Signal-Adaptive Trust Regions for Gradient-Free Optimization of Recurrent Spiking Neural Networks

Jinhao Li [*1,2] **Yuhao Sun** [*1] **Zhiyuan Ma** [2] **Hao He** [2] **Xinche Zhang** [2] **Xing Chen** [1] **Jin Li** [1] **Sen Song** [2]

Abstract

Recurrent spiking neural networks (RSNNs) are a promising substrate for energy-efficient control policies, but training them for high-dimensional, long-horizon reinforcement learning remains challenging. Population-based, gradient-free optimization circumvents backpropagation through non-differentiable spike dynamics by estimating gradients. However, with finite populations, high variance of these estimates can induce harmful and overly aggressive update steps. Inspired by trust-region methods in reinforcement learning that constrain policy updates in distribution space, we propose **Signal-Adaptive Trust Regions (SATR)**, a distributional update rule that constrains relative change by bounding KL divergence normalized by an estimated signal energy. SATR automatically expands the trust region under strong signals and contracts it when updates are noise-dominated. We instantiate SATR for Bernoulli connectivity distributions, which have shown strong empirical performance for RSNN optimization. Across a suite of high-dimensional continuous-control benchmarks, SATR improves stability under limited populations and reaches competitive returns against strong baselines including PPO-LSTM. In addition, to make SATR practical at scale, we introduce a bitset implementation for binary spiking and binary weights, substantially reducing wall-clock training time and enabling fast RSNN policy search.

1. Introduction

Reinforcement Learning (RL) has achieved strong performance on high-dimensional continuous-control problems, enabling agents to acquire complex behaviors ranging from locomotion to manipulation (Schulman et al., 2017; Lillicrap et al., 2015). However, these successes typically rely on computationally intensive artificial neural networks (ANNs), which can limit scalable deployment on power and latency constrained edge platforms (Hua et al., 2023).

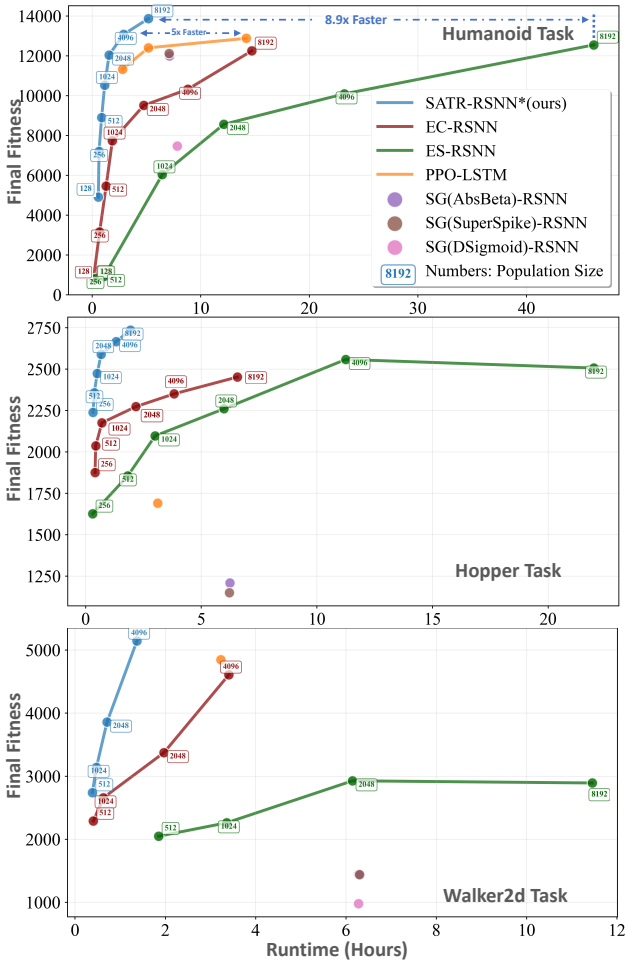

Figure 1. **Reward–runtime trade-off for RSNN policy optimization.** Final episodic return versus end-to-end wall-clock training time on *Humanoid*, *Hopper*, and *Walker2d*. For population-based methods, each marker corresponds to one population size, and lines connect settings within the same method. Our approach (SATR-RSNN*) achieves a more favorable reward–runtime trade-off, particularly in the limited-population regime.

*Equal contribution [1]Sapient Intelligence [2]Tsinghua University. Correspondence to: Sen Song <songsen@tsinghua.edu.cn>, Jin Li <jin@sapient.inc& electrixoul@outlook.com>.

Preprint. January 30, 2026.

Spiking neural networks (SNNs) provide a compelling alternative: their event-driven dynamics enable sparse computation and can deliver substantial energy efficiency, particularly on neuromorphic hardware (Roy et al., 2019; Pei et al., 2019; Davies et al., 2018; 2021; Liu et al., 2021; Modha et al., 2023; Chen et al., 2024; Shen et al., 2023; Shrestha et al., 2022). Despite this promise, recurrent spiking policies have historically struggled to match ANN performance on high-dimensional, long-horizon control (Tavanaei et al., 2019; Zanatta et al., 2024; Xu et al., 2025; Akl et al., 2023; Wu et al., 2022). Figure 1 previews our main empirical finding: a recurrent spiking RL agent that attains strong returns with substantially improved wall-clock efficiency and stability under limited population budgets.

A major obstacle for recurrent SNN control is reliable credit assignment through non-differentiable spike dynamics (Neftci et al., 2019; Guo et al., 2023). The dominant paradigm, Backpropagation Through Time (BPTT) with surrogate gradients (Werbos, 2002), and the memory cost of long unrolls often necessitates truncation, weakening the long-range temporal credit assignment required by complex control tasks (Neftci et al., 2019; Bellec et al., 2018; Xiao et al., 2022; Zhu et al., 2022; Yin et al., 2023). These challenges motivate gradient-free alternatives, such as Evolutionary Strategies (ES) and related population-based methods, which circumvent differentiability constraints by optimizing policies through population sampling (Salimans et al., 2017). Among such approaches, Evolving Connectivity (EC) has emerged as a particularly effective framework for recurrent spiking policies (Wang et al., 2023). EC models synapses as stochastic binary variables and optimizes a distribution over connectivity using Bernoulli parameters, and has demonstrated strong empirical performance on challenging high-dimensional control benchmarks, including high degree-of-freedom (DoF) tasks such as Humanoid (Wang et al., 2023).

At the same time, a key practical difficulty is shared by many population based optimizers: with finite populations, the resulting update estimates can be noisy, and training can become fragile when population budgets are limited (Salimans et al., 2017; Lehman et al., 2018; Choromanski et al., 2019). We argue that this fragility is not solely a matter of estimator variance; it is also a consequence of uncontrolled update step. Population-based methods update a sampling distribution over parameters, and under finite-sample noise, a single iteration can induce an outsized change in distribution space, leading to abrupt behavioral shifts and destabilizing learning. This issue is amplified in low-entropy regimes that are central to efficient spiking control—near-deterministic policies and especially sparse Bernoulli connectivity—where the curvature of distribution space is high. In particular, for Bernoulli variables, as probabilities approach the boundaries (0 or 1), small changes in the usual parameterization can correspond to disproportionately large changes in KL divergence (Amari, 1998; 2016; Nielsen, 2020).

A natural way to control such displacement is to bound distributional change using information-theoretic trust-region ideas (Schulman et al., 2015; Abdolmaleki et al., 2018). However, fixed trust-region budgets implicitly assume that the underlying update direction is uniformly reliable. This assumption breaks in population-based RL, where the reliability of the Monte Carlo update varies drastically with population size, task stochasticity, and training phase. Consequently, a fixed KL budget can be overly conservative when the signal is strong, yet dangerously permissive when the signal is weak or noise-dominated—precisely the vulnerable regime where limited populations are most brittle.

We introduce **Signal-Adaptive Trust Regions (SATR)**, a population based update rule that couples the allowable KL divergence between successive sampling distributions to an estimate of signal strength. Concretely, SATR bounds KL divergence normalized by an estimated signal energy, the squared norm of the population gradient estimate, expanding the trust region when the optimization signal is strong and contracting it when updates are noise-dominated. We instantiate SATR for Bernoulli connectivity distributions in EC-trained RSNNs and evaluate it on a suite of high DoF continuous control benchmarks, where it improves stability under limited population budgets and achieves competitive returns against strong baselines including PPO-LSTM. To complement algorithmic stability with practical throughput, we also introduce a bitset implementation for binary spikes and binary weights that substantially reduces wall-clock training time. Our contributions can be summarized as follows:

- **Signal-adaptive trust regions for population-based RL.** We propose SATR, a novel distributional update rule that controls the relative change between successive sampling distributions by bounding KL divergence normalized by an empirical signal energy, automatically expanding under strong signals and contracting when updates are noise-dominated.
- **High performance on high-dimensional continuous control.** Across a diverse suite of long-horizon continuous-control benchmarks, we show that SATR improves training stability when population sizes are small and reaches returns competitive with strong baselines, including PPO-LSTM, while preserving the advantages of population-based RSNN optimization.
- **Bitset acceleration for binary spikes and binary weights.** We introduce a highly optimized bitset implementation for binary spiking and binary weights that significantly reduces wall-clock training time, enabling faster RSNN policy search and improving the

reward–runtime trade-off.

2. Problem Setup and Preliminaries

2.1. RL objective and RSNN policies

We consider finite-horizon episodic reinforcement learning in an MDP with observations o_t, actions a_t, and rewards r_t. A recurrent spiking policy $\pi_\theta(a_t \mid o_{0:t})$ is parameterized by network parameters θ and is evaluated by rolling out an episode τ to obtain the (undiscounted) return

$$R(\tau) = \sum_{t=0}^{T-1} r_t \qquad (\gamma = 1). \tag{1}$$

Since our setting is inherently finite-horizon and we do not differentiate through time, using $\gamma = 1$ is natural and avoids introducing an additional discount hyperparameter; all reported results are evaluated under a unified undiscounted evaluation protocol (Appendix C.3). We write the expected return under parameters θ as

$$R(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]. \tag{2}$$

We use an LIF-based recurrent SNN policy with a recurrent spiking layer and linear read-in/read-out connections; full neuron and synapse dynamics are provided in the Appendix. Crucially, our method treats π_θ entirely as a black box: it requires only sampled returns and does not rely on backpropagation through spike dynamics.

2.2. Population-based policy search

To avoid differentiating through non-differentiable spike dynamics, we optimize a parameterized distribution through sampling. Let $p_\rho(\theta)$ be a parametric distribution with parameters ρ. The distributional objective is

$$J(\rho) = \mathbb{E}_{\theta\sim p_\rho}[R(\theta)]\,. \tag{3}$$

A score-function identity yields

$$\nabla_\rho J(\rho) = \mathbb{E}_{\theta\sim p_\rho}[R(\theta)\nabla_\rho \log p_\rho(\theta)]\,. \tag{4}$$

In practice, we estimate update directions from a finite population $\{\theta^{(n)}\}_{n=1}^N$ sampled from p_ρ. To reduce sensitivity to reward scale and improve robustness, we apply a centered-rank fitness shaping to returns (Wierstra et al., 2014; Salimans et al., 2017). Let $\mathrm{rank}(R(\theta^{(n)})) \in \{1,\dots,N\}$ denote the rank of $R(\theta^{(n)})$ within the population (ties handled by average rank), and define

$$\tilde{R}^{(n)} = \frac{\mathrm{rank}(R(\theta^{(n)})) - 1}{N-1} - \frac{1}{2}, \tag{5}$$

so that $\tilde{R}^{(n)} \in [-\frac{1}{2}, \frac{1}{2}]$ and is approximately zero-mean across the population. We then subsequently form the Monte Carlo estimate as

$$\hat{g} = \frac{1}{N}\sum_{n=1}^{N} \tilde{R}^{(n)} \nabla_\rho \log p_\rho(\theta^{(n)}), \tag{6}$$

and update ρ using $\hat{g}$. In this paper, we refer to $\hat{g}$ as the population gradient estimate.

2.3. Evolving Connectivity with Bernoulli distributions

Following Evolving Connectivity (EC) (Wang et al., 2023), we optimize a distribution over binary connectivity variables. For each synapse we introduce independent binary random variables $\theta_i \in \{0,1\}$ with Bernoulli parameters $\rho_i \in (0,1)$ and assume a factorized sampling distribution

$$p_\rho(\theta) = \prod_{i=1}^{d} \mathrm{Bern}(\theta_i;\rho_i). \tag{7}$$

For factorized Bernoulli variables,

$$\nabla_{\rho_i} \log p_\rho(\theta) = \frac{\theta_i - \rho_i}{\rho_i(1-\rho_i)}. \tag{8}$$

EC applies a natural-gradient scaling, which multiplying by the inverse Fisher Information matrix for Bernoulli variables, yielding the natural-gradient direction

$$\tilde{\nabla}_{\rho_i} J(\rho) = \mathbb{E}_{\theta\sim p_\rho}\left[\tilde{R}(\theta)\,(\theta_i - \rho_i)\right], \tag{9}$$

where $\tilde{R}(\theta)$ denotes the centered-rank transformed return.

2.4. Trust regions and KL divergence in reinforcement learning

Trust-region methods in RL explicitly control the size of policy updates to avoid destructive changes in behavior (Schulman et al., 2015; Abdolmaleki et al., 2018). A standard way to measure update size is the Kullback–Leibler (KL) divergence between the old and new policies, which is invariant to parameterization and closely related to the Fisher information metric. TRPO solves a constrained first-order improvement problem: it maximizes a linear approximation of the objective subject to a KL trust-region constraint. Under the standard second-order approximation of the KL, this yields the classical normalized natural-gradient step:

$$\Delta\rho = \alpha\, F(\rho)^{-1} g, \qquad \alpha = \sqrt{\frac{2\delta}{g^\top F(\rho)^{-1} g}}. \tag{10}$$

where g is the (Euclidean) gradient of the local surrogate objective w.r.t. ρ, $F(\rho)$ is the Fisher information matrix, and $\delta > 0$ is the (scalar) KL budget. Because the constraint introduces a single Lagrange multiplier, the resulting step length is necessarily a scalar normalization factor.

3. Methods

3.1. Overview

This section introduces *Signal-Adaptive Trust Regions (SATR)*, a distribution-space trust-region rule for the population-based updates described in Sec. 2. We assume a parameterized sampling distribution $p_\rho(\theta)$ over binary connectivity θ, and a population estimate of the update direction g obtained from episodic returns via centered-rank normalization. SATR selects the distribution update $\Delta\rho$ by regulating the KL divergence between successive sampling distributions and scaling the allowable KL change with the *signal energy* $\|g\|_2^2$.

We first formalize SATR at the level of general distributions (Sec. 3.2). We then specialize SATR to the factorized Bernoulli family used in Evolving Connectivity, obtaining a closed-form update that we call **SATR-EC** and explaining why it is stable in both low-signal and near-boundary regimes (Sec. 3.3). Finally, we describe a bitset implementation that accelerates RSNN rollouts by exploiting binary spikes and binary connectivity/weights (Sec. 3.4).

3.2. Signal-Adaptive Trust Regions (SATR)

We operate in a distributional optimization view of population-based RL (Sec. 2), where each iteration updates the parameters ρ of a sampling distribution $p_\rho(\theta)$ rather than updating a single parameter vector. A natural notion of update size is therefore the change in distribution space. Following trust-region RL methods such as TRPO/MPO, we measure distributional change using KL divergence:

$$D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho'}) \;\triangleq\; \mathbb{E}_{\theta\sim p_\rho}\left[\log \frac{p_\rho(\theta)}{p_{\rho'}(\theta)}\right]. \tag{11}$$

Bounding $D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho'})$ constrains how far the sampling distribution can move per iteration, which is particularly important when updates are estimated from finite populations.

Signal Energy. Let g denote the population update estimate for ρ (Sec. 2); in our setting g is computed from sampled connectivities and centered-rank normalized returns. Because g uses a finite population, its reliability varies with population size, task stochasticity, and training phase. We quantify signal strength by the *signal energy*

$$E \triangleq \|g\|_2^2, \tag{12}$$

Intuitively, when sampled perturbations produce a coherent update direction, g tends to have a larger norm; when the update is noise-dominated, cancellations drive $\|g\|$ down. Moreover, using centered-rank returns makes g insensitive to the raw reward scale, so $\|g\|$ primarily reflects cross-sample agreement rather than magnitude of rewards.

Definition of SATR. A single global KL budget introduces only one constraint and therefore yields a single Lagrange multiplier, implying a scalar step-length (as in TRPO, Eq. (10)). In contrast, our sampling distribution is factorized across dimensions (Eq. 7), so the KL decomposes across coordinates. This enables a stronger and more appropriate element-wise trust-region design that (i) produces an element-wise step-length vector and (ii) prevents any single coordinate from causing an outsized distributional jump.

Assume a factorized family $p_\rho(\theta) = \prod_{i=1}^d p_{\rho_i}(\theta_i)$ so that the KL decomposes as $D_{\mathrm{KL}}(p_\rho \| p_{\rho'}) = \sum_i D_i(\rho_i, \rho_i')$, where $D_i(\rho_i, \rho_i') \triangleq D_{\mathrm{KL}}(p_{\rho_i} \| p_{\rho_i'})$. We define SATR via the following element-wise signal-adaptive trust region:

$$\max_{\Delta\rho\in\mathbb{R}^d} \; g^\top \Delta\rho \quad \text{s.t.} \quad D_{KL}(\rho_i || \rho_i + \Delta\rho_i) \;\le\; \delta\, g_i^2, \;\forall i \tag{13}$$

where $\delta > 0$ is a constant. Consequently, when the population signal is strong, the feasible KL radius expands; when the signal is weak, the trust region contracts automatically, preventing noise-driven jumps in distribution space.

Local KL model and Fisher geometry. For small $\Delta\rho_i$, each one-dimensional KL admits the standard second-order expansion in terms of the scalar Fisher information $F_i(\rho_i)$:

$$D_i(\rho_i, \rho_i + \Delta\rho_i) = \frac{1}{2} F_i(\rho_i)\,(\Delta\rho_i)^2 + o\big((\Delta\rho_i)^2\big)\,. \tag{14}$$

Substituting (14) into the constraints of (13) yields a separable quadratic constraint:

$$\frac{1}{2} F_i(\rho_i)(\Delta\rho_i)^2 \;\le\; \delta\, g_i^2, \qquad \forall i. \tag{15}$$

Because the objective also separates as $\sum_i g_i \Delta\rho_i$, the optimization decouples across dimensions. For any i with $g_i \neq 0$, the optimum lies on the constraint boundary with $\mathrm{sign}(\Delta\rho_i) = \mathrm{sign}(g_i)$, giving the closed-form solution

$$\Delta\rho_i^\star = \alpha_i g_i = \sqrt{\frac{2\delta}{F_i(\rho_i)}}\, g_i, \tag{16}$$

Since the distribution is assumed to be factorized, F is diagonal. Therefore, we could reform it into the form of vectorized trust region definition. Thus, SATR naturally yields an element-wise step-length vector α rather than a simple scalar normalization factor.

3.3. SATR-EC: SATR specialized to Bernoulli connectivity

We now instantiate SATR for the factorized Bernoulli family used by Evolving Connectivity (Sec. 2). Recall that $p_\rho(\theta) = \prod_{i=1}^d \mathrm{Bern}(\theta_i; \rho_i)$ with $\rho \in (0,1)^d$, and that

Algorithm 1 SATR-EC for RSNN policy optimization

Input: initial $\rho \in (0,1)^d$, population size N, stepsize η, environment $\mathcal{E}$
repeat
  Sample $\theta^{(1)}, \dots, \theta^{(N)} \sim \prod_i \mathrm{Bern}(\theta_i; \rho_i)$
  for (in parallel) $n = 1$ **to** N **do**
    Roll out one episode with RSNN policy instantiated by $\theta^{(n)}$
    Record return $R(\theta^{(n)})$
  end for
  Compute centered-rank returns $\tilde{R}^{(1)}, \dots, \tilde{R}^{(N)}$
  $g \leftarrow \frac{1}{N} \sum_{n=1}^{N} \tilde{R}^{(n)} (\theta^{(n)} - \rho)$
  $\rho \leftarrow \rho + \eta \sqrt{\rho \odot (1-\rho)} \odot g$
until converged

EC provides a population update estimate $g \in \mathbb{R}^d$ from centered-rank normalized returns. The Bernoulli family has a simple information geometry. For a single Bernoulli mean parameter ρ_i, the Fisher information is

$$F_i(\rho_i) = \frac{1}{\rho_i(1-\rho_i)}, \tag{17}$$

and for the factorized family the Fisher matrix is diagonal, $F(\rho) = \mathrm{diag}(F_1(\rho_1), \dots, F_d(\rho_d))$. Consequently, the KL divergence admits the standard local expansion

$$D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho+\Delta\rho}) = \frac{1}{2} \Delta\rho^\top F(\rho) \Delta\rho + o(\|\Delta\rho\|^2) \tag{18}$$

$$\approx \frac{1}{2} \sum_{i=1}^{d} \frac{(\Delta\rho_i)^2}{\rho_i(1-\rho_i)}. \tag{19}$$

This expression makes the boundary curvature explicit: as $\rho_i \to 0$ or 1, the denominator $\rho_i(1-\rho_i)$ vanishes and the same Euclidean step $\Delta\rho_i$ corresponds to a much larger displacement in distribution space.

Applying the element-wise SATR constraints (13) under the local model (19) yields, for each coordinate,

$$\Delta\rho = \eta \, F(\rho)^{-1/2} g = \eta \, \sqrt{\rho \odot (1-\rho)} \, \odot \, g. \tag{20}$$

where we define the learning rate as $\eta = \sqrt{2\delta}$, yielding the practical update used in implementation. We refer to Eq. 20 as **SATR-EC** (SATR for EC). The overall procedure is summarized in Algorithm 1.

Signal-adaptive KL property. SATR-EC satisfies the SATR principle in a particularly transparent way. Substituting $\Delta\rho_i = \eta\sqrt{\rho_i(1-\rho_i)}\, g_i$ into Eq. 19 gives

$$D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho+\Delta\rho}) \;\approx\; \frac{1}{2} \sum_{i=1}^{d} \frac{\eta^2 \rho_i(1-\rho_i) g_i^2}{\rho_i(1-\rho_i)} \;=\; \frac{\eta^2}{2} \|g\|_2^2. \tag{21}$$

Thus the normalized displacement $D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho+\Delta\rho}) / \|g\|_2^2$ is approximately constant (equal to $\eta^2/2$): the realized trust region expands when $\|g\|$ is large and contracts automatically when $\|g\|$ is small.

Why SATR-EC is stable. Eq. 21 directly captures the stability mechanism targeted by SATR. (i) under noisy finite-population updates, gradient cancellations reduce $\|g\|_2$, thereby shrinking the KL step automatically; (ii) SATR-EC is boundary-aware because $\sqrt{\rho_i(1-\rho_i)} \to 0$ as $\rho_i \to 0$ or 1, preventing curvature-driven KL blow-ups near deterministic connectivity.

To highlight the contrast, applying the baseline EC step $\Delta\rho = \eta g$ to Eq. 19 yields

$$D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho+\eta g}) \;\approx\; \frac{\eta^2}{2} \sum_{i=1}^{d} \frac{g_i^2}{\rho_i(1-\rho_i)}, \tag{22}$$

which can become arbitrarily large as any ρ_i approaches a boundary. SATR-EC removes this curvature amplification by equalizing the local KL geometry through $\sqrt{\rho_i(1-\rho_i)}$.

Practical update. We update $\rho \leftarrow \rho + \Delta\rho$ using Eq. 20. To maintain $\rho \in (0,1)$ we optionally apply a tiny numerical clamp $\rho \leftarrow \mathrm{clip}(\rho, \epsilon, 1-\epsilon)$ (e.g., $\epsilon = 10^{-3}$), which is employed only for basic numerical safety and does not play the role of a stability heuristic.

3.4. Bitset acceleration for binary RSNN rollouts

To improve rollout throughput, we exploit that (i) the sampled connectivity θ in EC and (ii) RSNN spiking activity are binary. Moreover, a sampled connectivity θ remains fixed throughout an episode. Instead of representing these objects as byte or floating-point arrays and performing dense matrix multiplications, we pack binary vectors and matrices into machine words (bitsets). This reduces memory traffic and enables synaptic integration to be computed using hardware-supported bitwise operations.

Consider a presynaptic spike vector $s_t \in \{0,1\}^{d_{\mathrm{in}}}$ at timestep t and a binary connectivity mask $m_j \in \{0,1\}^{d_{\mathrm{in}}}$ for postsynaptic neuron j. Let w denote the word size (e.g., $w = 64$) and $B = \lceil d_{\mathrm{in}}/w \rceil$ the number of packed words. We pack s_t into words $\{S_{t,b}\}_{b=1}^{B}$ and m_j into words $\{M_{j,b}\}_{b=1}^{B}$. The masked spike count (the binary dot product) can then be computed as

$$m_j^\top s_t \;=\; \sum_{b=1}^{B} \mathrm{popcount}(M_{j,b} \,\&\, S_{t,b}), \tag{23}$$

where $\&$ denotes bitwise `AND` and `popcount` returns the number of set bits in a word. When synaptic weights are binary, Eq. 23 replaces the dominant multiply–accumulate

cost of synaptic integration with a small number of word-level bit operations and integer additions.

The complexity of Eq. 23 scales as $O(B)$ word operations per postsynaptic neuron, i.e., $O(d_{\text{in}}/w)$, and it benefits from highly optimized bitwise instructions on modern CPUs/GPUs. Because θ is fixed within an episode, the packed connectivity words $\{M_{j,b}\}$ are constructed once per episode, while spikes are packed on the fly at each timestep. In practice, this significantly improves cache locality and reduces memory bandwidth pressure compared to floating-point matrix multiplications, which is especially beneficial when evaluating many population members in parallel.

The bitset implementation is purely computational: it does not change the objective, estimator, or SATR-EC update rule. Its role is to significantly increase parallel rollout throughput and reduce wall-clock time per population evaluation, making limited-population training regimes and large benchmark suites computationally practical.

4. Experimental Setup

We evaluate the proposed SATR-RSNN on standard continuous control benchmarks provided by the Brax physics engine, specifically the `humanoid(17-Dof)`, `hopper(3-Dof)`, and `walker2d(6-Dof)` environments. We compare our method against 3 primary baselines:

- **PPO-LSTM:** A standard Proximal Policy Optimization agent with LSTM recurrence, representing the state-of-the-art in gradient-based RL.
- **ES-RSNN:** An RSNN trained via Evolutionary Strategies (Salimans et al., 2017), representing a gradient-free biological baseline.
- **EC-RSNN:** An RSNN trained with the original Evolving Connectivity (Wang et al., 2023), serving as an ablation baseline.
- **SG-RSNN:** An RSNN trained in PPO with the surrogate gradient method, which is specially designed for spiking neuron networks. We implemented three different surrogate gradient functions: AbsBeta(Esser et al., 2016), SuperSpike(Zenke & Ganguli, 2018), and $\text{Sigmoid}'$ (DSigmoid) (Zenke & Vogels, 2021).

All recurrent policies (RSNNs and the LSTM baseline) use a single recurrent hidden layer and are configured to have comparable numbers of trainable parameters across methods. Full detailed description of all networks and training parameters is in Appendix C.

Training budget and evaluation protocol. All methods are implemented in JAX (Bradbury et al., 2018) and evaluated using the Brax (Freeman et al., 2021) physics engine on the same hardware, ensuring a strictly consistent execution backend and fully fair wall-clock comparison (see Appendix C.3 for the unified evaluation metric and protocol). Results are reported as mean $\pm$ standard deviation over **3 random seeds** in Table 1.

For population-based methods (ES, EC, and SATR), training proceeds in generations. We run 2,000 generations on Humanoid and 1,000 generations on Hopper and Walker2d. For gradient-based baselines, the training budget is controlled by the number of gradient updates. We consider two training settings for gradient-based baselines. In the non-accelerated setting, PPO is trained for 2.2 million gradient steps and surrogate-gradient (SG) baselines are trained for 250k gradient steps on each task. This setting serves as the comparison against the non-accelerated SATR. To enable fair wall-clock comparisons under identical rollout execution with bitset acceleration, PPO is trained for 500k and 850k gradient steps to match the runtime of SATR with population sizes of 4096 and 8192, respectively. This two-setting design disentangles optimization budget from system-level acceleration. It enables fair comparisons in terms of training budget (full PPO baseline) and wall-clock runtime under acceleration (runtime-matched settings), ensuring that any additional speedup observed for SATR reflects system-level acceleration rather than reduced optimization effort.

5. Results

5.1. Performance Analysis

We evaluate the proposed Signal-Adaptive Trust Region (SATR) on three Brax continuous-control benchmarks—Humanoid, Hopper, and Walker2d—and compare against population-based baselines (EC, ES) and gradient-based methods (SG, PPO).

Overall performance. As shown in Table 1, SATR consistently and significantly outperforms ES and standard EC across all tasks. On the most challenging Humanoid task with $N = 8192$, SATR achieves a return of **13,860**. Learning curves in Figure 2 further show that SATR converges faster and more stably than standard population-based methods across all benchmarks.

Robustness to limited population budgets. To assess robustness under noisy updates, we reduce the population size from $N = 8192$ to $N = 128$. Table 1 shows a clear performance divergence as N decreases, revealing the sensitivity of different methods to sampling noise. On Humanoid, ES drops from $12{,}544$ to 922 and EC to $5{,}462$ at $N = 512$, while SATR retains a substantial $8{,}928$ ($1.6\times$ over EC). At $N = 128$, SATR still achieves $4{,}908$, whereas EC collapses to 765. Similar trends are observed on Hopper and Walker2d. These results support the signal-adaptive trust-region design, which contracts updates under weak signals

Table 1. Performance Summary on Different Tasks.

Pop	Humanoid (17-Dof)			Hopper (3-Dof)			Walker2d (6-Dof)		
	ES	EC	**SATR(Ours)**	ES	EC	**SATR(Ours)**	ES	EC	**SATR(Ours)**
8192	12544 ± 1175	12242 ± 1903	$\mathbf{13860 \pm 1107}$	2505 ± 160	2451 ± 463	$\mathbf{2735 \pm 160}$	–	–	–
4096	10087 ± 1162	10319 ± 1196	$\mathbf{13072 \pm 941}$	2557 ± 294	2350 ± 405	$\mathbf{2665 \pm 188}$	2891 ± 167	4605 ± 148	$\mathbf{5143 \pm 459}$
2048	8558 ± 659	9506 ± 939	$\mathbf{12037 \pm 278}$	2260 ± 54	2273 ± 677	$\mathbf{2588 \pm 79}$	2925 ± 295	3372 ± 721	$\mathbf{3856 \pm 607}$
1024	6031 ± 993	7729 ± 1028	$\mathbf{10520 \pm 457}$	2096 ± 66	2175 ± 755	$\mathbf{2473 \pm 237}$	2259 ± 164	2656 ± 509	$\mathbf{3139 \pm 806}$
512	922 ± 16	5462 ± 434	$\mathbf{8928 \pm 119}$	1855 ± 56	2036 ± 329	$\mathbf{2356 \pm 138}$	2047 ± 295	2288 ± 616	$\mathbf{2738 \pm 298}$
256	823 ± 24	3155 ± 2508	$\mathbf{6656 \pm 1643}$	1625 ± 203	1875 ± 51	$\mathbf{2238 \pm 97}$	–	–	–
128	750 ± 42	765 ± 36	$\mathbf{4908 \pm 32}$	–	–	–	–	–	–

and avoids noise-induced distributional collapse.

Comparison to gradient-based baselines. We further compare SATR with gradient-based baselines (Zenke & Ganguli, 2018; Esser et al., 2016; Zenke & Vogels, 2021), including advanced surrogate-gradient (SG) RSNNs and PPO-LSTM. As shown in Figure S4, SATR outperforms the best-performing SG-RSNN configuration on the Humanoid task, consistent with the difficulty of applying surrogate gradients and truncated backpropagation-through-time to long-horizon recurrent spiking policies.

Under a matched wall-clock training budget, SATR achieves comparable or higher final returns than PPO-LSTM across Humanoid, Hopper, and Walker2d, despite relying on binary spiking dynamics and gradient-free optimization. These results indicate that signal-adaptive, distributional updates provide a competitive and stable alternative to gradient-based training for recurrent spiking control.

Table 2. Runtime (seconds) comparison between Binary RSNN($RSNN\ w/o\ bitset$) and Bitset Accelerated Binary RSNN($RSNN^*$) in humanoid task.

Pop	512	1024	2048	4096	8192
$RSNN^*$	3168	4178	5645	10431	18668
$RSNN\ w/o\ bitset$	5212	6912	15569	26944	51098
× faster	1.64	1.65	2.75	2.58	2.74

5.2. Computational Efficiency and Reward–Runtime Trade-off

Compute matching and attribution. Under matched population size and generations, SATR evaluates the same number of rollouts as EC. Therefore, under the same non-accelerated rollout implementation, SATR does not yield an inherent wall-clock speedup over EC; any runtime improvement must come from system-level acceleration, rather than differences in rollout volume or evaluation frequency.

Bitset execution for binary RSNNs. To make SATR practical at scale, we introduce a high-performance bitset execution engine tailored to binary spikes and binary weights. By replacing floating-point matrix multiplications with bitwise operations and `popcount`, it increases rollout throughput by up to $\sim$2.7$\times$ (Table 2), which directly improves the reward–runtime trade-off in Fig. 1.

Beyond final return, we evaluate computational efficiency in terms of end-to-end wall-clock training time. Figure 1 summarizes the reward–runtime trade-off on Humanoid, Hopper, and Walker2d. Across all tasks, SATR achieves higher final returns at the same or lower wall-clock time compared to population-based baselines such as ES and EC, with the advantage being most pronounced on the computationally demanding Humanoid task. At matched performance levels, SATR requires significantly less runtime, enabling faster exploration of the population size–performance frontier.

These gains arise from system-level acceleration rather than changes to the learning objective or update rule. Since connectivity samples remain fixed within each episode and spike activity is binary, bitset-based execution improves efficiency without altering the optimization algorithm. In practice, on Humanoid, EC already provides an approximately 3.24 speedup over ES due to its more efficient population-based optimization. Building on this baseline, our bitset backend further reduces wall-clock training time by an additional 2.74×, lowering runtime from 51,098 s to 18,668 s. Together, these effects result in an overall speedup of up to 8.9× compared to ES. At matched control performance, SATR also achieves approximately 5× faster training than PPO-LSTM, yielding a favorable reward–runtime trade-off.

Energy efficiency estimation. We additionally provide an analytical estimate of the on-chip energy (Loihi (Davies et al., 2018)), suggesting about 400$\times$–3200$\times$ lower energy consumption than PPO-LSTM on GPU, details in Appendix B. We emphasize that these RSNN numbers are analytical estimates based on published neuromorphic per-operation energy figures (Davies et al., 2018), intended for order-of-magnitude comparison; direct measurements on neuromorphic hardware are left for future work.

5.3. Ablation: Fixed vs. Signal-Adaptive Trust Regions

To isolate the role of signal adaptivity, we compare SATR with a fixed-KL-budget trust-region variant added to the EC baseline (denoted EC+TR). EC+TR imposes a TRPO-style

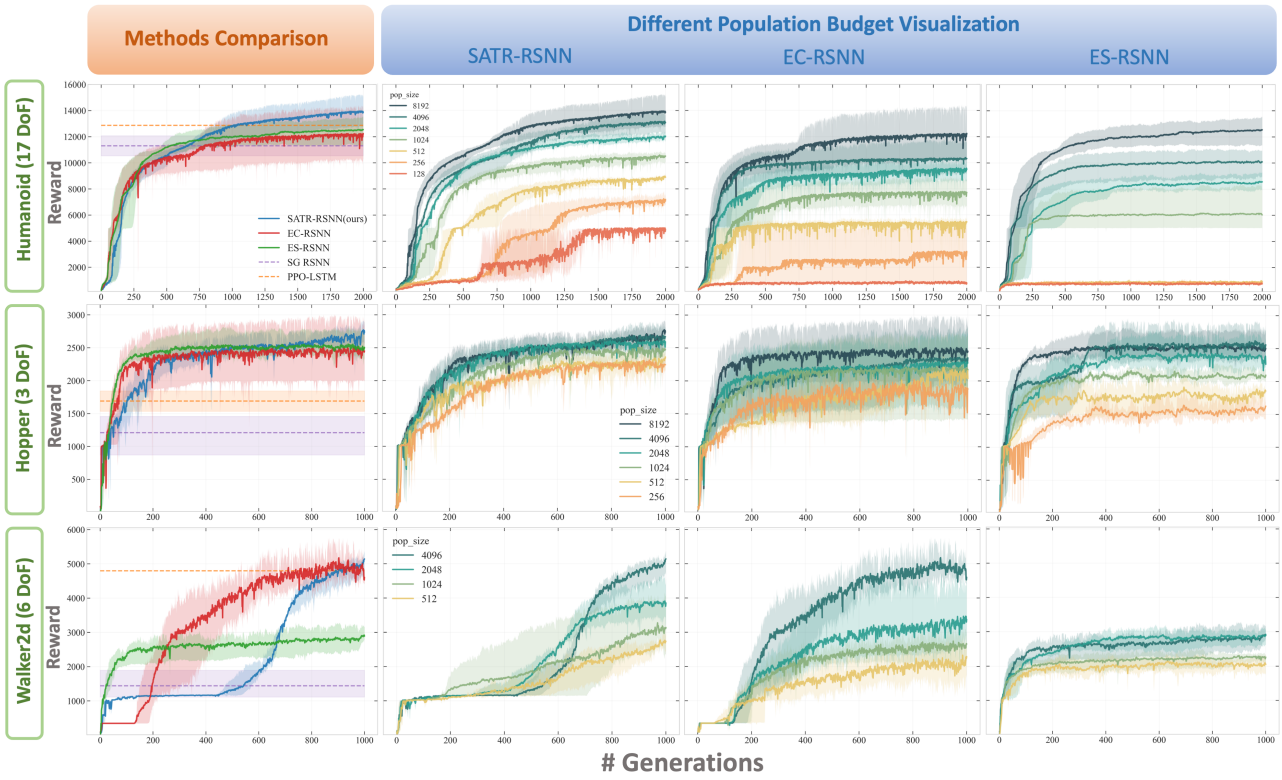

Figure 2. **Performance Benchmarks Across Tasks and Population Scales.** Learning curves for three continuous control tasks under varying population budgets (N). Each column represents a distinct population size, demonstrating the scalability of our approach. SATR (ours) consistently achieves the highest terminal returns and exhibits superior convergence stability compared to EC and ES.

KL constraint in the distribution-parameter space, which leads to a closed-form normalized natural-gradient step

$$\Delta\rho \leftarrow \sqrt{2\delta}\,\frac{g}{\sqrt{g^\top F^{-1} g}}, \tag{24}$$

where g is the population update direction in distribution space and F is the Bernoulli Fisher information matrix defining the local KL metric.

Table 3 shows that a fixed KL budget can improve EC stability under small populations, but performance is sensitive to the choice of δ and does not transfer reliably across population sizes. For example, at Pop=256, moderately sized budgets improve over EC, while overly large budgets can trigger collapse; at Pop=512 and 1024, the best-performing δ shifts substantially, and no single setting is strong across all regimes. This sensitivity makes EC+TR difficult to use in practice when the population budget changes.

In contrast, SATR-EC achieves strong performance across all tested populations without tuning a fixed KL budget (Table 3), consistently matching or outperforming the best EC+TR setting. These results support our central hypothesis: scaling the effective KL radius with the reliability of the population signal yields a more robust stability–performance trade-off than enforcing any single fixed KL budget.

Table 3. Ablation study of the difference trust region method in Humanoid task, TR means the traditional TRPO method. The **bold** number is the highest score, and the underline score is the score between the score of EC baseline and the proposed SATR.

Method	$\delta \times param$	Pop=256	Pop=512	Pop=1024
EC	-	3155 ± 2508	5462 ± 434	7729 ± 1028
EC+TR	3	-	3961 ± 1658	6832 ± 580
EC+TR	10	-	5614 ± 529	7510 ± 1242
EC+TR	30	2480 ± 1800	3738 ± 1670	8163 ± 865
EC+TR	100	4259 ± 639	3690 ± 1617	7855 ± 980
EC+TR	300	4682 ± 263	6592 ± 1846	8122 ± 343
EC+TR	1000	923 ± 121	4918 ± 1691	6812 ± 1275
SATR	-	$\mathbf{6656 \pm 1643}$	$\mathbf{8928 \pm 119}$	$\mathbf{10520 \pm 457}$

6. Conclusion

We propose Signal-Adaptive Trust Regions (SATR) for population-based learning, which scale the KL divergence between successive sampling distributions based on population update reliability measured via signal energy. Specialized to Bernoulli connectivity and RSNNs, and combined with a bitset implementation exploiting binary spikes and connectivity, SATR consistently improves stability under small population budgets and delivers an improved reward–runtime trade-off across Brax continuous-control benchmarks, enabling scalable RSNN policy search.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Akl, M., Ergene, D., Walter, F., and Knoll, A. Toward robust and scalable deep spiking reinforcement learning. *Frontiers in Neurorobotics*, 16:1075647, 2023.

Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Amari, S.-i. *Information geometry and its applications*, volume 194. Springer, 2016.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax. Version 0.3.13.

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

Choromanski, K. M., Pacchiano, A., Parker-Holder, J., Tang, Y., and Sindhwani, V. From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., Plank, P., and Risbud, S. R. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5): 911–934, 2021.

Ding, J., Zhang, J., Yu, Z., and Huang, T. Accelerating training of deep spiking neural networks with parameter initialization, 2022. URL https://openreview.net/forum?id=T8BnDXDTcFZ.

Esser, S. K., Merolla, P., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D., di Nolfo, C., Datta, P., Amir, A., Taba, B., Flickner, M., and Modha, D. S. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113:11441 – 11446, 2016.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Guo, Y., Huang, X., and Ma, Z. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023.

Hua, H., Li, Y., Wang, T., Dong, N., Li, W., and Cao, J. Edge computing with artificial intelligence: A machine learning perspective. *ACM Computing Surveys*, 55(9): 1–35, 2023.

Lehman, J., Chen, J., Clune, J., and Stanley, K. O. Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 117–124, 2018.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, D., Yu, H., and Chai, Y. Low-power computing with neuromorphic engineering. *Advanced Intelligent Systems*, 3(2):2000150, 2021.

Modha, D. S., Akopyan, F., Andreopoulos, A., Appuswamy, R., Arthur, J. V., Cassidy, A. S., Datta, P., DeBole, M. V., Esser, S. K., Otero, C. O., et al. Neural inference at the frontier of energy, space, and time. *Science*, 382(6668): 329–335, 2023.

Neftci, E. O., Mostafa, H., and Zenke, F. 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.

Nielsen, F. An elementary introduction to information geometry. *Entropy*, 22(10):1100, 2020.

Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

Roy, K., Jaiswal, A., and Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shen, G., Zhao, D., Dong, Y., Li, Y., Zhao, F., and Zeng, Y. Learning the plasticity: Plasticity-driven learning framework in spiking neural networks. *arXiv preprint arXiv:2308.12063*, 2023.

Shrestha, A., Fang, H., Mei, Z., Rider, D. P., Wu, Q., and Qiu, Q. A survey on neuromorphic computing: Models and hardware. *IEEE Circuits and Systems Magazine*, 22 (2):6–35, 2022.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

Wang, G., Sun, Y., Cheng, S., and Song, S. Evolving connectivity for recurrent spiking neural networks. *Advances in Neural Information Processing Systems*, 36:2991–3007, 2023.

Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 2002.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Wu, G., Liang, D., Luan, S., and Wang, J. Training spiking neural networks for reinforcement learning tasks with temporal coding method. *Frontiers in Neuroscience*, 16: 877701, 2022.

Xiao, M., Meng, Q., Zhang, Z., He, D., and Lin, Z. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35: 20717–20730, 2022.

Xu, Z., Bu, T., Hao, Z., Ding, J., and Yu, Z. Proxy target: Bridging the gap between discrete spiking neural networks and continuous control. *arXiv preprint arXiv:2505.24161*, 2025.

Yin, B., Corradi, F., and Bohté, S. M. Accurate online training of dynamical spiking neural networks through forward propagation through time. *Nature Machine Intelligence*, 5(5):518–527, 2023.

Zanatta, L., Barchi, F., Manoni, S., Tolu, S., Bartolini, A., and Acquaviva, A. Exploring spiking neural networks for deep reinforcement learning in robotic tasks. *Scientific Reports*, 14(1):30648, 2024.

Zenke, F. and Ganguli, S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

Zenke, F. and Vogels, T. P. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4): 899–925, 2021.

Zhu, Y., Yu, Z., Fang, W., Xie, X., Huang, T., and Masquelier, T. Training spiking neural networks with event-driven backpropagation. *Advances in Neural Information Processing Systems*, 35:30528–30541, 2022.

A. Comparison of different model architecture

Table S1 compares the model architectures used in our experiments in terms of parameter count, numerical precision, memory footprint, and dominant per-step operations. We design these architectures to support a fair comparison along two axes: (i) *algorithmic training rule* (SATR vs. EC/ES/SG/PPO) and (ii) *system implementation* (bitset-accelerated vs. conventional dense execution).

RSNN variants. All RSNN-based methods share the same recurrent spiking policy backbone (a single recurrent spiking layer with linear read-in/read-out), resulting in an identical hidden size (256) and comparable parameter count (≈193K). The difference between **RSNN (EC)** and **RSNN* (SATR)** is *not* the network structure, but the execution backend: RSNN (EC) uses a conventional dense implementation whose dominant cost is matrix multiplication, whereas RSNN* (SATR) employs our bitset engine that packs binary spikes and binary connectivity/weights into machine words and computes synaptic integration via `AND`+`popcount` (Sec. 3.4). This change preserves the underlying objective and policy class while significantly reducing memory traffic and wall-clock runtime.

Precision and memory footprint. For EC/SATR, the optimized variables are Bernoulli connectivity parameters and sampled connectivities are binary; combined with binary spikes, the core RSNN state can be represented in 1-bit precision. Consequently, RSNN and RSNN* have a compact model footprint (24KB in Table S1), making them suitable for deployment on memory-constrained platforms and aligning with neuromorphic execution. In contrast, the ES- and SG-trained RSNN baselines use FP32 weights in their standard formulations, increasing storage by $\sim 32\times$ (768KB) and relying on floating-point matrix multiplications.

LSTM baseline parameter matching. To compare against a strong gradient-based recurrent policy, we include an LSTM trained with PPO. We set the LSTM hidden size to 128 so that its total parameters (≈191K) closely match the RSNN parameter budget (≈193K), controlling for representational capacity and ensuring that performance differences are not primarily driven by model size. Despite comparable parameter counts, the LSTM requires FP32 storage (764KB) and dense matrix multiplications, which typically leads to higher runtime and energy consumption than binary RSNN execution.

Overall, Table S1 highlights that SATR-RSNN achieves competitive control performance with a model that is simultaneously compact (1-bit), compute-efficient (bitwise operations), and comparable in parameter count to standard recurrent baselines, supporting the reward–runtime advantages observed in Fig. 1.

Table S1. Comparison of Model Architectures

Model	Hidden size	Params	Precision	Size (KB)	Operation
RSNN* (SATR)	256	193K	1-bit	24	$AND + Popcount$
RSNN (EC)	256	193K	1-bit	24	$Matmul$
RSNN (ES&SG)	256	193K	FP32	768	$Matmul$
LSTM	128	191K	FP32	764	$Matmul$

B. On-Chip Energy Consumption Estimation

We estimate the *on-chip* energy consumption of SATR by counting spiking-network operations and multiplying by published energy-per-operation values reported for Loihi (Davies et al., 2018), which we use as a representative neuromorphic reference. All parameters used in the estimate are summarized in Table S2.

Table S2. Energy-per-operation and network/training parameters used for the analytical on-chip energy estimate.

Parameter	Value
Energy per synaptic spike op P_s	23.6 pJ
Within-tile spike energy P_w	1.7 pJ
Energy per neuron update P_u	81 pJ
# Generations G	2000
Population size P	$\{2^{10}, \ldots, 2^{13}\}$
# Time steps per rollout S	33,200
# Neurons N	256
Avg. spikes per neuron per step R	0.025
# Connections per neuron C	128
# Update ops per neuron per step I	4

Energy per policy evaluation (one rollout). We define E_{one} as the energy to execute one RSNN policy evaluation over S discrete timesteps. At each timestep, we account for: (i) neuron state updates and (ii) synaptic/spike-processing events. With N neurons, I update operations per neuron per timestep, and energy P_u per neuron update, the neuron-update energy is:

$$E_{\text{upd}} = P_u \, N \, I \, S.$$

We model spiking activity via an average spike rate R (spikes per neuron per timestep), yielding an expected NRS spikes per rollout. For each spike, we include the synaptic processing energy P_s plus an intra-tile spike delivery energy P_w amortized over C connections per neuron, giving:

$$E_{\text{syn}} = (P_s + CP_w) \, NRS.$$

The total energy per rollout is therefore

$$E_{\text{one}} = E_{\text{upd}} + E_{\text{syn}} = P_u NIS + (P_s + CP_w) \, NRS \approx 2.8\,\text{mJ}. \tag{25}$$

Total energy over training. SATR evaluates a population of P individuals over G generations. Assuming each individual requires one rollout evaluation, the total on-chip energy is

$$E_{\text{tot}} = E_{\text{one}}\, G\, P, \tag{26}$$

which consequently yields **5.7–45.6 kJ** for $P \in \{1024, 2048, 4096, 8192\}$ with the Humanoid settings in Table S3.

Table S3. Estimated total energy consumption for SATR-based RSNN training on Humanoid under different population sizes, compared with a PPO-LSTM baseline.

Population size P	1024	2048	4096	8192	PPO-LSTM
Estimated RSNN on-chip energy [kJ]	5.7	11.4	22.8	45.6	–
Measured GPU energy [MJ]	–	–	–	–	18.4

GPU baseline. For reference, PPO-LSTM training on the same task consumed **18.4 MJ** of measured GPU energy. The gap is consistent with the sparse, event-driven computation of RSNNs versus dense floating-point operations in conventional deep RL.

Limitations. These calculations are analytical and intended for order-of-magnitude comparison. They rely on published per-operation energy figures (Davies et al., 2018) and on the operation counts implied by Table S2. Future work will validate these estimates via direct experiments on neuromorphic hardware.

C. Performance Comparison with PPO and SG

C.1. Full PPO Results

To further evaluate the effectiveness of the EC-RSNN approach in comparison with contemporary deep reinforcement learning (RL) methods, we conducted additional training of LSTM models using Proximal Policy Optimization (PPO) (Schulman et al., 2017) on the Humanoid task, which constitutes the most challenging environment in our benchmark suit. To ensure a fair comparison under comparable computational budgets, the training runtime of PPO-LSTM was matched to different population settings of $SATR\text{-}RSNN^*$. Specifically, three PPO-LSTM variants were trained with 500k iterations (corresponding to the runtime of SATR-pop4096), 850k iterations (corresponding to the runtime of SATR-pop8192), and 2.2M iterations (corresponding to the runtime of SATR-pop8192 without bitset acceleration).

As an additional sanity check on PPO training sufficiency, we compare our final returns to the tuned PPO performance reported by a **Brax maintainer on Humanoid** (11,300 average return over 3 seeds) in `https://github.com/google/brax/discussions/230#discussioncomment-3879848`. Our 500k-iteration run already reaches 11,326.8, and longer training further improves performance to 12,396.8 (850k) and 12,876.0 (2.2M), indicating our PPO baseline is not under-trained.

The resulting learning curves are shown in Figures S1, S2 and S3. Under these runtime-matched conditions, SATR-RSNN* achieves higher returns and demonstrates more stable learning dynamics than PPO-LSTM on the Humanoid task. These results indicate that SATR-RSNN* attains performance comparable to strong gradient-based deep RL baselines, while relying on binary spiking dynamics and gradient-free optimization.

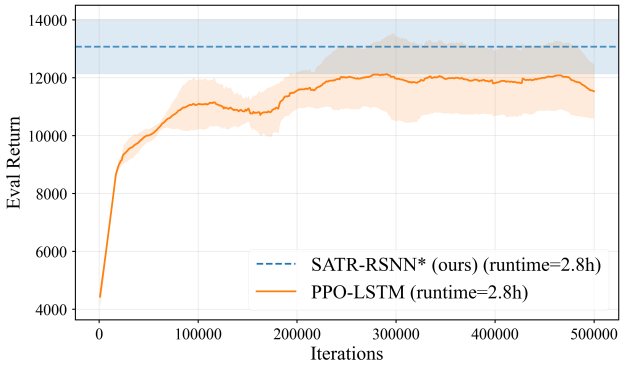

Figure S1. Performance comparison between PPO-LSTM and SATR-RSNN* (population = 4096) on the Humanoid task under matched computational budgets. Learning curves are smoothed for visualization.

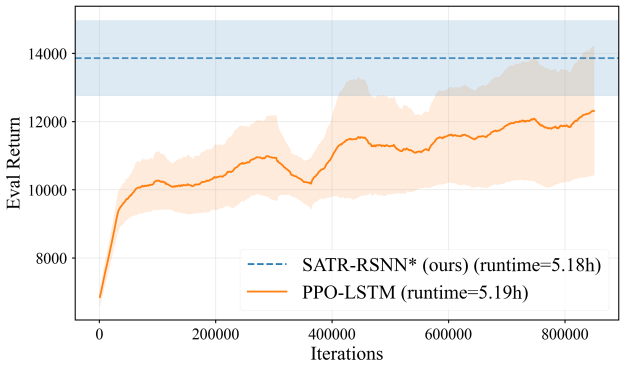

Figure S2. Performance comparison between PPO-LSTM and SATR-RSNN* (population = 8192) on the Humanoid task under matched computational budgets. Learning curves are smoothed for visualization.

C.2. Full SG Results

We present a comprehensive analysis of the Surrogate Gradient (SG) results, considering various damping factors (γ) and the parameter (β). The outcomes are illustrated in Fig-

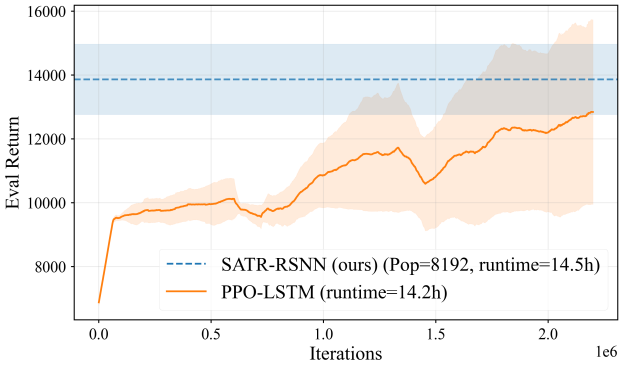

Figure S3. Performance comparison between PPO-LSTM and SATR-RSNN* **without acceleration** (population = 8192) on the Humanoid task under matched computational budgets. Learning curves are smoothed for visualization.

ure S4.

C.3. Training Objective vs. Evaluation Metric

We distinguish the *training objective* optimized by each algorithm from the *evaluation metric* reported throughout the paper. Some baselines (e.g., PPO) optimize a discounted surrogate objective (e.g., with $\gamma < 1$ and GAE), whereas our method optimizes an undiscounted objective ($\gamma = 1$). This difference is intrinsic to the algorithms and does not affect the comparability of the reported performance numbers, because **all methods are evaluated using the same metric under an identical evaluation protocol**:

- **Evaluation policy.** At evaluation time, we take the *current* policy of each method (no parameter averaging unless explicitly stated).
- **Evaluation rollouts.** We run **128 parallel environments** with this policy in inference mode (no learning updates during evaluation).
- **Metric.** For each rollout, we compute the **undiscounted episodic return** from the raw environment rewards and report the **mean return across the 128 episodes**.
- **Episode termination.** We follow the environment's standard termination conditions and horizon.

Note that γ only affects training-time credit assignment and the surrogate objective; it does not enter the evaluation metric, which is computed from the raw environment rewards. Therefore, while training objectives may differ, the reported numbers measure the same quantity: the expected episodic performance of the learned policy under a standardized evaluation setting.

D. Derivation of the EC+TR Closed-Form Step

This appendix derives the analytic update used by the fixed-budget trust-region ablation added to the EC baseline (EC+TR). Our trust region is defined over the search distribution $p_\rho(\theta)$, not over an empirical policy distribution. Under the local quadratic (Fisher) approximation of the KL divergence, the resulting trust-region subproblem admits a closed-form solution, eliminating the need for TRPO-style line search/bisection.

D.1. KL trust region in distribution space

EC+TR constrains the KL change between the current and updated search distributions:

$$D_{\mathrm{KL}}\big(p_\rho \,\|\, p_{\rho'}\big) \le 2\delta, \qquad \rho' = \rho + \Delta\rho. \tag{27}$$

Using the standard second-order expansion of KL around ρ,

$$D_{\mathrm{KL}}\big(p_\rho \,\|\, p_{\rho+\Delta\rho}\big) = \frac{1}{2}\,\Delta\rho^\top F(\rho)\,\Delta\rho \;+\; o(\|\Delta\rho\|^2), \tag{28}$$

where $F(\rho)$ is the Fisher information matrix of the distribution family $\{p_\rho\}$,

$$F(\rho) = \mathbb{E}_{\theta\sim p_\rho}\big[\nabla_\rho \log p_\rho(\theta)\,\nabla_\rho \log p_\rho(\theta)^\top\big], \tag{29}$$

the constraint (27) reduces (up to second order) to the quadratic trust region

$$\frac{1}{2}\,\Delta\rho^\top F(\rho)\,\Delta\rho \le \delta. \tag{30}$$

D.2. Trust-region subproblem with a linearized objective

Let $\mathcal{L}(\rho)$ denote the objective optimized by EC in distribution-parameter space (e.g., an expected fitness or a surrogate improvement). We linearize $\mathcal{L}$ locally:

$$\mathcal{L}(\rho + \Delta\rho) \approx \mathcal{L}(\rho) + \tilde{g}^\top \Delta\rho, \tag{31}$$

where $\tilde{g}$ is the local ascent direction in the Euclidean parameterization. Then EC+TR solves

$$\max_{\Delta\rho} \quad \tilde{g}^\top \Delta\rho \quad \text{s.t.} \quad \frac{1}{2}\,\Delta\rho^\top F(\rho)\,\Delta\rho \le \delta. \tag{32}$$

This is a standard trust-region problem in the local KL geometry induced by $F(\rho)$.

D.3. Closed-form normalized natural step

Form the Lagrangian

$$\mathcal{J}(\Delta\rho, \lambda) = \tilde{g}^\top \Delta\rho - \lambda\left(\frac{1}{2}\Delta\rho^\top F \Delta\rho - \delta\right), \qquad \lambda \ge 0. \tag{33}$$

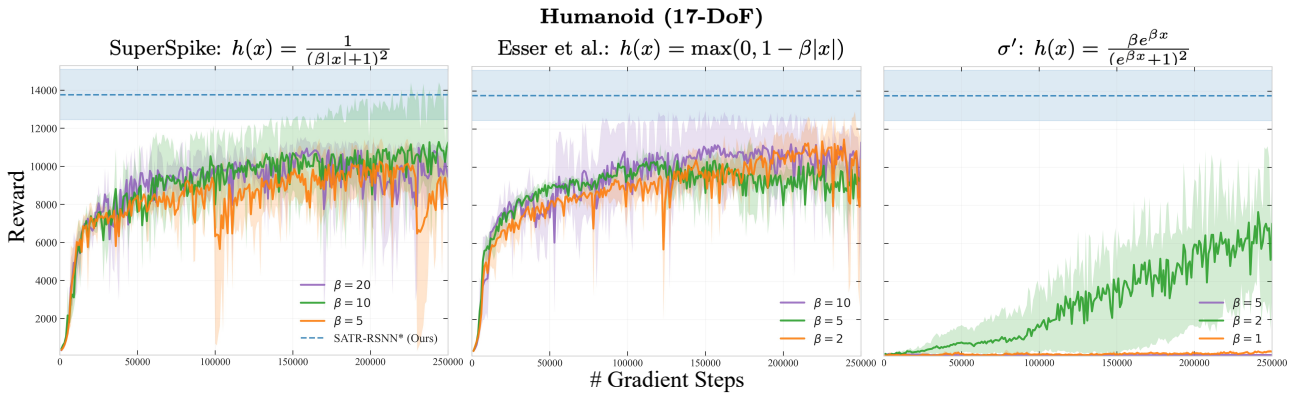

Figure S4. Performance of different surrogate gradient methods with different training parameters in the Humanoid task, which indicates our method beats the best parameter combination of surrogate gradient methods.

Stationarity w.r.t. $\Delta\rho$ yields

$$\nabla_{\Delta\rho}\mathcal{J} = \tilde{g} - \lambda F \Delta\rho = 0 \quad \Rightarrow \quad \Delta\rho = \frac{1}{\lambda} F^{-1}\tilde{g}. \tag{34}$$

At optimum the constraint is active (unless $\tilde{g} = 0$), hence

$$\frac{1}{2}\Delta\rho^\top F \Delta\rho = \delta. \tag{35}$$

Substituting (34) into (35) gives

$$\frac{1}{2}\left(\frac{1}{\lambda}F^{-1}\tilde{g}\right)^\top F \left(\frac{1}{\lambda}F^{-1}\tilde{g}\right) = \frac{1}{2\lambda^2}\,\tilde{g}^\top F^{-1}\tilde{g} = \delta, \tag{36}$$

so that

$$\lambda = \sqrt{\frac{\tilde{g}^\top F^{-1}\tilde{g}}{2\delta}}. \tag{37}$$

Plugging back yields the closed-form normalized natural-gradient step:

$$\Delta\rho = \sqrt{2\delta}\,\frac{F^{-1}\tilde{g}}{\sqrt{\tilde{g}^\top F^{-1}\tilde{g}}}. \tag{38}$$

Consistency with the main-text notation. In our implementation and main text, we denote by

$$g \triangleq F^{-1}\tilde{g} \tag{39}$$

the Fisher-preconditioned (natural) ascent direction in distribution space. Under this convention, (38) becomes

$$\Delta\rho = \sqrt{2\delta}\,\frac{g}{\sqrt{g^\top F^{-1} g}}, \tag{40}$$

which is exactly the update used by EC+TR in the main text. Moreover, (40) satisfies the quadratic trust-region constraint $\frac{1}{2}\Delta\rho^\top F\Delta\rho = \delta$ (up to the local KL approximation).

D.4. Bernoulli Fisher for factorized distributions

For the factorized Bernoulli family $p_\rho(\theta) = \prod_{i=1}^d \text{Bernoulli}(\theta_i;\rho_i)$ with $\rho_i \in (0,1)$,

$$\log p_\rho(\theta) = \sum_{i=1}^{d} \left[\theta_i \log\rho_i + (1-\theta_i)\log(1-\rho_i)\right], \tag{41}$$

and

$$\frac{\partial}{\partial\rho_i}\log p_\rho(\theta) = \frac{\theta_i - \rho_i}{\rho_i(1-\rho_i)}. \tag{42}$$

Therefore the Fisher information is diagonal:

$$F_{ij}(\rho) = \begin{cases} \frac{1}{\rho_i(1-\rho_i)}, & i = j \\ 0, & i \neq j \end{cases} \tag{43}$$

Substituting (43) into (40) yields an element-wise scaled step for EC+TR.

D.5. Why EC+TR does not require line search

Canonical TRPO often relies on backtracking/line search because the KL constraint is enforced on an empirical policy distribution (over visited states) and may not be satisfied by approximate updates under sampling noise and function approximation. In contrast, EC+TR defines the KL directly over $p_\rho(\theta)$ and uses the local quadratic KL approximation (28), which turns the constraint into the quadratic form (30). The resulting trust-region subproblem (32) has the analytic solution (40), so the step size is obtained in closed form and no bisection/line search is needed.

E. Hardware

All experiments were conducted on a single GPU server with the following specifications. Additionally, all efficiency

experiments, including those that measure wall-clock time, were executed on a single GPU within this server.

- 1× NVIDIA GeForce RTX 4090
- 2× Intel Xeon Silver 4110 CPUs
- 252GB system Memory

F. Hyperparameters

In this section, we provide the hyperparameters for our framework, Evolving Connectivity (EC), and the baselines, including Evolution Strategies (ES) and Surrogate Gradient (SG). All methods utilize the same hyperparameter set for all three locomotion tasks. We also list the settings of neural networks below.

Table S4. Hyperparameters for SATR (ours) and Evolving Connectivity (EC)

Hyperparameter	Value
Learning rate η	0.15
Exploration probability ϵ	10^{-3}

Table S5. Hyperparameters for Evolution Strategies

Hyperparameter	Value
Learning rate η	0.15
Noise standard deviation σ	0.3
Weight decay	0.1

Table S6. Hyperparameters for Surrogate Gradient

Hyperparameter	Value
Proximal Policy Optimization (PPO)	
Batch size	2048
BPTT length	16
Learning rate η	3×10^{-4}
Clip gradient norm	0.5
Discount γ for **Training**	0.99
Discount γ for **Evaluation**	1
GAE λ	0.95
PPO clip	0.2
Value loss coefficient	1.0
Entropy coefficient	10^{-3}
Surrogate Gradient	
Surrogate function	AbsBeta
Surrogate function parameter β	10

Table S7. Settings of neural networks

Hyperparameter	Value
Recurrent Spiking Neural Network (RSNN)	
Number of neurons d_h	256
Excitatory ratio	50%
Simulation time per environment step	16.6 ms
Simulation timestep Δt	0.5 ms
Synaptic time constant τ_{syn}	5.0 ms
Membrane time constant τ_m	10.0 ms
Output time constant τ_{out}	10.0 ms
Input membrane resistance [1] R_{in}	$0.15 \cdot \tau_m \sqrt{\frac{2}{d_{in}}}$
Hidden membrane resistance [1] R_h	$1.0 \cdot \frac{\tau_m}{\tau_{syn}} \sqrt{\frac{2}{d_h}}$
Output membrane resistance [1] R_{out}	$5.0 \cdot \tau_{out} \sqrt{\frac{2}{d_h}}$
Long-short term memory (LSTM)	
Hidden size	128

G. Discussion

SATR targets a practical failure mode of population-based RSNN training: with finite populations, noisy updates can cause overly large *distributional* jumps, especially for sparse Bernoulli connectivity where KL curvature explodes near $\rho \to 0/1$. By tying the allowable KL displacement to the signal energy $\|g\|_2^2$, SATR expands when the update direction is coherent and contracts when it is noise-dominated, which explains its stability advantage in small-population regimes (Table 1) and its reduced sensitivity relative to fixed-budget trust regions (Table 3). Specializing to Bernoulli yields the boundary-aware scaling $\sqrt{\rho(1-\rho)}$ (Eq. (20)), preventing curvature-induced KL blow-ups that can destabilize EC.

SATR improves optimization robustness, while our bitset backend improves throughput without changing the learning rule. Because connectivity and spikes are binary, `AND`+`popcount` replaces dense matmuls, yielding substantial wall-clock gains and a better reward–runtime trade-off (Fig. 1, Table 2).

H. Limitation and future work.

Limitations include reliance on the local (Fisher) KL approximation, the factorized distribution assumption, and energy results being analytical rather than measured on neuromorphic hardware. Future work includes extending SATR to structured connectivity distributions, incorporating uncertainty estimates of g to better calibrate trust-region size, and validating energy/latency on neuromorphic deployments.

[1] Resistance is set following (Ding et al., 2022) to preserve variance.