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

Jinhao Li[1,2,4,*] **Yuhao Sun**[2,*] **Zhiyuan Ma**[1,3] **Hao He**[2,5]
Xinche Zhang[1,3] **Xing Chen**[2] **Jin Li**[2,†] **Sen Song**[1,3†]
[1]Tsinghua Laboratory of Brain and Intelligence, Tsinghua University
[2]Sapient Intelligence
[3]School of Biomedical Engineering, Tsinghua Medicine, Tsinghua University
[4]School of Basic Medical Sciences, Tsinghua Medicine, Tsinghua University
[5]Department of Psychological and Cognitive Sciences, Tsinghua University

Abstract

Recurrent spiking neural networks (RSNNs) offer a promising and efficient substrate for locomotion control, yet optimizing them for high-dimensional, long-horizon reinforcement learning remains challenging. Population-based, gradient-free methods avoid backpropagation through non-differentiable spike dynamics by optimizing a sampling distribution over network parameters, but finite-population estimates can be highly variable and may induce unstable distributional shifts. Inspired by trust-region methods in reinforcement learning, we propose Signal-Adaptive Trust Regions (SATR), a distributional update rule that couples coordinate-wise local Kullback–Leibler budgets to the squared components of the population update. The resulting predicted distributional step scales with the realized update energy, which reflects both the optimization signal and finite-population fluctuations. We instantiate SATR for factorized Bernoulli connectivity distributions, yielding a simple and effective boundary-aware update for recurrent spiking policies. To improve rollout throughput, we further develop a bitset backend for binary spikes and binary weights, replacing dense floating-point operations with word-level bitwise operations. Experiments on high-dimensional continuous-control benchmarks show that SATR improves stability and final returns over population-based RSNN baselines across population budgets. With bitset acceleration, SATR-RSNN achieves a favorable reward–runtime trade-off and competitive performance against PPO-LSTM under matched wall-clock training budgets. Code is available at https://github.com/SWF-hao/SATR-NeurIPS26-Official.

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]. These successes, however, 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]. This motivates policy representations and training algorithms that can retain strong control performance while improving computational efficiency. 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,

*Equal contribution.
†Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

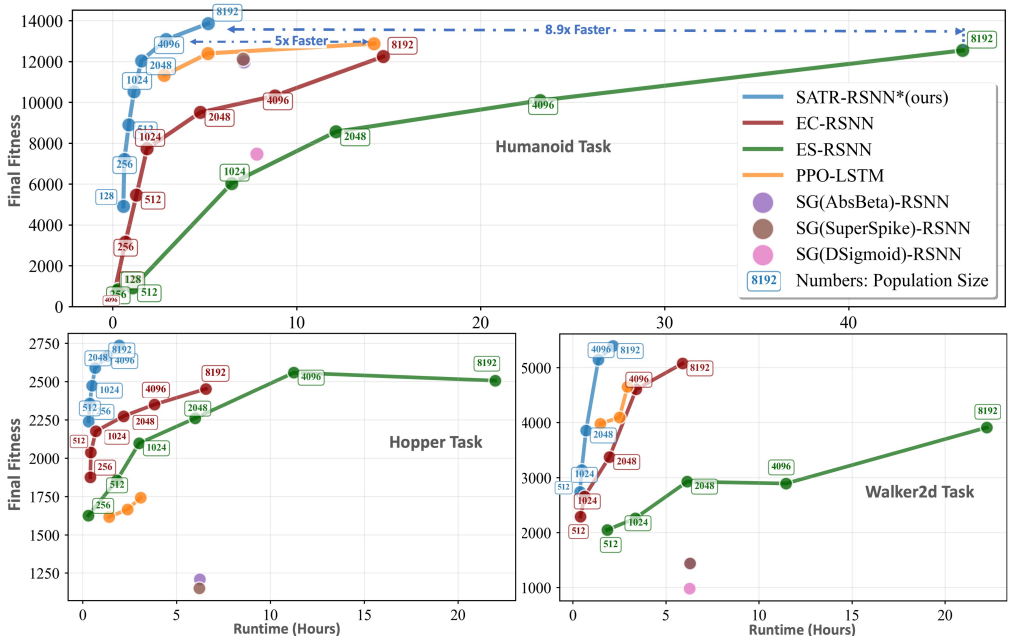

Figure 1: **Reward–runtime trade-off for RSNN policy optimization.** Final episodic return versus end-to-end wall-clock training time under the evaluated optimizer/backend configurations. SATR changes the optimizer and primarily affects return, while the bitset backend changes execution and primarily affects rollout time. Their combination yields a favorable trade-off in our evaluated Brax settings, especially at smaller population budgets.

Liu et al., 2021, Modha et al., 2023, Chen et al., 2024, Shen et al., 2023, Shrestha et al., 2022, Xu et al., 2026a]. 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 summarizes our main empirical observation: in the evaluated Brax settings, combining SATR with a bitset RSNN backend improves the reward–runtime trade-off over population-based RSNN baselines, particularly under limited population budgets.

A major obstacle for RSNN control is reliable credit assignment through non-differentiable spike dynamics [Neftci et al., 2019, Guo et al., 2023, Xu et al., 2026b]. The dominant paradigm is Backpropagation Through Time (BPTT) [Werbos, 1990] combined with surrogate gradients [Neftci et al., 2019]. However, the memory cost of long unrolls often necessitates truncation, which weakens 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].

However, population-based optimization updates can be noisy, making population-based RSNN training fragile under limited budgets [Salimans et al., 2017, Lehman et al., 2018, Choromanski et al., 2019]. The issue is not only estimator variance: a noisy update can also cause an uncontrolled jump in the sampling distribution, producing abrupt behavioral changes. This is especially severe for sparse Bernoulli connectivity, where probabilities near 0 or 1 induce high KL curvature, so small parameter changes can yield large distributional shifts [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 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 propose **Signal-Adaptive Trust Regions (SATR)**, a distribution-space update rule for gradient-free RSNN optimization. SATR couples each coordinate's local KL budget to the squared component of the Fisher-preconditioned population update. The total predicted KL displacement is therefore proportional to the squared update norm: it contracts when population averaging yields a small update and expands when the realized update is larger. We instantiate SATR for factorized Bernoulli connectivity distributions used in EC, yielding a simple boundary-aware update for recurrent spiking policies. To make population-based RSNN policy search practical at scale, we further introduce a bitset execution backend for binary spikes and binary weights. This backend replaces dense floating-point operations with word-level bitwise operations, improving rollout throughput without changing the policy class, sampling distribution, fitness estimator, or SATR update. In summary, SATR contributes algorithmic gain in stability and return, while bitset execution contributes systems gain in wall-clock efficiency.

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. The agent acts through a recurrent spiking policy $\pi_\theta(a_t \mid o_{0:t})$ parameterized by binary network parameters $\theta \in \{0,1\}^d$. The distributional optimization objective aims to maximize the expected return $J(\rho) = \mathbb{E}_{\theta\sim p_\rho}[R(\theta)]$ under a parametric sampling distribution $p_\rho(\theta)$ with parameters ρ. Crucially, our framework treats π_θ as a black box and requires only sampled episodic returns, bypassing backpropagation through non-differentiable LIF spike dynamics (neuron settings in Appendix H).

2.2 Population-Based Policy Search and Evolving Connectivity

To optimize the distributional objective without backpropagating through the policy, population-based methods sample $\{\theta^{(n)}\}_{n=1}^N$ from p_ρ and evaluate their episodic returns. We apply centered-rank fitness shaping [Salimans et al., 2017], mapping returns to $\tilde{R}^{(n)} \in [-\frac{1}{2}, \frac{1}{2}]$ with $\sum_{n=1}^N \tilde{R}^{(n)} = 0$. The Euclidean score-function estimate for these shaped fitness values is

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

Rank shaping makes the estimate invariant to order-preserving changes in raw rewards; it is not, in general, an unbiased estimate of the gradient of the unshaped objective $J(\rho)$. Following Evolving Connectivity (EC) [Wang et al., 2023], we use a factorized Bernoulli distribution $p_\rho(\theta) = \prod_{i=1}^d \mathrm{Bern}(\theta_i; \rho_i)$, with $\rho_i \in (0,1)$. Its coordinate-wise score is $(\theta_i - \rho_i)/[\rho_i(1-\rho_i)]$, and its Fisher information matrix is diagonal with $F_i(\rho_i) = 1/[\rho_i(1-\rho_i)]$. The Fisher-preconditioned population direction is therefore

$$u \triangleq F(\rho)^{-1}\hat{g}^{\mathrm{score}} = \frac{1}{N}\sum_{n=1}^{N} \tilde{R}^{(n)}(\theta^{(n)} - \rho). \tag{2}$$

We consistently distinguish the Euclidean score estimate $\hat{g}^{\mathrm{score}}$ from the natural-gradient direction u. The baseline EC update is $\Delta\rho_{\mathrm{EC}} = \eta u$, where $\eta > 0$ is the learning rate.

2.3 Trust Regions in Reinforcement Learning

Information-theoretic trust regions control distributional updates through a KL budget $D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho+\Delta\rho}) \le \delta$. The local second-order approximation replaces this constraint with

$\frac{1}{2}\Delta\rho^\top F(\rho)\Delta\rho \le \delta$. Maximizing the linear score objective $(\hat{g}^{\text{score}})^\top\Delta\rho$ under this quadratic constraint gives the TRPO-style normalized natural step [Schulman et al., 2015] $\Delta\rho = \sqrt{2\delta}\,u/\sqrt{u^\top F(\rho)u}$ for $u \neq 0$, and zero otherwise. This scalar normalization fixes the predicted local KL; satisfying the exact KL constraint requires additional verification.

3 Methods

SATR adapts the distributional step to the realized population update and the curvature of each coordinate. We first define its signal-dependent coordinate budgets, then derive the Bernoulli update used for RSNN connectivity search. A separate bitset backend accelerates policy evaluation without changing the sampling distribution or update rule.

3.1 Signal-Adaptive Trust Regions (SATR)

Each iteration updates the parameters ρ of a sampling distribution rather than a single policy parameter vector. Following information-theoretic trust-region methods [Schulman et al., 2015, Abdolmaleki et al., 2018], we measure the resulting distributional displacement using

$$D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho'}) \triangleq \mathbb{E}_{\theta\sim p_\rho}\left[\log \frac{p_\rho(\theta)}{p_{\rho'}(\theta)}\right]. \tag{3}$$

Our design assigns separate KL budgets to individual coordinates so that their steps can respond to both local curvature and the corresponding update components.

Signal energy and statistical decomposition. The finite-population direction u in Eq. 2 varies with the sampled connectivities, rollout stochasticity, and training phase. We use its realized squared norm as a proxy for population signal energy:

$$E \triangleq \|u\|_2^2. \tag{4}$$

To distinguish coherent signal from sampling fluctuations, let $u^{(n)} = \tilde{R}^{(n)}(\theta^{(n)} - \rho)$, so $u = N^{-1}\sum_n u^{(n)}$. Conditioning on the current ρ, define $\mu = \mathbb{E}[u]$ and the zero-mean fluctuations $\xi^{(n)} = u^{(n)} - \mathbb{E}[u^{(n)}]$. Then

$$\mathbb{E}[\|u\|_2^2] = \|\mu\|_2^2 + \frac{1}{N^2}\sum_{n=1}^{N}\mathbb{E}[\|\xi^{(n)}\|_2^2] + \frac{1}{N^2}\sum_{n\neq m}\mathbb{E}[(\xi^{(n)})^\top\xi^{(m)}]. \tag{5}$$

The covariance terms matter because centered ranks depend on the entire population. Under the illustrative noise-only null in which returns are independent of all sampled connectivities, conditioning on the rank weights gives $\mathbb{E}[\|u\|_2^2 \mid \{\tilde{R}^{(n)}\}] = N^{-2}\sum_n(\tilde{R}^{(n)})^2\sum_i \rho_i(1-\rho_i) \le d/(16N)$. Thus population averaging suppresses the noise floor in this null model, whereas a nonzero mean direction contributes $\|\mu\|_2^2$. More generally, the familiar $O(1/N)$ variance scaling requires control of the covariance terms and is not implied by rank shaping alone. Centered ranks remove raw reward-scale dependence, but E still contains sampling variance and is a signal-strength proxy rather than a direct estimate of signal-to-noise ratio or directional agreement.

Coordinate-wise signal-adaptive trust regions. For a factorized family with one scalar parameter per factor, $p_\rho(\theta) = \prod_{i=1}^d p_{\rho_i}(\theta_i)$, the KL decomposes as $D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho'}) = \sum_i D_i(\rho_i, \rho_i')$, where D_i is the KL between the corresponding one-dimensional distributions. We define coordinate-wise budgets proportional to u_i^2 and choose a step aligned with the population direction:

$$\max_{\Delta\rho:\ \rho+\Delta\rho\in\Omega} u^\top\Delta\rho \quad \text{s.t.} \quad D_i(\rho_i, \rho_i + \Delta\rho_i) \le \delta u_i^2, \quad i = 1, \ldots, d, \tag{6}$$

where Ω is the distribution's parameter domain and $\delta > 0$ sets the budget scale. The sum of these coordinate budgets is $\delta\|u\|_2^2$. Small realized update components receive small budgets, while larger components permit larger distributional steps.

Local KL model and the choice of $F^{-1/2}$. For an interior parameter with positive scalar Fisher information, the local expansion is

$$D_i(\rho_i, \rho_i + \Delta\rho_i) = \frac{1}{2} F_i(\rho_i)(\Delta\rho_i)^2 + o\big((\Delta\rho_i)^2\big). \tag{7}$$

Replacing the exact KL in Eq. 6 with this quadratic model yields the separable constraints

$$\frac{1}{2} F_i(\rho_i)(\Delta\rho_i)^2 \le \delta u_i^2, \qquad i = 1, \dots, d. \tag{8}$$

For $u_i \neq 0$, the linear objective is maximized at the quadratic constraint boundary with the sign of u_i. For $u_i = 0$, the constraint requires a zero step. Solving this quadratic model before enforcing the parameter domain gives

$$\Delta\rho_i^\star = \sqrt{\frac{2\delta}{F_i(\rho_i)}}\, u_i = \eta F_i(\rho_i)^{-1/2} u_i, \qquad \eta \triangleq \sqrt{2\delta}. \tag{9}$$

Among positive coordinate-wise scalings $\Delta\rho_i = \alpha_i u_i$, this uniquely selects $\alpha_i = \eta F_i^{-1/2}$ for nonzero components: each predicted KL contribution equals δu_i^2. Equation 9 solves the unprojected local quadratic problem; it does not by itself guarantee feasibility for the parameter domain or the exact KL constraints in Eq. 6.

3.2 SATR-EC: Specialization to Bernoulli Connectivity

For the factorized Bernoulli family, the diagonal Fisher entries are

$$F_i(\rho_i) = \frac{1}{\rho_i(1-\rho_i)}, \tag{10}$$

and, for valid sufficiently small parameter steps, the joint KL has the local approximation

$$D_{\mathrm{KL}}(p_\rho \,\|\, p_{\rho+\Delta\rho}) \approx \frac{1}{2} \sum_{i=1}^{d} \frac{(\Delta\rho_i)^2}{\rho_i(1-\rho_i)}. \tag{11}$$

Substituting Eq. 10 into the local SATR solution gives the implemented SATR-EC rule:

$$\Delta\rho_{\mathrm{SATR}} = \eta F(\rho)^{-1/2} u = \eta \sqrt{\rho \odot (1-\rho)} \odot u. \tag{12}$$

We apply $\rho \leftarrow \mathrm{clip}(\rho + \Delta\rho_{\mathrm{SATR}}, \epsilon, 1-\epsilon)$ with $\epsilon = 10^{-3}$ to keep probabilities away from 0 and 1. Algorithm 1 summarizes the training procedure.

Signal-adaptive local KL and boundary conditioning. Let $Q_\rho(\Delta\rho) = \frac{1}{2}\Delta\rho^\top F(\rho)\Delta\rho$ denote the quadratic KL prediction. Before clipping, its contribution from coordinate i is $\frac{1}{2}F_i(\Delta\rho_{\mathrm{SATR},i})^2 = \frac{\eta^2}{2}u_i^2$. Summing these contributions gives

$$Q_\rho(\Delta\rho_{\mathrm{SATR}}) = \frac{\eta^2}{2} \sum_{i=1}^{d} \frac{\rho_i(1-\rho_i)u_i^2}{\rho_i(1-\rho_i)} = \frac{\eta^2}{2}\|u\|_2^2. \tag{13}$$

The factor $F^{-1/2}$ is therefore selected by a coordinate-wise KL criterion, beyond its property of shrinking steps near a boundary. For a given realized u_i, it removes the explicit dependence of the predicted KL weighting on ρ_i; the direction u itself still depends on the current sampling distribution. This equality concerns the quadratic prediction before clipping. Appendix E evaluates the realized exact KL between consecutive clipped distributions.

The standard EC update has a different local KL weighting:

$$Q_\rho(\eta u) = \frac{\eta^2}{2} \sum_{i=1}^{d} \frac{u_i^2}{\rho_i(1-\rho_i)}. \tag{14}$$

The curvature coefficient is 4 at $\rho_i = 0.5$ and approximately 1001 at $\rho_i = \epsilon = 10^{-3}$. Thus an identical realized component u_i receives approximately 250 times greater local-KL weighting at the clipping boundary than at the center. This is a cross-coordinate scale imbalance, not a proof that EC's expected KL diverges: the moments of its Fisher-preconditioned estimator also depend on ρ. SATR removes this explicit curvature imbalance in the quadratic model, motivating the empirical stability comparisons below.

Comparison of EC, fixed EC+TR, and SATR. Using $u = F(\rho)^{-1}\hat{g}^{\mathrm{score}}$, the three update rules are

$$\Delta\rho_{\mathrm{EC}} = \eta u, \qquad \Delta\rho_{\mathrm{EC+TR}} = \sqrt{2\delta}\,\frac{u}{\sqrt{u^\top F(\rho)u}}, \qquad \Delta\rho_{\mathrm{SATR}} = \eta F(\rho)^{-1/2}u. \tag{15}$$

The EC+TR step is defined as zero when $u = 0$. EC directly applies the natural-gradient direction, retaining the curvature-dependent weighting in Eq. 14. Fixed-budget EC+TR instead uses a single scalar normalization to attain predicted KL δ whenever $u \neq 0$, regardless of the magnitude of the population update. SATR applies coordinate-wise conditioning and uses the predicted total budget $\delta\|u\|_2^2$. This distinguishes its local signal-dependent geometry from both a fixed learning rate and a fixed global KL budget.

3.3 Bitset Acceleration for Binary RSNN Rollouts

To improve rollout throughput, we exploit the binary structure of sampled connectivity and spiking activity. A sampled connectivity remains fixed throughout an episode, so we pack it once into machine words and pack the spike vector at each timestep. This compact representation reduces memory traffic and allows synaptic integration through bitwise operations.

Consider a presynaptic spike vector $s_t \in \{0,1\}^{d_{\mathrm{in}}}$ and the binary mask $m_j \in \{0,1\}^{d_{\mathrm{in}}}$ for output neuron j. For word size w (e.g., $w = 64$), let $B = \lceil d_{\mathrm{in}}/w \rceil$ and denote their packed words by $\{S_{t,b}\}_{b=1}^{B}$ and $\{M_{j,b}\}_{b=1}^{B}$, with unused spike bits padded by zero. The binary dot product is

$$m_j^\top s_t = \sum_{b=1}^{B} \mathrm{popcount}(M_{j,b}\ \&\ S_{t,b})\,, \tag{16}$$

where $\&$ denotes bitwise `AND` and `popcount` counts the set bits in a word. This replaces the dominant dense synaptic multiply–accumulate kernel with $O(B)$ word operations and integer additions per output neuron.

The signed-weight derivation is provided in Appendix A.1.

Packing preserves these binary and signed synaptic sums exactly while reducing connectivity storage and memory bandwidth relative to floating-point arrays. Neuron-state updates, leak integration, and threshold comparisons remain part of the RSNN rollout. The backend changes execution of the policy, while the population estimator and SATR update remain the same.

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, including the most complex 17-DoF environment.

We compare several baselines in our evaluation: **PPO-LSTM** represents the strong baseline in gradient-based RL, utilizing a standard Proximal Policy Optimization agent with LSTM recurrence. To provide a gradient-free biological baseline, **ES-RSNN** consists of an RSNN trained via Evolutionary Strategies [Salimans et al., 2017]. We also include **EC-RSNN** as an ablation baseline, which features an RSNN trained with the original Evolving Connectivity [Wang et al., 2023]. Finally, **SG-RSNN** denotes an RSNN trained under PPO using the surrogate gradient method specifically designed for SNNs. For this baseline, we implemented three surrogate gradient functions: Abs-Beta Esser et al. [2016], SuperSpike Zenke and Ganguli [2018], and $\mathrm{Sigmoid}'$ Zenke and 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. Network architectures and parameter matching are described in Appendix A, and all training hyperparameters are listed in Appendix H.

All methods are implemented in JAX [Bradbury et al., 2018] and evaluated in Brax [Freeman et al., 2021] on the same hardware. The main results in Tables 1 and 4 retain the original evaluations over three independent random seeds, reported as mean $\pm$ standard deviation. Additional five-seed evaluations are reported separately in Appendix D. Population-based methods run 2,000 generations

on Humanoid and 1,000 generations on Hopper and Walker2d. SG-RSNN uses 250k gradient steps on all tasks. PPO-LSTM is evaluated with task-specific wall-clock-matched budgets: 500k and 850k training iterations match bitset SATR-RSNN at populations 4096 and 8192, while the longest PPO budgets match non-bitset SATR-RSNN at population 8192: 2.2M on Humanoid, 1.1M on Hopper, and 1.0M on Walker2d. PPO training-iteration counts differ from total environment transitions; the corresponding budget details are provided in Appendix B.1.

5 Results

5.1 Reward–Runtime Trade-off Against PPO-LSTM

The central empirical question is whether a binary, gradient-free RSNN policy can be competitive with a strong gradient-based recurrent baseline when wall-clock training time is controlled. We therefore place the runtime-matched comparison with PPO-LSTM as the primary result. PPO-LSTM is not evaluated at arbitrary update counts; instead, its training budgets are chosen to match the measured wall-clock time of specific SATR-RSNN configurations: bitset SATR-RSNN* (N=4096), bitset SATR-RSNN* (N=8192), and non-bitset SATR-RSNN (N=8192). The full budget protocol is described in Appendix B.1.

Figure 1 summarizes the overall reward–runtime trade-off. The most challenging task is Humanoid, where the runtime-matched PPO-LSTM learning curves are shown in Figure 2. Table 1 gives the corresponding quantitative comparison. At matched wall-clock time, SATR-RSNN* achieves higher final return than PPO-LSTM under both matched budgets. With the runtime corresponding to PPO-LSTM 500k training iterations, SATR-RSNN* (N=4096) reaches $13{,}072$, compared with PPO-LSTM's $11{,}327$. With the runtime corresponding to PPO-LSTM 850k training iterations, SATR-RSNN* (N=8192) reaches $13{,}860$, compared with PPO-LSTM's $12{,}397$. Moreover, SATR-RSNN* (N=4096) already reaches a slightly higher Humanoid return than the longest PPO-LSTM run, while using $4.9\times$ shorter wall-clock time.

Algorithm 1 SATR-RSNN

Input: $\epsilon = 10^{-3}$, initial $\rho \in [\epsilon, 1-\epsilon]^d$, N, η
repeat
  Sample $\theta^{(1)}, \ldots, \theta^{(N)} \sim \prod_i \mathrm{Bern}(\theta_i; \rho_i)$
  for $n = 1$ **to** N (parallel) **do**
    Roll out episode with $\theta^{(n)}$
    Record return $R(\theta^{(n)})$
  end for
  Compute centered-rank returns $\tilde{R}^{(1)}, \ldots, \tilde{R}^{(N)}$
  $u \leftarrow \frac{1}{N} \sum_{n=1}^{N} \tilde{R}^{(n)} (\theta^{(n)} - \rho)$
  $\Delta\rho \leftarrow \eta \sqrt{\rho \odot (1-\rho)} \odot u$
  $\rho \leftarrow \mathrm{clip}(\rho + \Delta\rho, \epsilon, 1-\epsilon)$
until converged

Table 1: **Humanoid reward–runtime comparison against PPO-LSTM.** Returns are mean $\pm$ standard deviation over three seeds. The first two blocks approximately match wall-clock time; the last compares similar mean returns using different runtimes.

Comparison	Method / config	Hours	Final return
Matched time	PPO-LSTM, 500k	2.8	11327 ± 712
	SATR-RSNN*, 4096	2.9	$\mathbf{13072 \pm 941}$
Matched time	PPO-LSTM, 850k	5.2	12397 ± 1929
	SATR-RSNN*, 8192	5.2	$\mathbf{13860 \pm 1107}$
Matched perf.	PPO-LSTM, 2.2M	14.2	12876 ± 2861
	SATR-RSNN*, 4096	2.9	$\mathbf{13072 \pm 941}$

These results are the main empirical finding of the paper: SATR-RSNN* is not merely stronger than prior population-based RSNN optimizers; it achieves a favorable reward–runtime trade-off relative to PPO-LSTM, a strong gradient-based recurrent baseline, while using binary spiking dynamics and gradient-free distributional optimization.

Wall-Clock Efficiency vs. Environment-Step Sample Efficiency. The primary comparison controls training time because population evaluation and the bitset backend exploit parallel rollout execution. Comparable-environment-step comparisons are provided in Appendix B.1, and the additional five-seed results are reported in Appendix D. Those comparisons show a task-dependent sample-efficiency trade-off: PPO-LSTM achieves higher mean returns on Humanoid and Walker2d, while SATR-RSNN* achieves higher mean returns on Hopper. The reward–runtime advantage should therefore be distinguished from an advantage in return per environment transition.

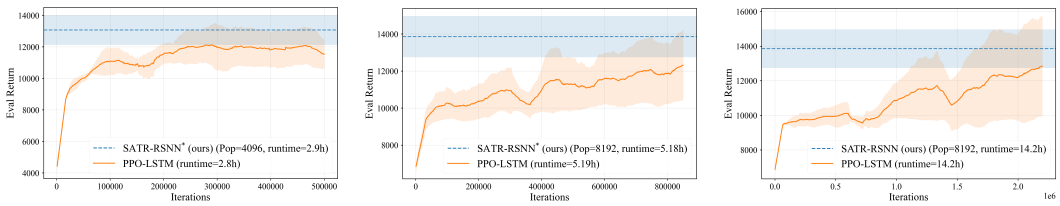

Figure 2: Original Humanoid PPO-LSTM learning curves, plotted against PPO training iterations, with horizontal SATR final-return references from the original three-seed evaluation. Left: PPO-LSTM at 500k iterations (2.8h) and bitset SATR-RSNN* at $N = 4096$ (2.9h). Middle: PPO-LSTM at 850k iterations and bitset SATR-RSNN* at $N = 8192$ (both approximately 5.2h). Right: PPO-LSTM at 2.2M iterations (14.2h) and the $N = 8192$ SATR reference; the corresponding non-bitset SATR runtime is 14.2h (51,098s). Curves are smoothed for visualization; the main-text comparison uses the original three-seed summaries, and the separate five-seed results appear in Appendix D.

5.2 Attribution of the Reward–Runtime Gain

The reward–runtime improvement combines two distinct components: the SATR optimizer and the bitset execution backend. It is important to disentangle them. SATR changes the distributional update rule and primarily affects return, whereas bitset execution changes rollout implementation and primarily affects wall-clock time.

Table 2: SATR-RSNN runtime acceleration on Humanoid (seconds).

Pop	$RSNN^*$	$RSNN\ w/o\ bitset$	$\times$ Faster
512	3168	5212	1.65
1024	4178	6912	1.65
2048	5645	15569	2.76
4096	10431	26944	2.58
8192	18668	51098	2.74

Table 3: Humanoid runtime attribution at $N = 8192$ (seconds).

Opt.	$RSNN^*$	$RSNN\ w/o\ bitset$	$\times$ Faster
SATR	18668	51098	2.74
EC	18806	51761	2.75
Runtime reduction	0.73%	1.28%	-

Table 3 shows that replacing EC with SATR has negligible runtime impact when the backend is fixed: the measured runtime reduction is 0.73% with bitset execution and 1.28% with dense execution. Thus, SATR itself does not accelerate rollout execution. In contrast, Table 2 shows that the bitset backend reduces SATR-RSNN wall-clock time by up to $2.76\times$ by exploiting binary spikes and binary connectivity. The reward–runtime gain therefore arises from complementary effects: SATR improves the final return through a more stable distributional update, while bitset execution improves rollout throughput without changing the policy class, sampling distribution, fitness estimator, or update rule.

5.3 SATR Improves Population-Based RSNN Optimization

Table 4: Performance across tasks and population sizes (mean $\pm$ standard deviation over the original three seeds). Additional five-seed results are reported separately in Appendix D.

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	**13860 ± 1107**	2505 ± 160	2451 ± 463	**2735 ± 160**	3909 ± 396	5077 ± 1544	**5394 ± 402**
4096	10087 ± 1162	10319 ± 1196	**13072 ± 941**	2557 ± 294	2350 ± 405	**2665 ± 188**	2891 ± 167	4605 ± 148	**5143 ± 459**
2048	8558 ± 659	9506 ± 939	**12037 ± 278**	2260 ± 54	2273 ± 677	**2588 ± 79**	2925 ± 295	3372 ± 721	**3856 ± 607**
1024	6031 ± 993	7729 ± 1028	**10520 ± 457**	2096 ± 66	2175 ± 755	**2473 ± 237**	2259 ± 164	2656 ± 509	**3139 ± 806**
512	922 ± 16	5462 ± 434	**8928 ± 119**	1855 ± 56	2036 ± 329	**2356 ± 138**	2047 ± 295	2288 ± 616	**2738 ± 298**
256	823 ± 24	3155 ± 2508	**6656 ± 1643**	1625 ± 203	1875 ± 51	**2238 ± 97**	2054 ± 305	1944 ± 70	**2299 ± 107**
128	750 ± 42	765 ± 36	**4908 ± 32**	1497 ± 56	1724 ± 95	**1949 ± 321**	1683 ± 118	1650 ± 88	**2040 ± 725**

After establishing the comparison with PPO-LSTM, we use ES-RSNN and EC-RSNN as population-based RSNN baselines to isolate the optimizer-level contribution of SATR. These comparisons answer a different question from the PPO-LSTM result: whether the proposed signal-adaptive trust region improves gradient-free optimization of recurrent spiking policies under the same RSNN policy class.

As shown in Table 4, SATR consistently outperforms ES and standard EC across Humanoid, Hopper, and Walker2d. On Humanoid with $N = 8192$, SATR reaches $13{,}860 \pm 1{,}107$, compared with $12{,}544 \pm 1{,}175$ for ES and $12{,}242 \pm 1{,}903$ for EC. The advantage becomes larger under smaller population budgets, where finite-population update noise is more severe. For example, at $N = 512$ on Humanoid, SATR obtains $8{,}928 \pm 119$, whereas EC obtains $5{,}462 \pm 434$ and ES collapses to 922 ± 16. At $N = 128$, SATR still reaches $4{,}908 \pm 32$, while EC remains near 765 ± 36.

Figure 3 shows the original learning curves across population sizes. The local KL model predicts contraction with the realized update energy; Appendix E examines the resulting exact distributional shifts during training.

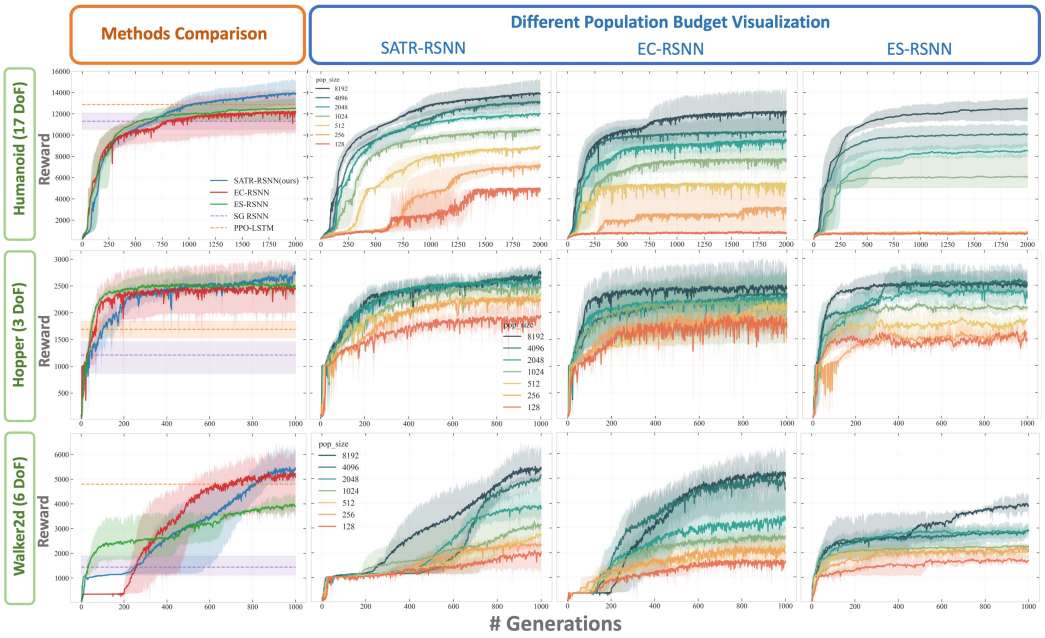

Figure 3: **Performance Benchmarks Across Tasks and Population Scales.** Original three-seed learning curves for three continuous control tasks under varying population budgets. Each color represents a distinct population size, demonstrating the scalability of our approach. SATR (ours) consistently achieves the highest final returns and exhibits superior convergence stability compared to EC and ES.

5.4 Comparison with Surrogate-Gradient RSNNs

We also compare against SG-RSNN baselines trained with PPO. This comparison is included to evaluate whether directly applying gradient-based SNN training is sufficient for long-horizon recurrent spiking control. As shown in Figure S1 (full results in Appendix C.1), SATR outperforms the best-performing SG-RSNN configuration on Humanoid. This result is consistent with the difficulty of training recurrent spiking policies through truncated BPTT and surrogate gradients in long-horizon high-dimensional control.

5.5 Ablation: Fixed vs. Signal-Adaptive Trust Regions

To compare signal-adaptive coordinate scaling with a fixed global KL budget, we evaluate a trust-region variant of EC, denoted EC+TR. EC+TR applies a TRPO-style distribution-space constraint to the Bernoulli search distribution. With u denoting the Fisher-preconditioned EC direction, the normalized fixed-budget step for $u \neq 0$ is

$$\Delta\rho \leftarrow \sqrt{2\delta}\,\frac{u}{\sqrt{u^\top F(\rho)u}}, \tag{17}$$

where $F(\rho)$ is the Bernoulli Fisher matrix; for $u = 0$, we take a zero step. This fixes the predicted quadratic KL budget, rather than the exact post-clipping KL. Table 5 shows that EC+TR can improve

over EC, but its performance depends strongly on the fixed KL budget δ. The best δ changes across population sizes, and overly large budgets can cause collapse. SATR instead scales the allowable local KL displacement with the realized update energy while retaining its learning-rate hyperparameter. As a result, SATR consistently outperforms the tested EC+TR variants across the evaluated population sizes.

Table 5: Original three-seed Humanoid sweep of fixed quadratic KL budgets for EC+TR. Bold marks the highest mean in each population column; underlining marks EC+TR entries between EC and SATR. The additional five-seed comparison is reported in Appendix D.

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

An additional five-seed comparison with the best-tested fixed-budget EC+TR baseline is reported in Appendix D.

5.6 Ablation: Learning-Rate Sensitivity

Table 6 evaluates learning-rate sensitivity on Humanoid. Across $\eta \in \{0.15, 0.2, 0.3, 0.5\}$ and all tested population sizes, SATR consistently outperforms EC and retains higher mean returns at the largest tested learning rate, including in low-population regimes. This supports the boundary-aware scaling in Eq. 12, which dampens aggressive Bernoulli updates near deterministic connectivity probabilities under noisy finite-population estimates.

Table 6: Original three-seed Humanoid learning-rate sweep (mean $\pm$ standard deviation). Additional five-seed Hopper results for lower EC learning rates are reported in Appendix D.

Pop Size	EC Baseline				SATR (Ours)			
	η=0.15 (orig)	η=0.2	η=0.3	η=0.5	η=0.15 (orig)	η=0.2	η=0.3	η=0.5
8192	12242 ± 1903	12700 ± 597	12276 ± 840	10125 ± 2076	$\mathbf{13860 \pm 1107}$	$\mathbf{14296 \pm 931}$	$\mathbf{13538 \pm 904}$	$\mathbf{13407 \pm 1048}$
4096	10319 ± 1196	10077 ± 2889	9270 ± 1909	8254 ± 1081	$\mathbf{13072 \pm 941}$	$\mathbf{12738 \pm 244}$	$\mathbf{12231 \pm 871}$	$\mathbf{11607 \pm 559}$
2048	9506 ± 939	9356 ± 234	8256 ± 624	8198 ± 610	$\mathbf{12037 \pm 278}$	$\mathbf{11078 \pm 734}$	$\mathbf{11054 \pm 593}$	$\mathbf{10275 \pm 608}$
1024	7729 ± 1028	7186 ± 305	6498 ± 1604	6260 ± 1254	$\mathbf{10520 \pm 457}$	$\mathbf{9584 \pm 1005}$	$\mathbf{9040 \pm 556}$	$\mathbf{7387 \pm 2306}$
512	5462 ± 434	4711 ± 569	4781 ± 152	3076 ± 2173	$\mathbf{8928 \pm 119}$	$\mathbf{8195 \pm 290}$	$\mathbf{7585 \pm 949}$	$\mathbf{4430 \pm 3432}$
256	3155 ± 2508	3342 ± 2574	811 ± 130	720 ± 49	$\mathbf{6656 \pm 1643}$	$\mathbf{6479 \pm 1529}$	$\mathbf{5654 \pm 698}$	$\mathbf{3612 \pm 2666}$
128	765 ± 36	753 ± 46	776 ± 37	713 ± 79	$\mathbf{4908 \pm 32}$	$\mathbf{3725 \pm 2994}$	$\mathbf{3263 \pm 2435}$	$\mathbf{2119 \pm 1396}$

Appendix D additionally compares SATR at $\eta = 0.15$ with EC tuned over lower learning rates on Hopper, using five seeds. This tests whether reducing a global EC step size reproduces the benefit of SATR within the tested range.

6 Conclusion and Limitations

We propose Signal-Adaptive Trust Regions (SATR) for population-based learning. 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.

A limitation of this work is that its efficiency advantage is in wall-clock time rather than sample efficiency: at comparable environment-step budgets (Appendix B.1), PPO-LSTM attains higher returns on Humanoid and Walker2d, while SATR-RSNN* is higher on Hopper. In addition, all runtime and energy measurements are specific to our JAX/GPU implementation on a single RTX 4090, and the neuromorphic energy figures are analytical projections rather than hardware measurements. Future work should improve the sample efficiency of population-based RSNN training and evaluate SATR on a broader set of tasks and hardware platforms.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2025ZD0215701; the National Natural Science Foundation of China under Grants T2341003 and U2336214; and the Beijing Natural Science Foundation under Grant L257019. Computational resources were provided by Sapient Intelligence.

We thank Tianshu Sun of the University of Cambridge for her continued involvement in this project and for valuable discussions.

The authors declare no additional competing interests.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Mahmoud Akl, Deniz Ergene, Florian Walter, and Alois Knoll. Toward robust and scalable deep spiking reinforcement learning. *Frontiers in Neurorobotics*, 16:1075647, 2023.

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

Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`. Version 0.3.13.

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.

Krzysztof M Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, and Vikas Sindhwani. From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021.

Jianhao Ding, Jiyuan Zhang, Zhaofei Yu, and Tiejun Huang. Accelerating training of deep spiking neural networks with parameter initialization, 2022. URL `https://openreview.net/forum?id=T8BnDXDTcFZ`.

Steven K. Esser, Paul Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113:11441 – 11446, 2016.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

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

Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. Edge computing with artificial intelligence: A machine learning perspective. *ACM Computing Surveys*, 55(9):1–35, 2023.

Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O Stanley. Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 117–124, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

Dharmendra S Modha, Filipp Akopyan, Alexander Andreopoulos, Rathinakumar Appuswamy, John V Arthur, Andrew S Cassidy, Pallab Datta, Michael V DeBole, Steven K Esser, Carlos Ortega Otero, et al. Neural inference at the frontier of energy, space, and time. *Science*, 382(6668):329–335, 2023.

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

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

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

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

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

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Guobin Shen, Dongcheng Zhao, Yiting Dong, Yang Li, Feifei Zhao, and Yi Zeng. Learning the plasticity: Plasticity-driven learning framework in spiking neural networks. *arXiv preprint arXiv:2308.12063*, 2023.

Amar Shrestha, Haowen Fang, Zaidao Mei, Daniel Patrick Rider, Qing Wu, and Qinru Qiu. A survey on neuromorphic computing: Models and hardware. *IEEE Circuits and Systems Magazine*, 22(2): 6–35, 2022.

Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

Guan Wang, Yuhao Sun, Sijie Cheng, and Sen Song. Evolving connectivity for recurrent spiking neural networks. *Advances in Neural Information Processing Systems*, 36:2991–3007, 2023.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Guanlin Wu, Dongchen Liang, Shaotong Luan, and Ji Wang. Training spiking neural networks for reinforcement learning tasks with temporal coding method. *Frontiers in Neuroscience*, 16:877701, 2022.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35: 20717–20730, 2022.

Zijie Xu, Tong Bu, Zecheng Hao, Jianhao Ding, and Zhaofei Yu. Proxy target: Bridging the gap between discrete spiking neural networks and continuous control. *arXiv preprint arXiv:2505.24161*, 2025.

Zijie Xu, Zihan Huang, Yiting Dong, Kang Chen, Wenxuan Liu, and Zhaofei Yu. Error amplification limits ANN-to-SNN conversion in continuous control. *arXiv preprint arXiv:2601.21778*, 2026a. URL `https://arxiv.org/abs/2601.21778`.

Zijie Xu, Xinyu Shi, Yiting Dong, Zihan Huang, and Zhaofei Yu. CaRe-BN: Precise moving statistics for stabilizing spiking neural networks in reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL `https://openreview.net/forum?id=AaZVrbElhC`.

Jason Yik, Korneel Van den Berghe, Douwe den Blanken, Younes Bouhadjar, Maxime Fabre, Paul Hueber, et al. The NeuroBench framework for benchmarking neuromorphic computing algorithms and systems. *Nature Communications*, 16(1):1545, 2025. doi: 10.1038/s41467-025-56739-4. URL `https://www.nature.com/articles/s41467-025-56739-4`.

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

Luca Zanatta, Francesco Barchi, Simone Manoni, Silvia Tolu, Andrea Bartolini, and Andrea Acquaviva. Exploring spiking neural networks for deep reinforcement learning in robotic tasks. *Scientific Reports*, 14(1):30648, 2024.

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

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

Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. *Advances in Neural Information Processing Systems*, 35:30528–30541, 2022.

A Policy Architecture and Parameter Matching

Table S1 compares the model architectures used in our experiments in terms of parameter count, numerical precision, parameter storage, and the dominant synaptic kernel in the evaluated GPU backend. 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 **RSNN (EC)** and **RSNN* (SATR)** rows share this architecture but differ in both their optimization rule and evaluated 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. 3.3). The backend change preserves the underlying objective and policy class while reducing memory traffic and wall-clock runtime; the optimizer/backend attribution is reported separately in Table 3.

Precision and memory footprint. For EC/SATR, the optimized variables are real-valued Bernoulli probabilities, while sampled binary connectivity or weight encodings can be stored using one bit per parameter. The 24KB reported for RSNN and RSNN* in Table S1 describes packed parameter storage; it excludes neuron state buffers, Bernoulli probabilities, and optimizer state. Binary spikes do not make membrane potentials or other LIF state variables binary. 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 ($\approx$191K) closely match the RSNN parameter budget ($\approx$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: Policy architectures, approximate parameter storage, and dominant synaptic kernels in the evaluated GPU backend. Storage figures exclude recurrent state buffers and training variables.

Model	Hidden size	Params	Precision	Size (KB)	Dominant Synaptic Kernel (Evaluated GPU Backend)
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	$Matmul$

Interpretation of the operation column. The final column of Table S1 identifies the dominant synaptic-accumulation kernel in the evaluated JAX/GPU implementation. It does not specify an algorithmic requirement of SATR or account for all RSNN computation: LIF state updates, leak integration, and threshold comparisons remain present in every RSNN variant. On event-driven neuromorphic hardware, binary spikes can instead trigger sparse synaptic accumulations (ACs), without requiring the packed `AND+popcount` kernel used by our GPU backend.

A.1 Signed synaptic integration by an affine transformation.

For a signed $\{-1, +1\}$ weight representation, the same masked reduction supports both excitatory and inhibitory synaptic inputs. Let $M \in \{0,1\}^{d_{\text{out}} \times d_{\text{in}}}$ encode the signs and let $W = 2M - \mathbf{1}_{d_{\text{out}}} \mathbf{1}_{d_{\text{in}}}^\top \in \{-1, +1\}^{d_{\text{out}} \times d_{\text{in}}}$. The signed input to neuron j is

$$(Ws_t)_j = 2(Ms_t)_j - \mathbf{1}_{d_{\text{in}}}^\top s_t = 2\sum_{b=1}^{B} \text{popcount}(M_{j,b} \,\&\, S_{t,b}) - \sum_{b=1}^{B} \text{popcount}(S_{t,b}). \tag{18}$$

The first term requires one masked `AND+popcount` reduction per output row. The total presynaptic spike count in the second term is computed once per timestep and shared across all output neurons, avoiding separate excitatory and inhibitory masked reductions. This identity treats M as a sign encoding; representing absent edges in addition to signed weights requires a separate connectivity mask.

B Analytical Operation and Energy Estimate

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

$$E_{\text{one}} = E_{\text{upd}} + E_{\text{syn}} = P_u NIS + (P_s + CP_w) \, NRS \approx 2.8 \, \text{mJ}. \tag{19}$$

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

$$E_{\text{tot}} = E_{\text{one}} \, G \, P, \tag{20}$$

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

Table S3: **Energy comparison on Humanoid.** The first row reports an analytical on-chip energy estimate for SATR-RSNN based on published neuromorphic operation costs. The second row reports measured end-to-end GPU energy for the actual SATR-RSNN and PPO-LSTM training runs on the same server. SATR-RSNN* with $N = 4096$ achieves comparable performance to the longest PPO-LSTM run while consuming $4.9\times$ **less measured GPU energy** on the GPU.

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]	1.8	2.5	4.7	8.4	23.0

GPU energy measurement. In addition to the analytical on-chip estimate for SATR-RSNN, we also measure end-to-end GPU energy for the actual training runs of both SATR-RSNN and PPO-LSTM on the same server. These GPU measurements include environment simulation, rollout execution, and optimization, and therefore provide a like-for-like measurement of the experiments as executed. As shown in Table S3, on Humanoid, bitset SATR-RSNN* with $N = 4096$ consumes 4.7 MJ of measured GPU energy, whereas the longest PPO-LSTM run consumes 23.0 MJ. This SATR configuration reaches comparable or slightly higher final return than PPO-LSTM, while using $4.9\times$ lower measured GPU energy.

Scope of the analytical on-chip estimate. Separately, Table S3 reports an analytical on-chip energy estimate for projected SATR-RSNN execution on Loihi-based neuromorphic reference model. We compute this estimate by multiplying the operation counts in Table S2 by published Loihi per-operation energy values, including synaptic spike operations, within-tile spike delivery, and neuron updates. These on-chip values are intended as order-of-magnitude projections rather than direct hardware measurements of our policy. They are therefore reported separately from the measured end-to-end GPU training energy. Future work will validate the projected estimates through direct neuromorphic deployment.

Hardware-independent complexity metrics (NeuroBench perspective). NeuroBench separates hardware-independent algorithmic metrics—including model footprint, connection sparsity, activation sparsity, and synaptic operations—from hardware-dependent execution measurements [Yik et al., 2025]. We use this distinction to interpret the parameter counts in Table S1 and the analytical assumptions in Table S2. The quantities below are analytical estimates, not results obtained by running the NeuroBench evaluation harness.

- **Parameter storage and precision.** Approximately 193K binary parameters require about 24KB when packed, compared with approximately 764KB for the 191K-parameter FP32 LSTM (about $32\times$ less parameter storage). This is a parameter-only estimate; NeuroBench's full model footprint also includes model buffers, including recurrent states.
- **Activation and connection sparsity.** The average firing rate $R = 0.025$ spikes per neuron per simulation timestep reported in Table S2 corresponds to $1 - R = 97.5\%$ zero spike activations. We use this rate to estimate the event-driven workload below. Connection sparsity is a separate quantity: a dense signed $\{-1, +1\}$ matrix has no zero weights, regardless of the proportion of positive and negative signs.
- **Operation counts per analytical rollout.** With $N = 256$ neurons, $I = 4$ state-update operations per neuron per timestep, and $S = 33{,}200$ simulation timesteps, the assumed state-update count is

  $$NIS = 33{,}996{,}800 \approx 3.40 \times 10^7.$$

  For the assumed recurrent fan-out $C = 128$ and firing rate R, the corresponding expected event-driven synaptic accumulation count is

  $$NRCS = 27{,}197{,}440 \approx 2.72 \times 10^7 \text{ ACs}.$$

  Here C counts nonzero outgoing connections; it is not determined by the excitatory/inhibitory ratio. These counts cover the recurrent operations represented by the analytical model, excluding read-in/read-out processing, environment simulation, and optimization. For scale, applying one MAC per parameter per environment step to a 191K-parameter dense recurrent ANN over $1{,}000$ environment steps gives 1.91×10^8 MACs. This last quantity is a parameter-count proxy, rather than a profiled operation count for the full PPO-LSTM implementation.

Neuron-state updates are reported separately because they are not part of NeuroBench's synaptic-operation metric. These analytical counts describe potential event-driven execution; they do not measure the instructions executed by the GPU bitset backend or establish a measured neuromorphic energy advantage.

B.1 Additional PPO-LSTM Comparisons

PPO-LSTM is included as a strong recurrent ANN baseline trained with PPO [Schulman et al., 2017]. For each task, PPO-LSTM is trained with task-specific budgets matched to the measured wall-clock time of three SATR-RSNN configurations: bitset SATR-RSNN* at $N = 4096$, bitset SATR-RSNN* at $N = 8192$, and non-bitset SATR-RSNN at $N = 8192$. Table S4 reports the full task-wise wall-clock-matched results. SATR-RSNN* denotes the bitset backend, while "w/o bitset" denotes dense execution with the same SATR optimizer.

Additional environment-step results. Table S5 reports additional PPO-LSTM comparisons under comparable environment-step budgets.

Table S4: Full wall-clock-matched comparison between PPO-LSTM and SATR-RSNN. Each PPO-LSTM budget is selected to match the runtime of the SATR-RSNN configuration shown in the second column. SATR entries are reported as mean ± standard deviation over 3 seeds.

Task	SATR-RSNN setting matched in wall-clock	PPO-LSTM budget	PPO-LSTM return	SATR-RSNN return
Humanoid	$RSNN^*$, $N = 4096$	500k	11,327	$13{,}072 \pm 941$
	$RSNN^*$, $N = 8192$	850k	12,397	$13{,}860 \pm 1107$
	w/o bitset, $N = 8192$	2.2M	12,876	$13{,}860 \pm 1107$
Hopper	$RSNN^*$, $N = 4096$	500k	1,616	$2{,}665 \pm 188$
	$RSNN^*$, $N = 8192$	850k	1,665	$2{,}735 \pm 160$
	w/o bitset, $N = 8192$	1.1M	1,742	$2{,}735 \pm 160$
Walker2d	$RSNN^*$, $N = 4096$	500k	3,977	$5{,}143 \pm 459$
	$RSNN^*$, $N = 8192$	850k	4,095	$5{,}394 \pm 402$
	w/o bitset, $N = 8192$	1.0M	4,652	$5{,}394 \pm 402$

Table S5: Additional PPO-LSTM comparison under comparable environment-step budgets. SATR entries are reported as mean ± standard deviation over 3 seeds.

Task	Env-step budget	Method	Env. steps	Wall-clock (s)	Final return
Humanoid	≈1.02B	PPO-LSTM, 500k steps	1.02B	10,080	11,327
		SATR-RSNN*, $N = 512$	1.02B	3,168	$8{,}928 \pm 119$
	≈4–4.5B	PPO-LSTM, 2.2M steps	4.5B	51,120	12,876
		SATR-RSNN*, $N = 2048$	4.1B	5,645	$12{,}037 \pm 278$
Walker2d	≈1.02B	PPO-LSTM, 500k steps	1.02B	5,315	3,977
		SATR-RSNN*, $N = 1024$	1.02B	1,713	$3{,}139 \pm 806$
	≈2B	PPO-LSTM, 1.0M steps	2.0B	10,579	4,652
		SATR-RSNN*, $N = 2048$	2.0B	2,557	$3{,}856 \pm 607$
Hopper	≈1.02B	PPO-LSTM, 500k steps	1.02B	5,070	1,616
		SATR-RSNN*, $N = 1024$	1.02B	1,807	$2{,}473 \pm 237$
	≈2B	PPO-LSTM, 1.1M steps	2.2B	11,114	1,742
		SATR-RSNN*, $N = 2048$	2.0B	2,420	$2{,}588 \pm 79$

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.

C Performance Comparison with PPO and SG

C.1 Full SG Results

Figure S1 reports SG-RSNN results on Humanoid for the three surrogate functions (AbsBeta, SuperSpike, and Sigmoid′) across their tested parameter settings.

C.2 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**:

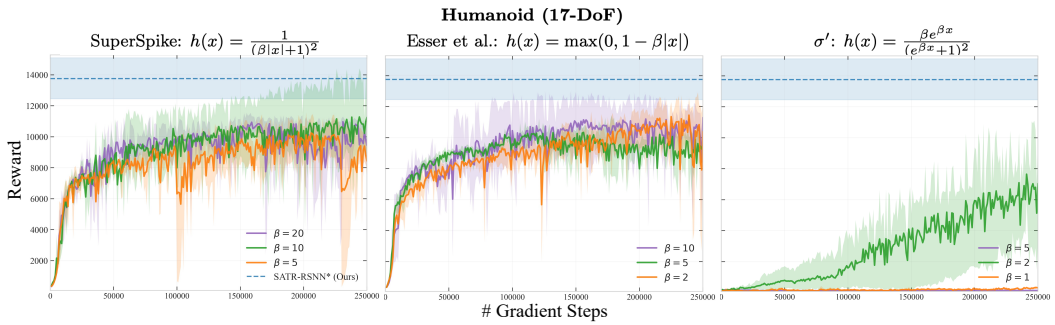

Figure S1: Performance of different surrogate gradient methods with different training parameters in the Humanoid task, which indicates our method beats the best parameter combination of surrogate gradient methods.

- **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 Additional Five-Seed Experiments

This appendix reports additional evaluations over five independent random seeds. The main-text results and existing learning-curve figures retain their original three-seed evaluations. We report the two sets separately, using mean $\pm$ standard deviation for the five-seed summaries below.

D.1 Reward–Runtime Comparison with PPO-LSTM

The five-seed Humanoid comparison preserves the reward–runtime pattern in the main text. Under approximately matched training times, SATR-RSNN* obtains $13{,}043 \pm 912$ versus PPO-LSTM's $11{,}420 \pm 650$ at 2.9 versus 2.8 hours, and $13{,}820 \pm 977$ versus $12{,}520 \pm 1{,}740$ at 5.2 hours. At $N = 4096$, SATR-RSNN* reaches a mean return comparable to PPO-LSTM's longest run ($13{,}030 \pm 2{,}560$), with a $4.9\times$ shorter training runtime. Table S6 reports these comparisons; the existing plots have not been re-estimated from five-seed trajectories.

D.2 Population-Based RSNN Results

Table S7 reports five-seed results for all 21 task–population settings. SATR achieves the highest mean return in every setting, with a median relative improvement of 17.6% over the stronger of ES and EC in each setting. The task-specific medians are 41.0% on Humanoid, 13.0% on Hopper, and 12.5% on Walker2d. These statistics summarize the additional five-seed evaluation and do not replace the three-seed measurements in Table 4.

D.3 Wall-Clock Efficiency vs. Environment-Step Sample Efficiency

Table S8 compares PPO-LSTM and bitset SATR-RSNN* under comparable environment-transition budgets. PPO-LSTM achieves higher mean returns on Humanoid and Walker2d, whereas SATR-

Table S6: Humanoid reward–runtime comparison against PPO-LSTM (mean ± standard deviation over five independent seeds). The first two blocks approximately match training time; the last compares similar mean final returns. PPO budgets denote training iterations.

Comparison	Method / Config	Hours	Final Return
Matched time	PPO-LSTM, 500k iterations	2.8	$11,420 \pm 650$
	SATR-RSNN*, $N = 4096$	2.9	$\mathbf{13,043 \pm 912}$
Matched time	PPO-LSTM, 850k iterations	5.2	$12,520 \pm 1,740$
	SATR-RSNN*, $N = 8192$	5.2	$\mathbf{13,820 \pm 977}$
Matched perf.	PPO-LSTM, 2.2M iterations	14.2	$13,030 \pm 2,560$
	SATR-RSNN*, $N = 4096$	**2.9**	$\mathbf{13,043 \pm 912}$

Table S7: Performance summary across tasks and population sizes (mean ± standard deviation over 5 independent random seeds). SATR achieves the highest mean return across all 21 settings, with a median improvement of 17.6% over the stronger of ES and EC (41.0% on Humanoid, 13.0% on Hopper, and 12.5% on Walker2d).

Population	ES	EC	SATR (Ours)
Humanoid (17-D o F)			
8192	$12,415 \pm 1,046$	$12,410 \pm 1,705$	$\mathbf{13,820 \pm 977}$
4096	$10,768 \pm 1,843$	$10,082 \pm 921$	$\mathbf{13,043 \pm 912}$
2048	$9,066 \pm 1,167$	$9,630 \pm 1,108$	$\mathbf{12,102 \pm 310}$
1024	$5,622 \pm 614$	$7,443 \pm 874$	$\mathbf{10,498 \pm 484}$
512	$1,974 \pm 1,249$	$5,655 \pm 551$	$\mathbf{9,022 \pm 174}$
256	796 ± 56	$3,009 \pm 1,786$	$\mathbf{6,578 \pm 1,565}$
128	725 ± 105	759 ± 39	$\mathbf{4,925 \pm 48}$
Hopper (3-DoF)			
8192	$2,549 \pm 204$	$2,505 \pm 546$	$\mathbf{2,728 \pm 136}$
4096	$2,508 \pm 245$	$2,381 \pm 352$	$\mathbf{2,664 \pm 71}$
2048	$2,295 \pm 89$	$2,238 \pm 426$	$\mathbf{2,593 \pm 165}$
1024	$2,051 \pm 93$	$2,217 \pm 437$	$\mathbf{2,454 \pm 105}$
512	$1,906 \pm 108$	$1,998 \pm 365$	$\mathbf{2,349 \pm 67}$
256	$1,705 \pm 265$	$1,922 \pm 64$	$\mathbf{2,252 \pm 74}$
128	$1,483 \pm 96$	$1,661 \pm 76$	$\mathbf{1,955 \pm 462}$
Walker2d (6-DoF)			
8192	$3,982 \pm 413$	$5,043 \pm 842$	$\mathbf{5,380 \pm 388}$
4096	$3,058 \pm 334$	$4,529 \pm 186$	$\mathbf{5,088 \pm 289}$
2048	$2,973 \pm 388$	$3,526 \pm 577$	$\mathbf{3,931 \pm 683}$
1024	$2,366 \pm 270$	$2,563 \pm 596$	$\mathbf{3,195 \pm 810}$
512	$2,123 \pm 372$	$2,355 \pm 542$	$\mathbf{2,886 \pm 445}$
256	$2,012 \pm 273$	$1,826 \pm 90$	$\mathbf{2,263 \pm 153}$
128	$1,641 \pm 92$	$1,702 \pm 68$	$\mathbf{2,023 \pm 288}$

RSNN* achieves higher mean returns on Hopper. SATR-RSNN* completes these runs $2.8\times$–$9.1\times$ faster. The larger Humanoid and Hopper budgets are approximate matches: PPO-LSTM uses 4.5B and 2.2B transitions, versus 4.1B and 2.0B for SATR, respectively. These results distinguish return per environment transition from return within a given runtime; the primary comparison in the main text concerns the latter.

D.4 Fixed-Budget Trust-Region Control

We evaluate the best-tested fixed-budget EC+TR baseline against SATR over five independent seeds on Humanoid. At $N \in \{256, 512, 1024\}$, EC+TR achieves $4{,}785 \pm 197$, $6{,}394 \pm 1{,}994$, and $8{,}447 \pm 268$, whereas SATR reaches $6{,}578 \pm 1{,}565$, $9{,}022 \pm 174$, and $10{,}498 \pm 484$, respectively (Table S9). The corresponding mean-return improvements are 37.5%, 41.1%, and 24.3%. This supports SATR's advantage over the tested fixed-budget comparator. Because SATR changes both

Table S8: PPO-LSTM and bitset SATR-RSNN* under comparable environment-transition budgets. All SATR and Humanoid PPO-LSTM returns report mean ± standard deviation over five seeds; other PPO-LSTM entries retain their original evaluation summaries. The two budget columns make approximate matches explicit.

Task	PPO Steps	SATR Steps	PPO Return	SATR (N: return)	PPO (s)	SATR (s) [Speedup]
Humanoid	1.02B	1.02B	$\mathbf{11,420 \pm 650}$	$N = 512$: $9,022 \pm 174$	10,080	$\mathbf{3,168}$ [3.2×]
Humanoid	4.5B	4.1B	$\mathbf{13,030 \pm 2,560}$	$N = 2048$: $12,102 \pm 310$	51,120	$\mathbf{5,645}$ [9.1×]
Walker2d	1.02B	1.02B	$\mathbf{3,977 \pm 770}$	$N = 1024$: $3,195 \pm 810$	5,315	$\mathbf{1,713}$ [3.1×]
Walker2d	2.0B	2.0B	$\mathbf{4,652 \pm 1,132}$	$N = 2048$: $3,931 \pm 683$	10,579	$\mathbf{2,557}$ [4.1×]
Hopper	1.02B	1.02B	$1,616 \pm 127$	$N = 1024$: $\mathbf{2,454 \pm 105}$	5,070	$\mathbf{1,807}$ [2.8×]
Hopper	2.2B	2.0B	$1,742 \pm 158$	$N = 2048$: $\mathbf{2,593 \pm 165}$	11,114	$\mathbf{2,420}$ [4.6×]

Table S9: Additional Humanoid comparison with the best-tested fixed-budget EC+TR baseline (mean ± standard deviation over five seeds).

Method	$N = 256$	$N = 512$	$N = 1024$
EC+TR (best tested)	$4{,}785 \pm 197$	$6{,}394 \pm 1{,}994$	$8{,}447 \pm 268$
SATR	$\mathbf{6{,}578 \pm 1{,}565}$	$\mathbf{9{,}022 \pm 174}$	$\mathbf{10{,}498 \pm 484}$
SATR improvement	37.5%	41.1%	24.3%

coordinate scaling and total-budget adaptivity, this comparison does not isolate either mechanism alone.

D.5 Lower Learning-Rate Control for EC

We test whether a smaller global EC learning rate can account for SATR's advantage. Table S10 compares SATR at $\eta = 0.15$ with the highest-mean EC configuration among $\eta \in \{0.05, 0.10, 0.15\}$ over five seeds on Hopper. SATR exceeds this tuned EC baseline at all seven population sizes, with mean-return gains of 8.9%–17.0%. Within this tested range, reducing EC's global step size does not reproduce SATR's performance.

E Empirical Mechanism Verification: Realized Exact KL Diagnostic

We measure exact distributional movement to test whether SATR's local scaling also reduces finite-step KL in practice. On Humanoid, we record the exact forward joint Bernoulli KL between consecutive post-update, post-clipping sampling distributions at every generation of a 2,000-generation run ($\eta = 0.15$ for EC and SATR):

$$D_{\mathrm{KL}}\big(p_{\rho^{(t)}} \,\big\|\, p_{\rho^{(t+1)}}\big) = \sum_{i=1}^{d} \left[\rho_i^{(t)} \log \frac{\rho_i^{(t)}}{\rho_i^{(t+1)}} + (1 - \rho_i^{(t)}) \log \frac{1 - \rho_i^{(t)}}{1 - \rho_i^{(t+1)}} \right]. \tag{21}$$

This quantity sums KL across approximately 193K connectivity coordinates and includes the effect of clipping to $[\epsilon, 1 - \epsilon]$. It is distinct from the local second-order approximation in Eq. 11. We compare EC, a shared fixed-budget EC+TR configuration selected from the ablation, and SATR. EC+TR fixes its local quadratic prediction before clipping; this does not guarantee an equal exact KL after the finite update.

Table S11 reports a single-run diagnostic for each method and population setting. Over the final 200 updates, EC has mean joint KL of 519.65–527.69 nats, EC+TR has 330.20–355.38 nats, and SATR has 58.19–59.70 nats across $N \in \{256, 512, 1024, 2048\}$. Relative to EC at the same population size, EC+TR reduces the final-window mean KL by 31.6%–37.3%, while SATR reduces it by 88.5%–88.9%. EC+TR's corresponding values remain 5.67×–5.95× those of SATR. Across the full diagnostic grid $N \in \{128, 256, 512, 1024, 2048, 4096, 8192\}$, the reported final-window SATR means range from 54.81 to 59.70 nats. The 50-generation moving means at generations 200, 1000, and 2000 also show smaller distributional steps for SATR. These observations support the proposed reduction in realized distributional movement; the diagnostic does not establish an exact KL bound or explain return differences on its own.

Table S10: Lower learning-rate control on Hopper (mean $\pm$ standard deviation over five seeds). Best EC reports the highest mean among $\eta \in \{0.05, 0.10, 0.15\}$; SATR uses $\eta = 0.15$.

Population	Selected EC η	Best EC	SATR	Improvement
128	0.05	$1{,}756 \pm 116$	$\mathbf{1{,}955 \pm 462}$	11.3%
256	0.10	$1{,}924 \pm 156$	$\mathbf{2{,}252 \pm 74}$	17.0%
512	0.05	$2{,}023 \pm 155$	$\mathbf{2{,}349 \pm 67}$	16.1%
1024	0.10	$2{,}246 \pm 140$	$\mathbf{2{,}454 \pm 105}$	9.3%
2048	0.05	$2{,}281 \pm 375$	$\mathbf{2{,}593 \pm 165}$	13.7%
4096	0.15	$2{,}381 \pm 352$	$\mathbf{2{,}664 \pm 71}$	11.9%
8192	0.15	$2{,}505 \pm 546$	$\mathbf{2{,}728 \pm 136}$	8.9%

Table S11: Exact post-clipping joint Bernoulli KL on Humanoid, in nats (one run per method and population size). Left: range across $N \in \{256, 512, 1024, 2048\}$ of the 50-generation moving mean at representative generations. Right: mean over the final 200 updates for each population size. Percentage reductions use matched-population final-window means.

Method	**50-generation moving-mean range**			**Final 200-update mean**			
	Gen 200	Gen 1000	Gen 2000	$N = 256$	$N = 512$	$N = 1024$	$N = 2048$
EC	427.02–438.08	503.41–512.22	521.37–529.24	519.65	525.08	527.69	526.58
EC+TR (fixed)	343.85–357.34	330.29–350.41	330.42–358.74	355.38	351.71	337.61	330.20
SATR	29.89–30.20	48.21–50.09	59.00–60.15	59.70	59.69	59.22	58.19
SATR reduction vs. EC	—	—	—	88.5%	88.6%	88.8%	88.9%

F 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 for the predicted KL budget; it does not enforce the exact post-clipping KL divergence.

F.1 KL trust region in distribution space

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

$$D_{\mathrm{KL}}\big(p_\rho \,\|\, p_{\rho'}\big) \le \delta, \qquad \rho' = \rho + \Delta\rho. \tag{22}$$

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

$$D_{\mathrm{KL}}\big(p_\rho \,\|\, p_{\rho+\Delta\rho}\big) = \frac{1}{2}\,\Delta\rho^\top F(\rho)\,\Delta\rho \;+\; o(\|\Delta\rho\|^2), \tag{23}$$

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

$$F(\rho) = \mathbb{E}_{\theta\sim p_\rho}\big[\nabla_\rho \log p_\rho(\theta)\,\nabla_\rho \log p_\rho(\theta)^\top\big], \tag{24}$$

the constraint (22) reduces (up to second order) to the quadratic trust region

$$\frac{1}{2}\,\Delta\rho^\top F(\rho)\,\Delta\rho \le \delta. \tag{25}$$

F.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:

$$\mathcal{L}(\rho + \Delta\rho) \approx \mathcal{L}(\rho) + (\hat{g}^{\mathrm{score}})^\top \Delta\rho, \tag{26}$$

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

$$\max_{\Delta\rho} \quad (\hat{g}^{\mathrm{score}})^\top \Delta\rho \quad \text{s.t.} \quad \frac{1}{2}\,\Delta\rho^\top F(\rho)\,\Delta\rho \le \delta. \tag{27}$$

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

F.3 Closed-form normalized natural step

For $F \succ 0$, $\delta > 0$, and $\hat{g}^{\mathrm{score}} \neq 0$, form the Lagrangian

$$\mathcal{J}(\Delta\rho, \lambda) = (\hat{g}^{\mathrm{score}})^\top \Delta\rho - \lambda\left(\frac{1}{2}\Delta\rho^\top F \Delta\rho - \delta\right), \qquad \lambda \ge 0. \tag{28}$$

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

$$\nabla_{\Delta\rho}\mathcal{J} = \hat{g}^{\mathrm{score}} - \lambda F \Delta\rho = 0 \quad \Rightarrow \quad \Delta\rho = \frac{1}{\lambda}\,F^{-1}\hat{g}^{\mathrm{score}}. \tag{29}$$

At optimum the constraint is active (unless $\hat{g}^{\mathrm{score}} = 0$), hence

$$\frac{1}{2}\Delta\rho^\top F \Delta\rho = \delta. \tag{30}$$

Substituting (29) into (30) gives

$$\frac{1}{2}\left(\frac{1}{\lambda}F^{-1}\hat{g}^{\mathrm{score}}\right)^\top F \left(\frac{1}{\lambda}F^{-1}\hat{g}^{\mathrm{score}}\right) = \frac{1}{2\lambda^2}\,(\hat{g}^{\mathrm{score}})^\top F^{-1}\hat{g}^{\mathrm{score}} = \delta, \tag{31}$$

so that

$$\lambda = \sqrt{\frac{(\hat{g}^{\mathrm{score}})^\top F^{-1}\hat{g}^{\mathrm{score}}}{2\delta}}. \tag{32}$$

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

$$\Delta\rho = \sqrt{2\delta}\;\frac{F^{-1}\hat{g}^{\mathrm{score}}}{\sqrt{(\hat{g}^{\mathrm{score}})^\top F^{-1}\hat{g}^{\mathrm{score}}}}. \tag{33}$$

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

$$u \triangleq F^{-1}\hat{g}^{\text{score}} \tag{34}$$

the Fisher-preconditioned (natural) ascent direction in distribution space. Under this convention, (33) becomes

$$\Delta\rho = \sqrt{2\delta}\,\frac{u}{\sqrt{u^\top F u}}, \tag{35}$$

which is exactly the update used by EC+TR in the main text for $u \neq 0$; when $u = 0$, we take $\Delta\rho = 0$. For $u \neq 0$, before clipping, (35) gives the exact quadratic identity $\frac{1}{2}\Delta\rho^\top F \Delta\rho = \delta$. Only its interpretation as the exact KL divergence is approximate.

F.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 $\rho_i \in (0, 1)$,

$$\log p_\rho(\theta) = \sum_{i=1}^{d} \left[\theta_i \log \rho_i + (1 - \theta_i)\log(1 - \rho_i)\right], \tag{36}$$

and

$$\frac{\partial}{\partial \rho_i} \log p_\rho(\theta) = \frac{\theta_i - \rho_i}{\rho_i(1 - \rho_i)}. \tag{37}$$

Therefore the Fisher information is diagonal:

$$F_{ij}(\rho) = \begin{cases} \frac{1}{\rho_i(1-\rho_i)}, & i = j \\ 0, & i \neq j \end{cases} \tag{38}$$

Substituting (38) into (35) yields a single scalar normalization of the natural-gradient direction for EC+TR.

F.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 (23), which turns the constraint into the quadratic form (25). The resulting trust-region subproblem (27) has the analytic solution (35), so the step size is obtained in closed form without bisection or line search for this quadratic model. The exact KL after the update and probability clipping can differ from the prescribed budget, as measured in Appendix E.

G 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

H 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 S12: Hyperparameters for SATR (ours) and Evolving Connectivity (EC)

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

Table S13: Hyperparameters for Evolution Strategies

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

Table S14: 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 S15: 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† R_{in}	$0.15 \cdot \tau_m \sqrt{\frac{2}{d_{in}}}$
Hidden membrane resistance† R_h	$1.0 \cdot \frac{\tau_m}{\tau_{syn}} \sqrt{\frac{2}{d_h}}$
Output membrane resistance† R_{out}	$5.0 \cdot \tau_{out} \sqrt{\frac{2}{d_h}}$
Long short-term memory (LSTM)	
Hidden size	128

†Resistances are set following Ding et al. [2022] to preserve variance.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 discuss the limitations in the conclusions section.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: A full set of assumption and proof is provided in method section and appendix.

Guidelines:

- The answer [N/A] 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 a detailed description of methods and provide the source code for reproduction.

   Guidelines:

   - The answer [N/A] 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 the code in a Github repository.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All details are provided.

   Guidelines:

   - The answer [N/A] 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 main performance comparisons use 3 seeds, with supplementary 5-seed results in the appendix. Means and standard deviations are reported for these comparisons; the exact-KL mechanism diagnostic is explicitly identified as single-run.

   Guidelines:

   - The answer [N/A] 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: Yes. The compute resource is listed in appendix.

   Guidelines:

   - The answer [N/A] 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: We conform the code of ethics in this paper.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: This work is a work on RL in simulated environment. There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks

    Guidelines:

    - The answer [N/A] 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: Packages used in this work is correctly listed and cited.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: We provide a detailed description and readme in the provided code.

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM is used for editing.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.