{Proxy}(s)$ is continuously differentiable with respect to ϕ', we apply a first-order Taylor expansion:

$$\lim_{lr_{proxy}\to 0} \left\| \pi_{\phi'_{new}}^{Proxy}(s) - \pi_{\phi'_{old}}^{Proxy}(s) \right\| = \lim_{lr_{proxy}\to 0} \left\| (\phi'_{new} - \phi'_{old}) \nabla_{\phi'} \pi_{\phi'_{old}}^{Proxy}(s) \right\| = 0.$$

A.2 Spiking actor network architecture

The spiking actor network (SAN) consists of a population encoder with Gaussian receptive fields, a multi-layer SNN with population output, and a decoder with non-firing neurons.

A.2.1 Forward propagation of the SAN

In the state encoder, each input dimension consists of N_{in} soft reset IF neurons with different Gaussian receptive fields with trainable parameters μ and σ. The neurons receive a stimulation A_E at every time step and outputs spikes S^{in} according to:

$$A_E = \exp\left[-\frac{1}{2}\frac{(s-\mu)^2}{\sigma^2}\right], \tag{10}$$

$$\begin{aligned} V_t^{in} &= V_{t-1}^{in} - S_{t-1}^{in} + A_E, \\ S_t^{in} &= \Theta(V_t^{in} - V_E), \end{aligned} \tag{11}$$

where V_E is the threshold of the encoding populations.

The last layer of the SNN consists of N_{out} neurons for each action dimension, respectively. The decoder layer is made up of non-spiking integrate neurons connected to the last layer of SNN:

$$V_t^{out} = V_{t-1}^{out} + W^{out} \cdot S_t^L + b^{out}, \tag{12}$$

where W^{out} and b^{out} are weights and biases. The final output action is determined by the membrane potential in the last time step $a = V_T^{out}$. The detailed forward propagation of the spiking actor network is shown in Algo. 2.

A.2.2 Back propagation of the SAN

SAN parameters are trained by the gradient with respect to the output action $\frac{\partial L}{\partial a}$, where $a = V_T^{out}$.

The output decoder can be updated by:

$$\begin{aligned} \frac{\partial L}{\partial W^{out}} &= \frac{\partial L}{\partial a} \cdot \frac{\partial V_T^{out}}{\partial W^{out}} \\ \frac{\partial L}{\partial b^{out}} &= \frac{\partial L}{\partial a} \cdot \frac{\partial V_T^{out}}{\partial b^{out}} \end{aligned} \tag{13}$$

Then, the main SNN is trained by STBP with the rectangular surrogate function defined as:

$$\Theta'(x) = \begin{cases} \frac{1}{2\omega}, & -\omega \le x \le \omega \\ 0, & \text{else} \end{cases}, \tag{14}$$

Algorithm 2 Forward propagation of spiking actor network

Input: M_s-dimensional observation s
Compute the stimulation of input populations:

$$A_E = \exp\left[-\frac{1}{2}\frac{(s-\mu)^2}{\sigma^2}\right]$$

for t=1,...,T **do**
  Compute the output spikes of the population encoder:

$$V_t^{in} = V_{t-1}^{in} - S_{t-1}^{in} + A_E$$

$$S_t^{in} = \Theta(V_t^{in} - V_E)$$

  for l=1,...,L **do**
    Update neurons in layer l at timestep t
  end for
  Update the decoder neurons' membrane potential:

$$V_t^{out} = V_{t-1}^{out} + W^{out} \cdot S_t^L + b^{out}$$

end for
Output: M_a-dimensional action $a = V_T^{out}$

where ω is the window size.

Next, the gradient of the encoder stimulation A_E is written in Eq.15. Note that $\frac{\partial S_t^{in}}{\partial A_E}$ is manually set to 1 to simplify the gradient computation.

$$\frac{\partial L}{\partial A_E} = \sum_{t=1}^{T} \frac{\partial L}{\partial S_t^{in}} \cdot \frac{\partial S_t^{in}}{\partial A_E} = \sum_{t=1}^{T} \frac{\partial L}{\partial S_t^{in}} \tag{15}$$

Finally, the trainable parameters μ and σ in the encoder can be updated by:

$$\begin{aligned} \frac{\partial L}{\partial \mu} &= \frac{\partial L}{\partial A_E} \cdot \frac{\partial A_E}{\partial \mu} = \frac{\partial L}{\partial A_E} \cdot \frac{s-\mu}{\sigma^2} A_E \\ \frac{\partial L}{\partial \sigma} &= \frac{\partial L}{\partial A_E} \cdot \frac{\partial A_E}{\partial \sigma} = \frac{\partial L}{\partial A_E} \cdot \frac{(s-\mu)^2}{\sigma^3} A_E \end{aligned} \tag{16}$$

A.3 Other Spiking Neuron Models

Section 3.3 already shows the LIF neuron model, this section will show two other spiking neuron models conducted in the experiments.

A.3.1 Current-Based LIF neuron model

In the current-based LIF (CLIF) neurons proposed in Tang et al. [2021], the input current in Eq.4 is redefined as:

$$I_t^l = \alpha I_{t-1}^l + W^l S_t^{l-1} + b^l, \tag{17}$$

where α is the current leakage parameter. While other dynamics of CLIF neurons are the same as those of LIF neurons.

A.3.2 Dynamic neuron model

Zhang et al. [2022] designed a second-order dynamic neurons (DN) for continuous control. The DN consists of a membrane potential V and a resistance item U to simulate hyperpolarization. Its dynamics is shown as follows:

$$\frac{dV_t^l}{dt} = {V_t^l}^2 - V_t^l - U_t^l + I_t^l \tag{18}$$

$$\frac{dU_t^l}{dt} = \theta_v V_t^l - \theta_u U_t^l \tag{19}$$

where θ_v and θ_U are the conductivities of V and U, respectively. Once firing a spike, the membrane potential V is reset to V_{reset} and the resistance U is added by θ_s.

With a firs-order Taylor expansion, the iterative DN can be written as:

$$\begin{aligned}
&C_t^l = \alpha \cdot C_{t-1}^l + W^l S_t^{l-1} + b^l; \\
&V_t^l = \left(1 - S_{t-1}^l\right) \cdot V_{t-1}^l + S_{t-1}^l \cdot V_{\text{reset}}; \\
&U_t^l = U_{t-1}^l + S_{t-1}^l \cdot \theta_u; \\
&V_{\text{delta}} = {V_t^l}^2 - V_t^l - U_t^l + C_t^l; \\
&U_{\text{delta}} = \theta_v \cdot V_t^l - \theta_u \cdot U_t^l; \\
&V_t^l = V_t^l + V_{\text{delta}}\,; \\
&U_t^l = U_t^l + U_{\text{delta}}\,; \\
&S_t^l = \Theta\left(V_t^l - V_{th}\right).
\end{aligned} \tag{20}$$

A.4 Experiment details

A.4.1 Compute Resources

We conduct the experiments on an RTX 3090 GPU and an Intel(R) Xeon(R) Platinum 8362 CPU.

A.4.2 Spiking Neuron Parameters

The LIF and CLIF neuron parameters are shown in Tab.3, which are the same as those in Tang et al. [2021], except that the LIF neuron has no current leakage parameter. The DN parameters are shown in Tab.4, determined by the pre-learning process proposed in Zhang et al. [2022].

Table 3: Parameters of LIF and CLIF [Tang et al., 2021] neurons

Parameter	LIF	CLIF [Tang et al., 2021]
Membrane leakage parameter λ	0.75	0.75
Threshold voltage V_{th}	0.5	0.5
Reset voltage V_{reset}	0	0
Current leakage parameter α	-	0.5

Table 4: Parameters of the DN [Zhang et al., 2022]

Parameter	Value
SNN time steps	5
Threshold voltage V_{th}	0.5
Current leakage parameter α	0.5
Conductivity of membrane potential θ_v	-0.172
Conductivity of hidden state θ_u	0.529
Reset voltage V_{reset}	0.021
spike effect to hidden state θ_s	0.132

A.4.3 Specific Parameters for the Proxy Target Framework

Tab.5 shows hyper-parameters of the proxy target framework for different spiking neurons. To capture the behavior of the SNN, the hidden sizes of the proxy network is set wider than that of its online SNN. Since different spiking neurons exhibit different dynamics and learning speed, hidden sizes[3] and learning rate of the proxy network vary across spiking neurons. All other hyper-parameters are kept the same.

[3]Since the InvertedDoublePendulum environment is relatively easier, there is no needs for such a wide proxy network. Thus we set the hidden size is $(512, 512)$ specifically for that environment for the CLIF neuron.

Table 5: Hyper-parameters of the proxy network framework with different spiking neurons

Parameter	LIF	CLIF [Tang et al., 2021]	DN [Zhang et al., 2022]
Proxy network architecture	$(512, 512)$	$(800, 600)$	$(512, 512)$
Proxy network activation	ReLU	ReLU	ReLU
Proxy network learning rate	$1 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	$3 \cdot 10^{-3}$
Proxy network optimizer	Adam	Adam	Adam
Proxy update iterations K	3	3	3
Proxy update batch size N	256	256	256

A.4.4 Spiking Actor Network Parameters

All hyper-parameters of the spiking actor network are shown in Tab.6. This is the same as in a wide range of previous studies [Tang et al., 2021, Zhang et al., 2022, Chen et al., 2024].

Table 6: Hyper-parameters of the spiking actor network

Parameter	Value
Encoder population per dimension N_{in}	10
Encoder threshold V_E	0.999
Network hidden units	$(256, 256)$
Decoder population per dimension N_{out}	10
Surrogate gradient window size ω	0.5

A.4.5 RL algorithm parameters

We conduct the experiment based on the TD3 algorithm [Fujimoto et al., 2018], with hyper-parameters shown in Tab.7.

Table 7: Hyper-parameters of the implemented TD3 algorithm [Fujimoto et al., 2018]

Parameter	Value
Actor learning rate	$3 \cdot 10^{-4}$
Actor regularization	None
Critic learning rate	$3 \cdot 10^{-4}$
Critic regularization	None
Critic architecture	$(256, 256)$
Critic activation	ReLU
Optimizer	Adam
Target update rate τ	$5 \cdot 10^{-3}$
Batch size N	256
Discount factor γ	0.99
Iterations per time step	1.0
Reward scaling	1.0
Gradient clipping	None
Replay buffer size	10^6
Exploration noise $\mathcal{N}(0, \sigma)$	$\mathcal{N}(0, 0.1)$
Actor update interval d	2
Target policy noise $\mathcal{N}(0, \tilde{\sigma})$	$\mathcal{N}(0, 0.2)$
Target policy noise clip c	0.5

A.4.6 Experiment environments

Fig. 4 shows various MuJoCo environments [Todorov et al., 2012, Todorov, 2014b] on OpenAI Gymnasium benchmarks [Brockman, 2016, Towers et al., 2024], including InvertedDoublePendulum (IDP) [Todorov, 2014a], Ant [Schulman et al., 2015], HalfCheetah [Wawrzyński, 2009], Hopper

[Erez et al., 2012] and Walker2d. All environment setups used the default configurations without modifications.

It is worth noting that the state vector ranges from $-\infty$ to ∞, it is normalized to $(-1, 1)$ by a tanh function. In addition, since the action has the minimum and maximum limits, the output of actor network is normalized to $(-1, 1)$ by a tanh function and then linearly scaled to $(\text{Min action}, \text{Max action})$.

A.5 Pseudo codes for the proposed proxy target framework in conjunction with TD3

We present the detailed pseudocode of the general proxy target framework in Algo.1, in Section 4.3. Specifically, Algo.3 shows how to implement the proxy target framework in the TD3 algorithm [Fujimoto et al., 2018]. It is worth noting that the original TD3 algorithm updates the target actor with delay. However, in our framework, the proxy actor is updated without delay because of its inherently slow update pace.

Algorithm 3 Proxy target framework with TD3

Initialize SNN actor network $\pi_{\phi}^{\text{SNN}}(s)$, ANN critic networks $Q_{\theta_1}^{\text{ANN}}(s,a)$, $Q_{\theta_2}^{\text{ANN}}(s,a)$ with weights ϕ, θ_1 and θ_2
Initialize proxy actor $\pi_{\phi'}^{\text{Proxy}}(s)$ and ANN target critics $Q_{\theta_1'}^{\text{ANN}}(s,a)$, $Q_{\theta_2'}^{\text{ANN}}(s,a)$ with weights ϕ', θ_1' and θ_2'
Initialize replay buffer $\mathcal{D}$
for each iteration **do**
  Execute action $a = \pi_{\phi}^{\text{SNN}}(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and next state s'
  Store the transition (s, a, r, s') in $\mathcal{D}$
  for $k = 1$ to K **do**
    Sample a minibatch of N transitions (s_i, a_i, r_i, s_i') from $\mathcal{D}$
    Update proxy actor network parameters ϕ' by minimizing the loss:

$$L_{target} = \frac{1}{N} \sum_i \left\| \pi_{\phi'}^{\text{Proxy}}(s_i) - \pi_{\phi}^{\text{SNN}}(s_i) \right\|_2^2$$

  end for
  $y_i = r_i + \gamma \min_{j=1,2} Q_{\theta_j'}^{\text{ANN}}(s_i, \tilde{a}_i)$, $\tilde{a}_i = \pi_{\phi'}^{\text{Proxy}}(s_i) + \epsilon$, $\epsilon \sim \text{clip}\left(\mathcal{N}(0, \tilde{\sigma}), -c, c\right)$
  Update ANN critics by minimizing the critic loss: $L_{critic} = \frac{1}{N} \sum_i \left(Q_{\theta_j}^{\text{ANN}}(s_i, a_i) - y_i \right)^2$
  if t mod d **then**
    Update SNN actor by maximizing the objective function: $J = \frac{1}{N} \sum_i Q_{\theta_1}^{\text{ANN}}\left(s_i, \pi_{\phi}^{\text{SNN}}(s_i)\right)$
    Update ANN critic target parameters explicitly by the Polyak function: $\theta_j' \leftarrow \tau\theta_j + (1-\tau)\theta_j'$
  end if
end for

A.6 Additional experiments results

A.6.1 Additional results in terms of performance

In the main text, we show that our proxy target framework can increase performance for various spiking neurons. Fig. 7 shows the normalized learning curves of our proxy target framework for different spiking neurons. In addition, Tab. 8, Tab. 9, and Tab .10 show the maximum average returns and the average performance gains of the proxy network against vanilla SNN with LIF neuron, CLIF neuron, and dynamic neuron, respectively.

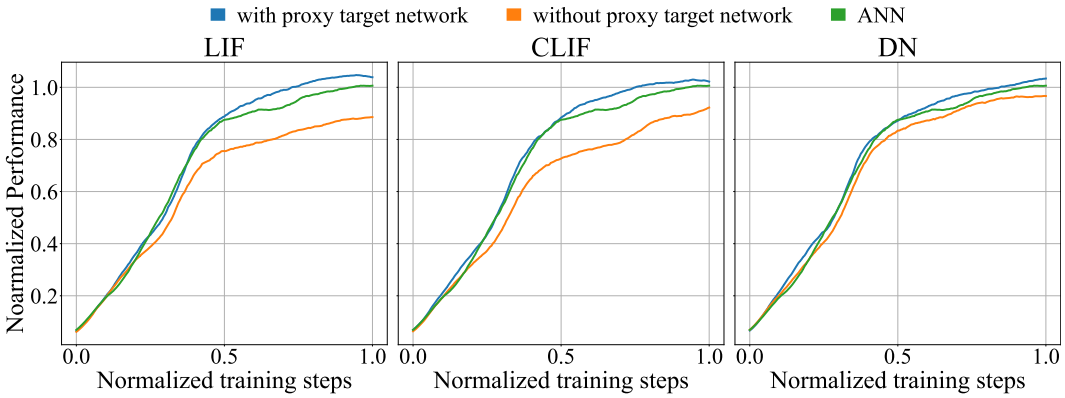


Figure 7: Normalized learning curves of the proposed proxy target framework with different spiking neurons across all environments. The performance and training steps are normalized linearly based on ANN performance. Curves are uniformly smoothed for visual clarity.

Table 8: Max average returns over 5 random seeds with LIF neurons.

Method	IDP	Ant	HalfCheetah	Hopper	Walker2d	APG
Vanilla LIF	9347 ± 1	4294 ± 1170	9404 ± 625	$\mathbf{3520 \pm 94}$	1862 ± 1450	$\mathbf{32.15\%}$
PT-LIF	$\mathbf{9348 \pm 1}$	$\mathbf{5383 \pm 250}$	$\mathbf{10103 \pm 607}$	3385 ± 157	$\mathbf{4314 \pm 423}$	

Table 9: Max average returns over 5 random seeds with CLIF neurons.

Method	IDP	Ant	HalfCheetah	Hopper	Walker2d	APG
Vanilla CLIF	$\mathbf{9351 \pm 1}$	4590 ± 1006	9594 ± 689	2772 ± 1263	3307 ± 1514	$\mathbf{15.03\%}$
PT-CLIF	$\mathbf{9351 \pm 1}$	$\mathbf{5014 \pm 1074}$	$\mathbf{9663 \pm 426}$	$\mathbf{3526 \pm 112}$	$\mathbf{4564 \pm 555}$	

Table 10: Max average returns over 5 random seeds with dynamic neurons.

Method	IDP	Ant	HalfCheetah	Hopper	Walker2d	APG
Vanilla DN	$\mathbf{9350 \pm 1}$	4800 ± 994	9147 ± 231	3446 ± 131	3964 ± 1353	$\mathbf{4.87\%}$
PT-DN	$\mathbf{9350 \pm 1}$	$\mathbf{5400 \pm 277}$	$\mathbf{9347 \pm 666}$	$\mathbf{3507 \pm 144}$	$\mathbf{4277 \pm 650}$	

A.6.2 Additional results in ANN

We show the normalized learning curves of the proxy target framework with ANN in Fig. 6(b). Here, we show the detailed learning curves and maximum average returns of 5 environments in Fig. 8 and Tab.11, respectively.

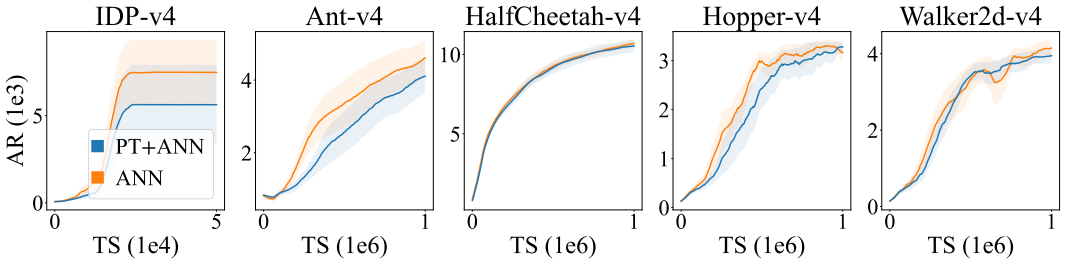


Figure 8: Learning curves of utilizing the proxy target framework in ANN. The PT represents the proxy target framework, AR denotes average returns, and TS is training steps. The shaded region represents half a standard deviation over 5 different seeds. Curves are uniformly smoothed for visual clarity.

Table 11: Max average returns over 5 random seeds with ANN (TD3).

Method	IDP	Ant	HalfCheetah	Hopper	Walker2d	APG
ANN	$\mathbf{7503 \pm 3713}$	$\mathbf{4770 \pm 1014}$	$\mathbf{10857 \pm 475}$	3410 ± 164	$\mathbf{4340 \pm 383}$	$\mathbf{-8.38\%}$
PN-ANN	5653 ± 4540	4234 ± 998	10708 ± 773	$\mathbf{3435 \pm 145}$	4106 ± 366	