EVOLUTION ONLY WHEN NEEDED: TRIGGERED EVOLUTIONARY DATA INJECTION FOR OFF-POLICY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Evolutionary reinforcement learning (ERL) can expand exploration for off-policy learners, but maintaining a population throughout training requires repeated auxiliary rollouts even when the learner is still improving. Moreover, indiscriminately inserting population-generated trajectories into replay can shift the replay distribution away from the current learner and degrade off-policy updates. We propose Triggered Evolutionary Data Injection (TEDI), which treats evolutionary assistance as an on-demand replay-data intervention rather than a persistent policy-optimization process. TEDI jointly decides when to invoke evolutionary exploration, which generated experience should enter replay, and how much influence each intervention may have. When persistent learning stagnation is detected, TEDI creates a temporary population of parameter-perturbed policy probes, collects their trajectories outside the main replay buffer, filters candidate experience at both transition and trajectory levels, and injects only a bounded, balanced subset before discarding the population. Thus, evolutionary search affects learning only through selectively admitted replay data. Across four MuJoCo continuous-control tasks, TEDI achieves the highest mean peak return in comparisons with six baselines. Ablation studies show that triggering, filtering, and bounded injection each contribute to performance, while runtime comparisons show that TEDI has the lowest average training time among the compared ERL methods.

1 INTRODUCTION

Off-policy deep reinforcement learning (RL) improves data efficiency by reusing previously collected interactions through experience replay (Fedus et al., 2020; Fujimoto et al., 2018). However, replay can also limit improvement when the buffer provides little behavioral support for qualitatively different actions or trajectories (Fujimoto et al., 2019; Kumar et al., 2019). In this case, policy updates may continue to refine an already visited behavioral mode without collecting evidence for better alternatives. Action-space perturbations offer a simple way to explore, but they do not necessarily induce coherent behavioral changes over an entire rollout. Parameter-space perturbations provide a complementary mechanism by producing temporally consistent policy-level exploration (Plappert et al., 2018).

Evolutionary reinforcement learning (ERL) exploits this complementarity by coupling population-based policy search with gradient learning. The original ERL framework shares population experience with an off-policy learner (Khadka & Tumer, 2018), CEM-RL combines cross-entropy search with actor updates (Pourchot & Sigaud, 2019), and later methods improve variation operators, representation sharing, and population interaction (Bodnar et al., 2020; Hao et al., 2023; Li et al., 2024). These methods show that population search can provide behavioral alternatives that are hard to obtain from local policy updates alone. This benefit, however, comes with two related costs. First, maintaining and evaluating a population throughout training requires repeated auxiliary rollouts, even when the gradient learner is still making progress. Second, population-generated trajectories may differ substantially from the learner's current replay distribution; inserting such heterogeneous experience indiscriminately can increase replay mismatch and degrade off-policy learning (Zheng & Cheng, 2023).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

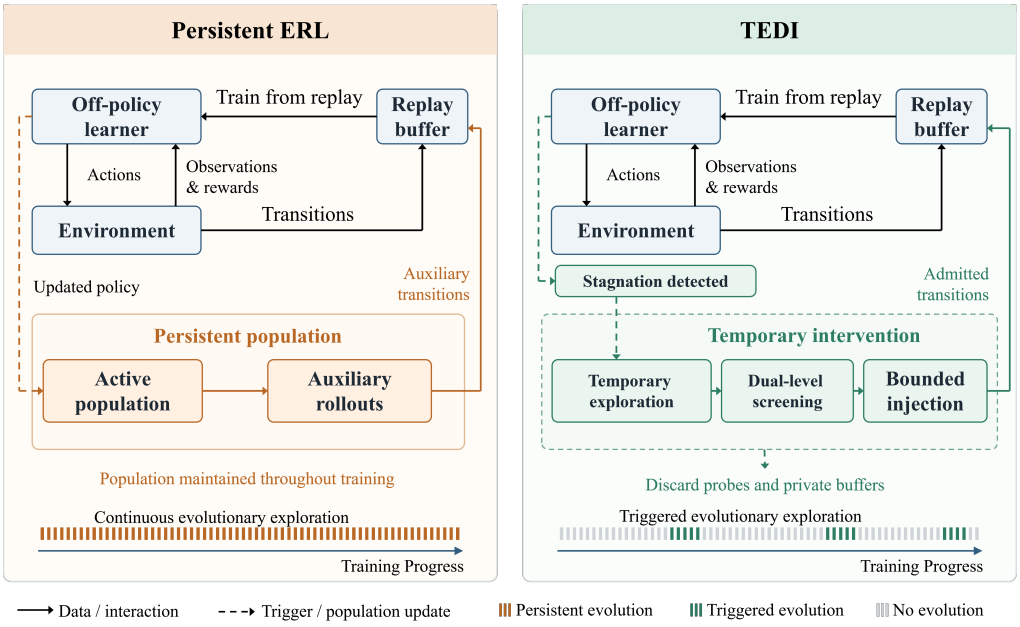


Figure 1: Persistent population assistance versus TEDI. Persistent ERL maintains and repeatedly evaluates an active evolutionary population throughout training. In contrast, TEDI activates temporary policy probes only when persistent learning stagnation is detected, evaluates their trajectories outside the main replay buffer, admits only a bounded and balanced subset of qualified experience, and then discards the temporary population before ordinary off-policy learning resumes.

Prior work has shown that evolutionary evaluation need not remain active throughout training. E-ERL, for example, invokes population evaluation only when the reinforcement learner has difficulty improving (Wu et al., 2022). Yet deciding *when* to activate evolution addresses only one part of the problem. Once additional trajectories are generated, the algorithm must still decide *which* experience should enter replay and *how much* influence a single intervention should have on subsequent learning. We therefore view evolutionary assistance as a replay-data intervention governed by three coupled decisions: activation, experience admission, and event-level influence.

We propose **Triggered Evolutionary Data Injection** (TEDI), which jointly controls these three decisions. During ordinary training, TEDI maintains only the underlying off-policy learner. When persistent learning stagnation is detected, it constructs a temporary population of parameter-perturbed policy probes and evaluates their trajectories outside the main replay buffer. Candidate experience is then screened at both transition and trajectory levels, and only a bounded, balanced subset is admitted to replay. The temporary population is discarded after the intervention, and ordinary off-policy learning resumes. Thus, evolutionary search serves as an on-demand source of selectively admitted replay data rather than as a continuously active policy optimizer.

Figure 1 illustrates the difference between persistent population assistance and TEDI. Persistent ERL repeatedly pays the cost of population maintenance and evaluation, whereas TEDI invokes temporary evolutionary exploration only when additional behavioral data is needed. By controlling what enters replay and how much can be injected at each event, TEDI preserves the exploratory benefit of population search while making both interaction cost and replay influence event-driven.

Our contributions are:

- We formulate evolutionary assistance as a three-part replay intervention that jointly controls activation, experience admission, and event-level replay influence, extending conditional evolution beyond activation timing alone.
- We develop a temporary population-based exploration and validation pipeline in which candidate trajectories remain outside replay until screening, and each intervention is limited through balanced, bounded replay insertion.

- We evaluate TEDI against six baselines on four MuJoCo continuous-control tasks. TEDI achieves the highest mean peak return on all four tasks and the lowest average wall-clock training time among the compared evolutionary RL methods.

2 Related Work

Population-assisted reinforcement learning. Evolutionary reinforcement learning combines population-based policy search with gradient-based RL to improve exploration and policy optimization. The original ERL framework couples an evolutionary population with an off-policy learner through shared experience and periodic policy transfer (Khadka & Tumer, 2018). CEM-RL integrates cross-entropy search with actor updates (Pourchot & Sigaud, 2019), while PDERL introduces behavior-preserving variation operators to improve evolutionary search in deep policies (Bodnar et al., 2020). More recent methods strengthen the interaction between evolutionary and gradient-based learning. ERL-Re^2 improves efficiency through shared state representations with individual policy heads (Hao et al., 2023), and EvoRainbow combines multiple improvements to evolutionary policy search into a stronger ERL framework (Li et al., 2024). Together, these methods show that population search can complement gradient learning by generating behavioral alternatives that are difficult to obtain through local policy updates alone.

Most ERL methods, however, maintain evolutionary assistance as a persistent component of training. E-ERL moves toward conditional evolution by invoking population evaluation only when the reinforcement learner has difficulty improving (Wu et al., 2022). This shows that evolutionary computation need not be scheduled uniformly throughout training, but it primarily addresses *when* additional evolutionary evaluation should occur. TEDI builds on this insight while targeting a broader replay-intervention problem. After evolutionary exploration is activated, TEDI also controls *which* generated experience is allowed to enter replay and *how much* influence one intervention can exert. Thus, TEDI is not another persistent population optimizer, but a temporary mechanism for acquiring selectively admitted replay data.

Structured exploration and temporary intervention. Many RL methods explore beyond independent action noise. Parameter-space perturbations produce temporally coherent changes in policy behavior and can therefore support exploration over an entire rollout (Plappert et al., 2018). Other approaches encourage exploration through intrinsic rewards (Pathak et al., 2017), randomized value functions or policy heads (Osband et al., 2016), or interventions that reset or modify the learner when training dynamics become unfavorable (Nikishin et al., 2022; D'Oro et al., 2023). These methods differ in whether they alter the policy, reward signal, value estimator, or optimization process. TEDI instead keeps the underlying off-policy actor–critic objective unchanged. It uses temporary parameter-perturbed policies only to collect candidate experience, discards them after each intervention, and lets evolutionary exploration affect learning only through the data that pass the admission mechanism.

Replay distribution and experience admission. Experience replay determines the empirical data distribution used for off-policy value estimation and policy improvement. Prioritized experience replay changes the sampling frequency of transitions already stored in the buffer (Schaul et al., 2016), while off-policy correction and replay-management methods address how previously collected data should be reused under distribution mismatch (Munos et al., 2016; Novati & Koumoutsakos, 2019). In population-assisted RL, the data-management problem arises one step earlier: trajectories generated by heterogeneous population policies may differ substantially from the learner's current data distribution. Zheng & Cheng (2023) show that population-generated experience can increase off-policy error and propose separating population and learner experience.

TEDI acts before population-generated experience is inserted into the main replay buffer. Candidate trajectories are first retained outside replay, screened at both transition and trajectory levels, and admitted only under an explicit event-level budget. This differs from methods that reweight, resample, or correct experience after insertion. TEDI jointly controls when additional data are acquired, which generated transitions and trajectories are accepted, and how much replay mass a single intervention can contribute. This joint control of activation, admission, and influence is the central distinction of TEDI from prior population-assisted and replay-management approaches.

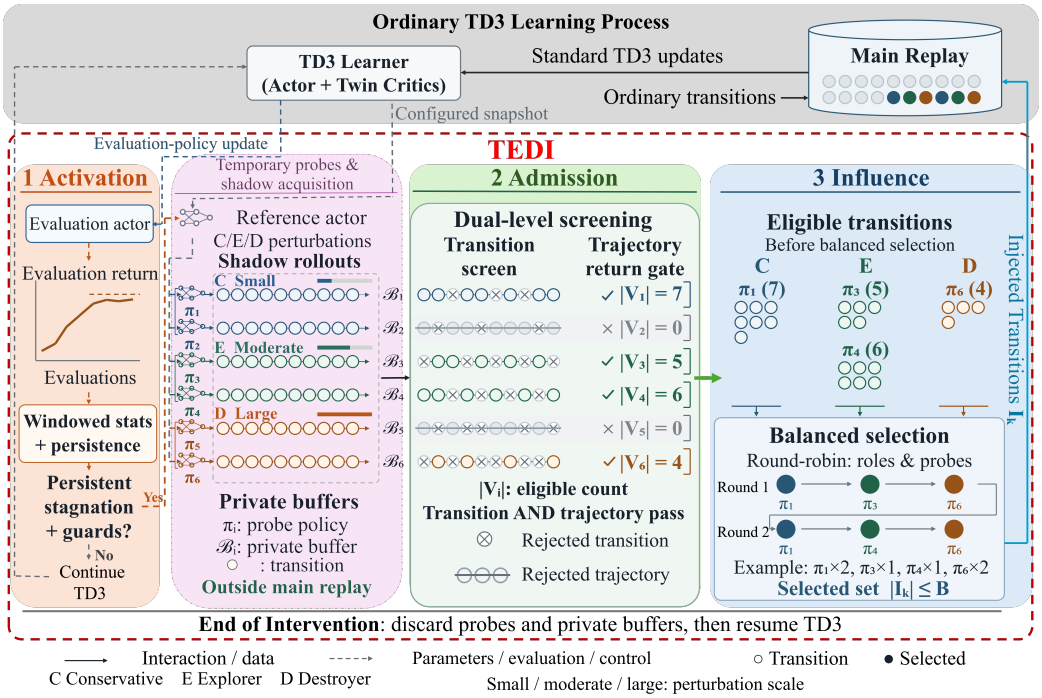


Figure 2: Overview of TEDI. Persistent stagnation activates a temporary population of parameter-perturbed policy probes. Candidate experience is collected outside the main replay buffer, screened at transition and trajectory levels, and inserted through balanced and bounded replay injection. The temporary population is discarded after the intervention.

3 Triggered Evolutionary Data Injection

3.1 Problem Setting and Overview

We consider a discounted Markov decision process $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ and instantiate TEDI with TD3 (Fujimoto et al., 2018) as the underlying off-policy learner. Figure 2 shows the resulting workflow. Along the top, ordinary main-environment transitions enter the replay buffer $\mathcal{D}$ and support standard updates to the TD3 actor and twin critics. The three numbered regions below show how TEDI activates a temporary intervention, admits candidate experience, and controls its influence on replay.

First, *activation* checks returns from the evaluation actor for persistent stagnation, subject to trigger guards. If the trigger is not met, ordinary TD3 training continues. Once triggered, the unnumbered temporary-probe stage snapshots the configured reference actor and generates policy probes π_i using conservative (C), explorer (E), and destroyer (D) perturbations. Each probe collects a shadow rollout in its private buffer $\mathcal{B}_i$, outside the main replay buffer. Second, *admission* applies the transition screen to candidate transitions and the trajectory return gate to their generating probes. The transitions passing both checks form the eligible set V_i for probe π_i. The figure shows $|V_i|$, the size of this set. Third, *influence* selects eligible transitions across probes and roles in a balanced order, forming the injection set $\mathcal{I}_k$ with at most B transitions. The selected transitions enter $\mathcal{D}$. The temporary probes and private buffers are then discarded, and ordinary TD3 training resumes.

3.2 Activation: Persistent Stagnation Detection

Activation determines whether temporary evolutionary exploration should be requested. Let J_j denote the mean return from a deterministic evaluation of the configured actor at evaluation opportunity j. Depending on the task, this is the current TD3 actor or its exponential-moving-average (EMA) copy. When used, the EMA actor tracks the learner actor during ordinary training but does not replace it. Evaluation rollouts do not enter $\mathcal{D}$. The reference actor used for probes is configured

separately. To avoid reacting to an isolated fluctuation, TEDI compares two adjacent windows of width w:

$$\mu_j^- = \frac{1}{w} \sum_{i=j-2w+1}^{j-w} J_i, \qquad \mu_j^+ = \frac{1}{w} \sum_{i=j-w+1}^{j} J_i. \tag{1}$$

The window means capture recent progress. For the normalized-window configuration, progress and recent variability are represented as

$$g_j = \frac{\mu_j^+ - \mu_j^-}{|\mu_j^-| + \epsilon}, \qquad v_j = \frac{\mathrm{Std}(J_{j-w+1:j})}{|\mu_j^+| + \epsilon}, \tag{2}$$

where $\epsilon > 0$ ensures numerical stability. The raw stagnation decision is written generally as

$$s_j = \mathcal{D}_{\mathrm{stag}}\left(\mu_j^-, \mu_j^+, g_j, v_j, \eta_{\mathrm{stag}}\right) \in \{0, 1\}, \tag{3}$$

where $\mathcal{D}_{\mathrm{stag}}$ denotes the configured window-based predicate and η_{stag} its thresholds. The legacy configuration uses the window means to determine raw stagnation, while the normalized-window configuration uses g_j and v_j. Depending on the configuration, an additional variability requirement can be imposed through the trigger guards. The detector remains inactive until two complete evaluation windows are available.

A raw signal alone does not activate an intervention. Let c_j count consecutive raw stagnation signals since the preceding intervention. It increases when $s_j = 1$ and resets to zero when $s_j = 0$. Let p denote the required persistence length and let $G_j \in \{0, 1\}$ be the conjunction of the enabled trigger guards, including any cooldown or stopping conditions. The event indicator is

$$z_j = G_j \, \mathbb{I}[c_j \geq p]. \tag{4}$$

After a triggered event, the counter is reset before the next evaluation. Persistence reduces reactions to transient evaluation noise, while the guards prevent interventions outside the admissible training regime. When enabled, the cooldown guard enforces a minimum interval between activations. Appendix B.4 derives the resulting upper bound on triggered events over a finite training horizon.

The trigger requests additional exploration. It does not establish that the learner has reached a local optimum or guarantee that an event will insert data into replay. Environment-specific detector types, persistence lengths, and trigger guards are summarized in Appendix C.4.

3.3 Temporary Policy Probes and Shadow Acquisition

Once an intervention is activated, TEDI constructs a temporary source of alternative experience. Let k index triggered events. At event k, the method snapshots reference actor parameters $\bar{\theta}$ and generates m temporary policy probes. Omitting the event index for clarity,

$$\theta_i = \bar{\theta} + \sigma_{r_i} \left(M_i \odot \epsilon_i\right), \qquad \epsilon_i \sim \mathcal{N}(0, I), \qquad i = 1, \ldots, m. \tag{5}$$

Here r_i denotes the perturbation role of probe i and σ_{r_i} controls its perturbation scale. The binary mask M_i selects parameters eligible for perturbation and is all ones when masking is disabled. Different roles therefore probe the neighborhood of the reference actor at different scales.

The resulting policy $\pi_i \equiv \pi_{\theta_i}$ is not a candidate replacement for the learner actor. Instead, it serves only as a temporary data-acquisition policy. Each probe collects a shadow rollout of at most H_s steps and stores the resulting transitions in a private buffer $\mathcal{B}_i$. When required by the trajectory-level admission rule, the reference policy is evaluated under a matched shadow initialization. The reference-actor choice, probe count, perturbation scales and masking fractions, and shadow horizon are specified in Appendix C.4.

Candidate transitions remain isolated from $\mathcal{D}$ throughout shadow acquisition. Probes are not retained across events, and their parameters do not directly update the learner. Consequently, evolutionary exploration influences subsequent policy optimization only through experience that is explicitly admitted to the main replay buffer.

3.4 Admission: Transition- and Trajectory-Level Screening

Admission determines which probe-generated transitions are eligible for replay insertion. Rather than inserting every candidate, TEDI evaluates experience at two complementary levels. Transition-level screening evaluates individual samples, while trajectory-level screening evaluates the generating probe's rollout.

Transition-level screening. For a candidate transition $x_{i,t} = (s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1}, d_{i,t})$, let $d_{i,t}$ denote true environment termination, with time-limit truncation treated as nonterminal for bootstrapping. TEDI first computes a critic-based quality estimate. Depending on the configured scoring rule,

$$\widehat{q}_{i,t} = \begin{cases} \min\limits_{h\in\{1,2\}} Q_{\phi_h}(s_{i,t}, a_{i,t}), & \text{current-}Q, \\ r_{i,t} + \gamma(1-d_{i,t}) \min\limits_{h\in\{1,2\}} Q_{\phi_h^-}\left(s_{i,t+1}, \pi_{\bar{\theta}}(s_{i,t+1})\right), & \text{one-step target}, \end{cases} \tag{6}$$

where $Q_{\phi_h^-}$ is a TD3 target critic and $\bar{\theta}$ is the reference-actor snapshot used to construct the probes. The one-step value is used for screening and does not change the ordinary TD3 update target.

When replay-relative novelty adjustment is enabled, a minibatch of replay states $\mathcal{S}_D$ provides a reference for how closely the candidate state lies to existing replay support:

$$d_{i,t}^{\text{rep}} = \min_{s\in\mathcal{S}_D} \|s_{i,t} - s\|_2, \qquad q_{i,t} = \widehat{q}_{i,t} - \lambda_k[1 - d_{i,t}^{\text{rep}}]_+, \tag{7}$$

where $[x]_+ = \max(x, 0)$ and $\lambda_k \geq 0$ is the configured replay-proximity penalty weight at event k. When the adjustment is disabled, $\lambda_k = 0$ and $q_{i,t} = \widehat{q}_{i,t}$.

Let $\bar{r}_t$ denote the mean immediate reward across active probes and let m_t^q denote the median adjusted quality score. The transition-level decision is written generally as

$$A_{i,t}^{\text{trans}} = F_{r_i}\left(r_{i,t}, q_{i,t}, \bar{r}_t, m_t^q, d_{i,t}\right) \in \{0, 1\}, \tag{8}$$

where F_{r_i} is the role-dependent screening predicate. This form allows admission to combine absolute transition quality with population-relative information while keeping the concrete predicate separate from the overall intervention framework.

Trajectory-level screening. Transition-level quality alone does not guarantee that the generating behavior is suitable for replay. Let

$$R_i = \sum_{t=0}^{h_i-1} r_{i,t} \tag{9}$$

denote the return of probe i over its shadow rollout. For perturbation role r, a role-dependent quantile level q_r defines

$$\tau_r = \text{Quantile}\left(\{R_j\}_{j=1}^m, q_r\right). \tag{10}$$

Probe i passes the population-relative return gate when

$$A_i^{\text{pop}} = \mathbb{I}[R_i \geq \tau_{r_i}]. \tag{11}$$

When a matched reference comparison is enabled, the probe additionally satisfies

$$A_i^{\text{parent}} = \mathbb{I}[R_i \geq R_p - \delta_{r_i}], \tag{12}$$

where R_p is the matched reference return and δ_{r_i} is the role-dependent tolerance. Otherwise, $A_i^{\text{parent}} = 1$. The trajectory-level decision is therefore

$$A_i^{\text{beh}} = A_i^{\text{pop}} A_i^{\text{parent}}. \tag{13}$$

If $q_r = 0$ and the parent comparison is disabled, this gate is nonrestrictive for role r.

The eligible transitions contributed by probe i are

$$V_i = \left\{x_{i,t} \in \mathcal{B}_i : A_{i,t}^{\text{trans}} A_i^{\text{beh}} = 1\right\}. \tag{14}$$

Hence, each eligible set V_i contains only transitions that pass both checks. Some or all candidate experience may be rejected. The admission criteria are not relaxed to force an event to contribute data. The instantiated transition predicates and trajectory-gate settings are provided in Appendix B.1.

3.5 Influence: Balanced and Bounded Replay Injection

Admission produces an eligible set V_i for each probe π_i, and some of these sets may be empty. Influence control determines which eligible transitions enter the main replay buffer and limits the contribution of one intervention. TEDI shuffles eligible transitions within each probe, then cycles across nonempty perturbation roles and nonempty probes within each role. Selection ends when the event budget B is reached or no eligible transitions remain:

$$\mathcal{I}_k = \text{BalancedSelect}\left(\{V_i\}_{i=1}^m, B\right), \qquad |\mathcal{I}_k| \leq B. \tag{15}$$

The cycling rule gives eligible roles and probes repeated opportunities to contribute, while B limits the number of transitions admitted in a single event. The budget is an upper bound rather than a target, so the admission criteria are not relaxed to fill it. The selection procedure is detailed in Appendix B.2.

Algorithm 1 Triggered Evolutionary Data Injection

Initialize the learner, replay buffer $\mathcal{D}$, reference state, evaluation history, and trigger state with $j \leftarrow 0$, $c_0 \leftarrow 0$, and $k \leftarrow 0$.
while training is not finished **do**
  Collect a main-environment transition, store it in $\mathcal{D}$, and perform the ordinary off-policy updates.
  Update the configured reference-policy state if required.
  if an evaluation opportunity is reached **then**
    $j \leftarrow j + 1$. Evaluate the configured actor and append mean return J_j to the history.
    Set $s_j \leftarrow 0$. If $j \geq 2w$, compute the window statistics and raw signal s_j using Eqs. (1)–(3).
    Update the persistence counter c_j and obtain z_j using Eq. (4).
    if $z_j = 1$ **then**
      $k \leftarrow k + 1$ and snapshot the reference actor $\bar{\theta}$.
      Construct m temporary probes using Eq. (5).
      Collect shadow rollouts into private buffers $\{\mathcal{B}_i\}_{i=1}^{m}$.
      Collect a matched reference rollout if required.
      Apply transition- and trajectory-level screening and construct $\{V_i\}_{i=1}^{m}$ using Eq. (14).
      $\mathcal{I}_k \leftarrow \mathrm{BalancedSelect}(\{V_i\}_{i=1}^{m}, B)$.
      Insert $\mathcal{I}_k$ into $\mathcal{D}$ ▷ $\mathcal{I}_k$ may be empty
      Discard all temporary probes and private buffers.
      Reset the persistence counter and update cooldown and trigger states.
    end if
  end if
end while

Algorithm 1 places the intervention within the ordinary TD3 training loop. Each main-environment step updates the learner and any configured reference state. Evaluation updates the return history and trigger counter, but shadow rollouts begin only when $z_j = 1$. The algorithm then snapshots the reference actor, collects probe and optional matched reference rollouts outside $\mathcal{D}$, and applies both admission checks before balanced selection. Only the selected set $\mathcal{I}_k$ enters replay. This set may be empty, in which case the event inserts nothing. After the event, the algorithm discards all temporary state, resets the persistence counter, updates cooldown, and resumes ordinary TEDI training. This order prevents rejected shadow data from entering replay and prevents probes from persisting between events.

4 Experiments

4.1 Experimental Setup

We evaluate TEDI on four MuJoCo continuous-control tasks: Ant-v2, HalfCheetah-v2, Hopper-v2, and Swimmer-v2 (Todorov et al., 2012). We compare against six baselines: TD3, ERL, CEM-RL, PDERL(TD3), ERL-Re^2(TD3), and EvoRainbow. PDERL and ERL-Re^2 use TD3 as their gradient-based learner so that their actor–critic backbone is aligned with TEDI. Appendix C.2 summarizes the configuration origins and comparison settings for the baselines.

All main comparisons and ablations use five training seeds and 10^6 main-environment steps per run. For TEDI, probe rollouts and any additional reference-policy shadow rollouts are counted separately as auxiliary interactions and are not subtracted from the main-environment training horizon. Appendix B.3 provides the corresponding interaction accounting.

Learning curves show the across-seed mean, with smoothing applied only for visualization. For each seed, peak return is the maximum evaluation return obtained from the unsmoothed evaluation record, and we report the mean across seeds. Benchmark and hardware settings are given in Appendix C.1. The evaluation and reporting protocol is described in Appendix C.3, and learner and intervention settings are provided in Appendix C.4. The anonymized implementation is available at `https://anonymous.4open.science/r/rl-review-artifact-27a9c`.

4.2 Main Comparison

Figure 3 compares TEDI with six baselines. TEDI attains the highest mean peak return on all four tasks (Appendix A.1). Relative to the strongest baseline on each task, the gains are 4.2% on Ant,

9.8% on HalfCheetah, 5.6% on Hopper, and 1.0% on Swimmer. The curves provide context for these peak values. On Ant and HalfCheetah, TEDI keeps improving late in training. On Hopper, it progresses more slowly early on but reaches a higher-return region later. On Swimmer, several evolutionary RL baselines approach the same late-stage return region.

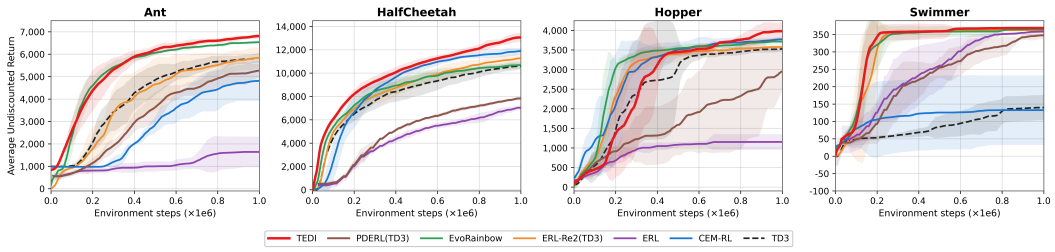


Figure 3: Learning curves of TEDI and six baselines on four MuJoCo continuous-control tasks. Curves show average undiscounted return over training; shaded regions indicate across-seed variability.

Table 1: Wall-clock training time of the compared methods across four MuJoCo continuous-control tasks.

Task	Wall-clock training time (s)						
	CEM-RL	ERL	PDERL(TD3)	ERL-Re^2(TD3)	EvoRainbow	TD3	TEDI
Ant-v2	6791.83	4560.40	5494.72	15423.35	11526.10	3035.50	3326.50
HalfCheetah-v2	6436.06	5302.90	4772.46	8387.25	10031.89	2178.87	4590.07
Hopper-v2	6484.24	4066.47	4574.89	15381.41	10742.29	2201.62	3026.78
Swimmer-v2	6361.24	3986.97	4724.99	15815.50	9539.78	2130.09	2706.30

Training time. Table 1 shows that TEDI has the lowest measured wall-clock training time among the compared evolutionary RL methods on all four tasks. Relative to the fastest evolutionary baseline on each task, its runtime is 3.8% lower on HalfCheetah and 25.6–32.1% lower on the other three tasks. TD3 remains faster overall. These timings reflect the reported implementations and execution platform. The realized events and shadow interactions in Appendix A.3 show how often temporary exploration was invoked, while Appendix B.3 accounts for its additional environment interactions.

4.3 Ablation Study

We evaluate three variants of TEDI. *TEDI w/o Trigger* removes stagnation-based activation, *TEDI w/o Filtering* removes transition-level admission filtering, and *TEDI Unlimited Injection* removes the event-level bound on replay insertion. Full TEDI retains all three controls.

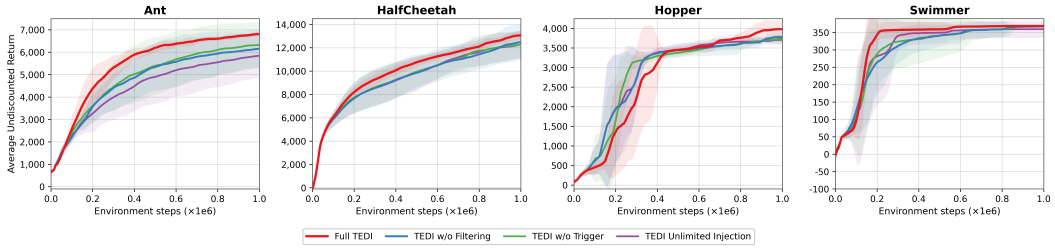


Figure 4: Learning curves of Full TEDI and its three ablations on four MuJoCo continuous-control tasks.

Figure 4 compares the four configurations. Removing any one of the three controls yields a lower mean peak return than Full TEDI on every task (Appendix A.2). Against the strongest ablated variant, the advantage is 4.5–7.6% on Ant, HalfCheetah, and Hopper and 0.24% on Swimmer. The three controls therefore make complementary contributions to the reported peak returns: activation limits when probe data are generated, admission selects which transitions survive, and the injection bound limits how much they affect replay.

Peak return captures the best attained evaluation, while the area under the learning curve (AUC) summarizes returns across the full training horizon. Figure 5 shows that Full TEDI also has the largest mean AUC on Ant, HalfCheetah, and Swimmer. Hopper has the opposite AUC ranking: Full TEDI reaches the highest peak, but its normalized AUC is 2,249 versus 2,281–2,499 for the ablations. Figure 4 explains the difference. Several variants improve faster early on Hopper, whereas Full TEDI reaches higher returns later. Its late gain does not fully offset the earlier shortfall in an average over the training horizon. Thus, the peak-return advantage on Hopper should not be interpreted as faster learning throughout training.



Figure 5: Mean AUC of the evaluation-return curves over five seeds, divided by 10^6. Error bars show 95% Student-t interval half-widths. Full denotes the complete method; w/o Filt., w/o Trig., and Unlim. remove filtering, triggering, and the injection bound, respectively.

5 Conclusion

We introduced TEDI, which reformulates evolutionary assistance as a controlled replay-data intervention rather than a continuously active policy-optimization process. TEDI jointly controls when temporary evolutionary exploration is invoked, which generated experience is admitted to replay, and how much influence each intervention can exert on subsequent learning. Across four MuJoCo continuous-control tasks, TEDI achieves the highest mean peak return in comparisons with six baselines and the lowest average wall-clock training time among the compared evolutionary RL methods. Ablation results show that stagnation-based activation, experience filtering, and bounded injection play complementary roles. These findings suggest that the effectiveness of evolutionary assistance depends not only on generating diverse experience, but also on controlling when and how that experience enters replay. This work instantiates the replay-intervention view with a TD3 backbone and task-specific intervention settings. Extending this framework to broader off-policy learners, more adaptive admission mechanisms, and more diverse domains is an important direction for future work.

AI USE STATEMENT

Generative AI tools were used during manuscript preparation for language polishing, formatting and consistency checks, and syntax checks of code. The authors reviewed and revised the AI-assisted content and take full responsibility for the scientific claims, experimental results, references, and final manuscript.

ETHICS STATEMENT

This work evaluates TEDI on standard MuJoCo simulation benchmarks. The reported experiments involve no human participants, personal data, or deployment on physical systems. TEDI's replay admission and injection controls do not provide formal safety guarantees for the learned policies. Any application to physical or safety-critical systems would require separate safety assessment and appropriate safeguards.

REPRODUCIBILITY STATEMENT

Section 3 describes the TEDI procedure. Appendix B provides implementation-level admission rules and auxiliary-interaction accounting. Appendix C reports the benchmark environments, execution platform, baseline configurations, evaluation protocol, and learner and intervention settings. All main comparisons and ablation studies use five different training seeds. Appendix A provides the numerical comparisons, ablation results, and recorded intervention and shadow-interaction counts. The anonymized implementation is linked in Section 4.1.

REFERENCES

Cristian Bodnar, Ben Day, and Pietro Liò. Proximal distilled evolutionary reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3283–3290, 2020.

Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G. Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *International Conference on Learning Representations*, 2023.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 3061–3071, 2020.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1587–1596, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2052–2062, 2019.

Jianye Hao, Pengyi Li, Hongyao Tang, Yan Zheng, Xian Fu, and Zhaopeng Meng. ERL-Re^2: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation. In *International Conference on Learning Representations*, 2023.

Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32, pp. 11761–11771, 2019.

Pengyi Li, Yan Zheng, Hongyao Tang, Xian Fu, and Jianye Hao. EvoRainbow: Combining improvements in evolutionary reinforcement learning for policy search. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 29427–29447, 2024.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 16828–16847, 2022.

Guido Novati and Petros Koumoutsakos. Remember and forget for experience replay. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 4851–4860, 2019.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2778–2787, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

Aloïs Pourchot and Olivier Sigaud. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *International Conference on Learning Representations*, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

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

Xiaoqiang Wu, Qingling Zhu, Qiuzhen Lin, Jianqiang Li, Jianyong Chen, and Zhong Ming. An efficient evaluation mechanism for evolutionary reinforcement learning. In *Intelligent Computing Theories and Application*, volume 13393 of *Lecture Notes in Computer Science*, pp. 41–50. Springer, 2022.

Bo Zheng and Ran Cheng. Rethinking population-assisted off-policy reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 624–632, 2023.

A Additional Experimental Results

A.1 Peak-return comparison

Table 2 provides the numerical peak-return summary corresponding to the main learning curves in Figure 3.

Table 2: Peak evaluation return over five different seeds. Entries report mean ± 95% Student-t interval half-width.

Method	Peak evaluation return			
	Ant-v2	HalfCheetah-v2	Hopper-v2	Swimmer-v2
TD3	5861.53 ± 222.64	10637.14 ± 264.77	3530.67 ± 69.43	140.10 ± 36.27
CEM-RL	4810.40 ± 891.11	11907.69 ± 245.06	3773.60 ± 66.17	132.54 ± 99.95
ERL	1640.22 ± 682.93	7030.60 ± 207.51	1151.43 ± 199.25	358.37 ± 3.54
ERL-Re^2(TD3)	5846.79 ± 423.72	11288.35 ± 409.59	3576.39 ± 52.26	362.99 ± 1.25
EvoRainbow	6542.82 ± 101.72	10702.54 ± 317.71	3715.24 ± 77.98	364.89 ± 1.33
PDERL(TD3)	5320.76 ± 285.92	7825.97 ± 146.47	3026.01 ± 483.12	347.62 ± 10.73
TEDI	$\mathbf{6815.74} \pm 94.93$	$\mathbf{13079.09} \pm 379.26$	$\mathbf{3984.43} \pm 224.01$	$\mathbf{368.70} \pm 1.77$

TEDI achieves the highest mean peak return on all four tasks. The improvement over the strongest baseline is most pronounced on HalfCheetah and remains clear on Ant and Hopper. On Swimmer, several population-based methods reach similar high-return regions, resulting in a smaller margin, while TEDI still obtains the highest mean peak return. These numerical results are consistent with the learning trajectories in Figure 3.

A.2 Ablation peak returns

Table 3 reports the peak-return summary for the four ablation variants shown in Figure 4.

Table 3: Peak-return ablation results over five different seeds. Entries are mean ± 95% Student-t interval half-width. Bold denotes the highest mean within each task.

Task	Variant	Peak return
Ant-v2	TEDI w/o Filtering	6172.14 ± 862.62
	TEDI w/o Trigger	6331.59 ± 1026.40
	TEDI Unlimited Injection	5865.51 ± 1043.57
	Full TEDI	$\mathbf{6815.74 \pm 94.93}$
HalfCheetah-v2	TEDI w/o Filtering	12516.24 ± 1488.21
	TEDI w/o Trigger	12284.69 ± 1197.50
	TEDI Unlimited Injection	12349.75 ± 1250.95
	Full TEDI	$\mathbf{13079.09 \pm 379.26}$
Hopper-v2	TEDI w/o Filtering	3779.72 ± 140.28
	TEDI w/o Trigger	3732.11 ± 113.57
	TEDI Unlimited Injection	3706.19 ± 128.98
	Full TEDI	$\mathbf{3984.43 \pm 224.01}$
Swimmer-v2	TEDI w/o Filtering	367.40 ± 2.79
	TEDI w/o Trigger	367.83 ± 2.83
	TEDI Unlimited Injection	359.55 ± 24.82
	Full TEDI	$\mathbf{368.70 \pm 1.77}$

Full TEDI attains the highest mean peak return on all four tasks. On Ant, HalfCheetah, and Hopper, it exceeds the strongest ablated variant by approximately 7.6%, 4.5%, and 5.4%, respectively. The differences are much smaller on Swimmer, where the full method and the trigger- and filtering-ablated variants reach similar peak-return levels. Overall, no single ablated variant is consistently

strongest across tasks, indicating that activation, experience filtering, and bounded injection affect the learning process in complementary ways.

A.3 Realized evolutionary intervention activity

The event-driven design produces different amounts of evolutionary assistance across tasks. Table 4 summarizes the realized intervention activity over the same five TEDI runs used for the final main comparison. The table reports the total number of triggered events together with their per-run range and the corresponding shadow-rollout interactions. Total shadow steps include both probe rollouts and any additional matched reference-policy rollouts. The parent-shadow column reports the reference-policy portion separately. These counters are distinct from the wall-clock measurements in Table 1.

Table 4: Recorded TEDI intervention activity over five runs per task. The event column reports the total number of triggered interventions across the five runs together with the per-run range. Shadow-step columns report cumulative interaction counts in thousands. The parent-shadow count is included in the total shadow count. "Zero-event runs" gives the number of runs in which no evolutionary intervention was activated.

Task	Events [range/run]	Shadow steps ($\times 10^3$)	Parent subset ($\times 10^3$)	Zero-event runs
Ant-v2	54 [5, 19]	125.96	0.00	0/5
HalfCheetah-v2	43 [6, 10]	301.00	43.00	0/5
Hopper-v2	5 [0, 3]	10.91	1.54	3/5
Swimmer-v2	92 [15, 22]	552.00	0.00	0/5

The realized activity varies substantially across tasks. Across the five runs, TEDI triggers 54 interventions on Ant and 43 on HalfCheetah. Hopper requires much less evolutionary assistance, with only five interventions in total and no intervention in three of the five runs. Swimmer triggers 92 interventions across the five runs and therefore has the largest shadow-interaction volume among the four tasks.

These results show that TEDI does not impose a fixed intervention frequency. Instead, the amount of evolutionary assistance depends on the observed learning dynamics and the configured trigger conditions. In particular, the zero-event Hopper runs show that ordinary off-policy learning can proceed without evolutionary intervention when the trigger condition is not satisfied. These measurements complement the analytical accounting in Appendix B.3 and the empirical runtime comparison in the main text.

B Implementation Details of TEDI

B.1 Exact admission predicates

Let $\bar{r}_t$ be the mean immediate reward across active temporary policies at shadow step t, and let m_t^q be the median adjusted transition-quality score. Table 5 gives the transition predicates used in the reported implementation.

Table 5: Environment-specific transition admission predicates. Here r is immediate reward, q is adjusted transition quality, $\bar{r}_t$ is the active-population reward mean, m_t^q is the active-population quality median, and e denotes true environment termination.

Task	Conservative	Explorer	Destroyer
Ant-v2	$q > m_t^q \wedge r \geq \bar{r}_t$	$q > 0.9m_t^q \wedge r > 0$; 0.5 retention	$r > 2.5\bar{r}_t \vee r > 5$
HalfCheetah-v2	$r \geq \bar{r}_t \wedge q \geq m_t^q$	$\neg e \wedge r \geq 0.75\bar{r}_t \wedge q \geq 0.90m_t^q$	$\neg e \wedge r \geq 0.60\bar{r}_t \wedge q \geq 0.80m_t^q$
Hopper-v2	$r \geq \bar{r}_t \wedge q > m_t^q$	$r > 0.8\bar{r}_t \wedge q > 0.8m_t^q$	$r > 0.9\bar{r}_t \wedge \neg e$
Swimmer-v2	$r \geq \bar{r}_t$	$r > 0.8\bar{r}_t \wedge q > 0.5m_t^q$	$r > 1.5\bar{r}_t \wedge \neg e$

The trajectory-level gate uses quantiles of the collected shadow-rollout returns. Ant uses quantile 0, HalfCheetah uses 0.67, and Hopper and Swimmer use 0.5. HalfCheetah additionally uses parent-return tolerances of 0, 50, and 100 for C/E/D, respectively; Hopper uses 0, 100, and 250. Ant and Swimmer do not use the parent-return gate. Time-limit truncation is excluded from the true-termination indicator used by the gate.

B.2 Balanced selection

Eligible transitions are shuffled within each probe and grouped by perturbation role. Selection cycles across nonempty roles and then across nonempty probe buffers within each role until the event budget B is reached or no eligible transition remains. The procedure does not relax admission thresholds to fill unused capacity, and an intervention may therefore insert an empty set.

B.3 Auxiliary interaction accounting and computational work

The replay-injection budget B limits how many accepted transitions enter $\mathcal{D}$, but it does not bound the environment interactions spent generating candidate experience. Let L denote the number of intervention opportunities, $z_j \in \{0,1\}$ the trigger indicator at opportunity j, and $K = \sum_{j=1}^{L} z_j$ the realized number of triggered events. For event k, let $h_{k,i} \le H_s$ be the realized shadow-rollout length of probe i, and let $h_{k,p} \le H_s$ denote an additional matched reference rollout when one is collected. The auxiliary interaction count is

$$N_{\mathrm{aux}} = \sum_{k=1}^{K} \left(h_{k,p} + \sum_{i=1}^{m} h_{k,i} \right) \le K(m+1)H_s. \tag{16}$$

Every newly generated probe or reference interaction is charged once, including rollouts whose transitions are later rejected and events with an empty injection set. The reference term is zero when no additional matched rollout is required. This accounting is separate from the 10^6 main-environment-step training horizon used in the experiments.

For fixed full-length probe rollouts,

$$N_{\mathrm{probe}} = KmH_s, \qquad N_{\mathrm{aux}} = K(m+1)H_s \tag{17}$$

when one additional full-length reference rollout is collected per event. A persistent population evaluated at all L opportunities, with population size n and rollout horizon H, generates $N_{\mathrm{pop}} = LnH$ population interactions. Defining the empirical intervention rate $\rho = K/L$, the corresponding restricted rollout ratios are

$$\frac{N_{\mathrm{probe}}}{N_{\mathrm{pop}}} = \rho \frac{mH_s}{nH}, \qquad \frac{N_{\mathrm{aux}}}{N_{\mathrm{pop}}} = \rho \frac{(m+1)H_s}{nH}. \tag{18}$$

Thus, when the probe and persistent-population rollout terms are matched, TEDI's evolutionary sampling cost scales with the realized intervention rate rather than with every scheduled population round. Any unmatched reference or evaluation overhead must be accounted for separately.

A simple work model also includes validation, perturbation, and selection. Let C_{roll} denote work per rollout transition, C_{val} validation work per probe transition, $C_{\mathrm{pert}}(P)$ perturbation work for a policy with P parameters, and $C_{\mathrm{sel},k}$ the selection work at event k. Then

$$\begin{aligned} W_{\mathrm{aux}} = \Theta\Bigg(& N_{\mathrm{aux}} C_{\mathrm{roll}} + \sum_{k=1}^{K} \sum_{i=1}^{m} h_{k,i} C_{\mathrm{val}} \\ & + KmC_{\mathrm{pert}}(P) + \sum_{k=1}^{K} C_{\mathrm{sel},k} \Bigg). \end{aligned} \tag{19}$$

This expression excludes the shared TD3 updates and implementation-specific synchronization, environment setup, and evaluation overhead. It therefore characterizes auxiliary work rather than measured wall-clock time; the latter is reported empirically in Table 1.

B.4 COOLDOWN-BASED EVENT BOUND

Suppose d intervention opportunities are blocked between two activations. Triggered-event indices are then separated by at least $d+1$. With L opportunities,

$$K \le \left\lceil \frac{L}{d+1} \right\rceil, \qquad N_{\text{aux}} \le \left\lceil \frac{L}{d+1} \right\rceil (m+1)H_s. \tag{20}$$

The bound follows by placing the first event at the earliest admissible opportunity and every later event at the earliest opportunity allowed by the cooldown. A cooldown specified in environment steps must first be converted to the corresponding number of evaluation opportunities. The bound describes the maximum event-driven auxiliary cost under the stated assumptions and does not by itself imply a wall-clock speedup.

C EXPERIMENTAL CONFIGURATION

C.1 PLATFORM AND BENCHMARK SETTINGS

The experiments use the MuJoCo continuous-control tasks Ant-v2, HalfCheetah-v2, Hopper-v2, and Swimmer-v2. The recorded execution environment uses Python 3.8.20, PyTorch with CUDA, and an NVIDIA RTX 3090-class GPU. Runs are executed as separate training jobs without parallel environment sampling. All methods use a 10^6-step main-environment training horizon. Table 6 summarizes the benchmark dimensions used in the experiments.

Table 6: MuJoCo task configuration used in the experiments.

Task	Observation dim.	Action dim.	Horizon
Ant-v2	111	8	1000
HalfCheetah-v2	17	6	1000
Hopper-v2	11	3	1000
Swimmer-v2	8	2	1000

C.2 COMPARED METHODS

The main comparison contains TD3, ERL, CEM-RL, PDERL(TD3), ERL-Re^2(TD3), EvoRainbow, and TEDI. TD3 serves as the non-evolutionary backbone baseline. PDERL and ERL-Re^2 are evaluated with a TD3 learner so that their gradient-based backbone is aligned with TEDI. The population-assisted baselines otherwise retain their respective evolutionary mechanisms. The configuration record used for the final comparison is summarized in Table 7.

Table 7: Configuration origins and comparison settings for the evaluated methods.

Method	Configuration origin	Comparison setting
TD3	Published TD3 implementation	Reference TD3 configuration
ERL	ERL implementation	MuJoCo task settings
CEM-RL	CEM-RL implementation	MuJoCo population and evaluation settings
PDERL(TD3)	PDERL mechanism	TD3 as gradient learner
ERL-Re^2(TD3)	ERL-Re^2 implementation	TD3 backbone and aligned task settings
EvoRainbow	Released implementation	Task-specific settings
TEDI	This work	Environment-specific settings in Table 9

C.3 EVALUATION AND REPORTING PROTOCOL

All main-comparison and ablation results use five different training seeds per method–task pair. Learning curves plot the across-seed mean. Any curve smoothing is applied only for visualization; peak-return and AUC summaries are computed from the underlying unsmoothed evaluation records. Peak return is the maximum evaluation return observed within the 10^6-step main-environment horizon.

For TEDI, periodic evaluation uses the current TD3 actor on Ant-v2 and the EMA actor on HalfCheetah-v2, Hopper-v2, and Swimmer-v2, with deterministic actions and exploration noise disabled. Ant and Swimmer are evaluated at episode boundaries, while HalfCheetah and Hopper use periodic evaluation every 5,000 main-environment steps. The recorded numbers of evaluation episodes are 1, 3, 5, and 1 for Ant, HalfCheetah, Hopper, and Swimmer, respectively. Temporary probe rollouts and any additional matched reference rollouts are kept separate from the 10^6 main-environment-step budget.

AUC is used only as a supplementary characterization of training dynamics. It is computed by trapezoidal integration of the unsmoothed evaluation-return curve against main-environment steps and normalized by 10^6. Wall-clock time is reported in seconds under the recorded implementation and execution platform. Because runtime depends on hardware and implementation details, it is interpreted as an empirical computational-cost measurement rather than a hardware-independent complexity result.

C.4 Learner and Intervention Settings

Table 8 lists the TD3 learner and replay settings for each task. Table 9 summarizes the environment-specific TEDI intervention settings.

Table 8: TD3 learner and replay settings used by TEDI. Values are environment-specific.

Parameter	Ant-v2	HalfCheetah-v2	Hopper-v2	Swimmer-v2
Actor hidden layers	256, 256 + LayerNorm + ReLU			
Critic hidden layers	Twin critic, 256, 256 + LayerNorm + ReLU			
Actor learning rate	3×10^{-4}	3×10^{-4}	1×10^{-4}	1×10^{-4}
Critic learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Discount factor γ	0.99	0.99	0.99	0.999
Target update rate τ	0.005	0.005	0.005	0.005
Minibatch size	256	256	256	256
Replay capacity	10^6	3.5×10^5	3×10^5	5×10^5
Policy update delay	2	2	2	2
Target smoothing noise	0.2	0.2	0.2	0.2
Target noise clipping	0.5	0.5	0.5	0.5
Exploration noise	$0.30 \rightarrow 0.10$	$0.15 \rightarrow 0.08$	$0.12 \rightarrow 0.02$	0.124 (fixed)
Actor gradient clipping	5.0	5.0	5.0	5.0
Critic gradient clipping	10.0	10.0	10.0	10.0
EMA decay β	0.99	0.99	0.995	0.99

Ant-v2 has the largest observation and action dimensions among the four tasks, making coordinated exploration particularly demanding. Its trajectory-level gate is therefore less restrictive: the return quantile is zero and the parent-return gate is disabled. This allows transitions from lower-return probes to remain candidates for transition-level screening (Appendix B.1), rather than discarding their entire rollouts. Ant also uses the current actor as the perturbation reference and imposes no cooldown, allowing exploration to recur when stagnation persists. The persistence requirement of 10 evaluations, the 500-step shadow horizon, and the per-event injection budget of 500 still limit activation frequency, shadow interactions, and replay insertion. These settings aim to provide sustained exploratory support for the more complex task while keeping admitted experience selective and bounded.

For Hopper, intervention stopping additionally uses an evaluation-return and survival guard. Exact transition-level admission predicates, trajectory-level gates, and role-specific rules are listed in Appendix B.1.

Table 9: Environment-specific TEDI settings. C, E, and D denote the conservative, explorer, and destroyer perturbation roles. Values reflect the actual run configuration.

Parameter	Ant-v2	HalfCheetah-v2	Hopper-v2	Swimmer-v2
Detector	legacy	normalized windows	legacy	legacy
Persistence	10	3	6	15
Minimum reward for evolution	-1200	5500	500	20
Minimum steps	0	180000	50000	50000
Cooldown	0	75,000	50,000	25,000
Stop reward	none	13750	3650	1000000
Population size	6	6	6	6
Shadow horizon	500	1000	1000	1000
Injection budget	500	200	80	150
Conservative σ	0.01	0.008	0.005	0.002
Explorer σ	0.05	0.022	0.015	0.01
Destroyer σ	0.25	0.030	0.04	0.03
Transition value	current Q	one-step target	current Q	current Q
Return quantile	0	0.67	0.5	0.5
Parent gate	off	on	on	off
Parent tolerance C/E/D	n/a	0 / 50 / 100	0 / 100 / 250	n/a
Mutation parent	actor	EMA actor	EMA actor	EMA actor
Mutation fraction	1.0	1.0	0.2	1.0