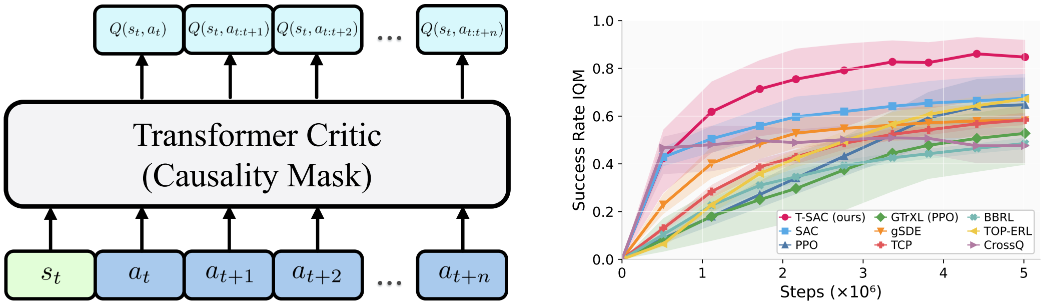
Chunking the Critic: A Transformer-based Soft Actor-Critic with N-Step Returns

Anonymous authors
Paper under double-blind review

Abstract

We introduce a sequence-conditioned critic for Soft Actor–Critic (SAC) that models trajectory context with a lightweight Transformer and trains on aggregated N-step targets. Unlike prior approaches that (i) score state–action pairs in isolation or (ii) rely on actor-side action chunking to handle long horizons, our method strengthens the critic itself by conditioning on short trajectory segments and integrating multi-step returns—without importance sampling (IS). The resulting sequence-aware value estimates capture the critical temporal structure for extended-horizon and sparse-reward problems. On local-motion benchmarks, we further show that freezing critic parameters for several steps makes our update compatible with CrossQ's core idea, enabling stable training *without* a target network. Despite its simplicity—a 2-layer Transformer with 128–256 hidden units and a maximum update-to-data ratio (UTD) of 1—the approach consistently outperforms standard SAC and strong off-policy baselines, with particularly large gains on long-trajectory control. These results highlight the value of sequence modeling and N-step bootstrapping on the critic side for long-horizon reinforcement learning.



(a) Transformer critic processes segments of actions rather than a single action, using causal self-attention so token i attends only to timesteps $\leq i$, preventing future-information leakage.

(b) Aggregated success on Meta–World ML1 (50 tasks) comparing T–SAC to step-based and episodic baselines. Curves show IQM success rate with 95% bootstrap confidence intervals; unless noted, results average 8 seeds and count success only at the final timestep.

Figure 1: T–SAC overview and aggregate Meta-World ML1 results.

1 Introduction

Off–policy actor–critic methods are the workhorses of continuous control. SAC (Haarnoja et al., 2018a) is notable for sample efficiency and stability, driven by tightly controlled bootstrap targets and mechanisms that curb overestimation. Recent work shows that similar stability can be achieved by slowing critic updates or regularizing its function class (Vincent et al., 2025; Piché et al., 2021; Gallici et al., 2024). In this vein, CrossQ (Bhatt et al., 2019) removes target networks by pairing Batch Renormalization (BRN) (Ioffe, 2017) with bounded activations, yielding strong locomotion results.

A complementary line of work uses *temporally extended actions* for temporal abstraction. Movement primitives and action chunking can aid exploration and produce smoother trajectories than purely step–based control (Otto et al., 2023a; Li et al., 2024b; Zhang et al., 2022). However, methods that jointly predict actions and repetition horizons (Sharma et al., 2017) or replace the policy with recurrent sequence models (Zhang et al., 2022) have not consistently produced robust gains.

Accurate value estimation remains difficult in sparse-reward, long-horizon, high-dimensional settings. Multi-step targets (N-step returns) (Sutton et al., 1998) can reduce bias at the cost of higher variance and typically require off-policy IS (Precup et al., 2000), which complicates—and can destabilize—training (Munos et al., 2016).

Transformers have become strong backbones for RL: self-attention supports sequence modeling, long-range credit assignment, and flexible conditioning on history (Vaswani et al., 2017; Parisotto et al., 2020; Agarwal et al., 2023). They parameterize policies and critics, serve as world models, and enable offline RL via trajectory/return conditioning (Chen et al., 2022; 2021; Janner et al., 2021).

This paper: Transformer-based Soft Actor-Critic (T-SAC). Our **primary contribution** is **T-SAC**, a *step-based* Soft Actor–Critic in which the standard MLP critic is replaced by a *sequence-conditioned Transformer critic* trained on *short trajectory segments* with *N-step returns*. By *chunking* temporal structure inside the critic—attending over brief state–action windows and aggregating multi-step TD targets—we improve long-horizon credit assignment while keeping the policy strictly one-step (no trajectory-level outputs or replanning) and the update rule free of IS.

Around this core design, we introduce several *supporting* choices that improve stability and practicality but are not the main conceptual novelty: (i) causal masking and a lightweight Transformer depth, following TOP-ERL in spirit but adapted to a step-based SAC setting; (ii) a simple gradient-averaged N-step loss; and (iii) a lightweight critic parameter-freezing schedule on locomotion benchmarks that enables stable training *without* target networks (in contrast to Polyak averaging as in Deep Q-Learning (Van Hasselt et al., 2016)). Empirically, T-SAC preserves SAC-style stability and is sample efficient, solving most Meta-World tasks in $\sim$5M interactions, achieving 96.8% success on Box–Pushing (dense), and remaining stable at low update-to-data (UTD) ratios (Meta-World $= 1$, Gymnasium MuJoCo $= 0.75$, Box–Pushing $= 0.25$), with 1M interactions at UTD$= 1$ completing in $\sim$3 hours.

2 Related Work

2.1 Transformer-based Critics for Episodic RL

Episodic RL (ERL) replaces per-step actions with trajectory-level primitives (Otto et al., 2023a; Li et al., 2024b), easing long-horizon reasoning but complicating temporal credit assignment, especially off-policy. TOP-ERL (Li et al., 2024a) partitions each episode into fixed-length segments and trains a Transformer critic that attends across segments. With truncated N-step targets (Sutton et al., 1998), the critic predicts per-segment returns, enabling off-policy replay while exploiting attention for partial observability and long-range dependencies. On manipulation benchmarks, TOP-ERL improves over prior ERL-based, step-based, and on-policy value-based baselines (Li et al., 2024a).

Transformer critics for episodic vs. step-based control. TOP-ERL (Li et al., 2024a) is an early Transformer-critic method for *episodic* control: a ProDMP policy (Li et al., 2023) outputs full trajectories, and the critic evaluates segment-level returns along them. Replanning is discussed but not implemented in the released experiments. The method additionally relies on a Trust Region Projection Layer (TRPL) (Otto et al., 2021), typically uses $\sim$20M interactions, and still underperforms on some multi-phase Meta-World tasks (Yu et al., 2020) (e.g., *Assembly*, *Disassemble*) and Box–Pushing at tight tolerances (Otto et al.).

By contrast, T-SAC remains in the standard step-based, closed-loop regime: the policy outputs an action at every time step from the current state, and the Transformer critic is *prefix-conditioned* on short state–action windows sampled from replay. It is trained with non-soft N-step TD targets *without* importance sampling, keeping it closer to conventional off-policy actor–critic methods than to episodic ERL: temporal abstraction lives in the critic's conditioning and targets, not in an open-loop policy.

Decision Transformer and related offline sequence-modeling approaches (Chen et al., 2021; Janner et al., 2021) instead perform *policy-side* sequence modeling on fixed datasets, mapping past trajectories and target returns directly to actions. These methods are complementary to T-SAC: we use a Transformer only for the critic, in an *online*, off-policy setting. In principle, an offline Decision Transformer policy could be paired with a T-SAC-style critic, or our critic architecture could be adapted to evaluate trajectories generated by such sequence policies.

2.2 Training without Target Network

Target networks stabilize bootstrapped critics but slow value propagation and add complexity (Kim et al., 2019; Piché et al., 2021). Recent work instead limits target drift or smooths backups. The strongest result, **CrossQ**, achieves state-of-the-art (SOTA) sample efficiency in continuous control by removing the target network and stabilizes a single bootstrapped critic with BRN (Ioffe, 2017). Related target-free strategies include value smoothing (mellowmax) (Asadi & Littman, 2017; Kim et al., 2019), constrained/proximal updates (Durugkar & Stone, 2018; Ohnishi et al., 2019), function-space regularization and partial freezing (Piché et al., 2021; Asadi et al., 2024; Vincent et al., 2025) feature decorrelation (Mavrin et al., 2019). Theory unifies these mechanisms as alternatives to target networks via partial freezing, regularization, and separation of optimization dynamics (Fellows et al., 2023).

2.3 Action Chunking in Reinforcement Learning

Action chunking replaces per-step control with short open-loop sequences of actions (“chunks”), which can capture temporal structure, accelerate value propagation via longer effective horizons, and promote temporally coherent exploration (Kalyanakrishnan et al., 2021; Zhang et al., 2022). The trade-off is reduced reactivity within a chunk (Liu et al., 2024), but for long-horizon, sparse-reward manipulation this bias often pays off (Zhang et al., 2021; Gupta et al., 2019).

Reinforcement Learning with Action Chunking (Q-chunking) (Li et al., 2025) applies TD-based actor–critic learning directly in the chunked action space: the policy proposes an H-step action sequence and the critic evaluates $Q(s_t, a_{t:t+H-1})$, enabling unbiased H-step backups and efficient updates (Li et al., 2025). In their implementation, the critic is a simple MLP that ingests the state concatenated with the proposed action chunk (rather than a sequence model), which keeps the method lightweight while still reaping the benefits of temporally extended actions (Li et al., 2025).

3 Preliminaries

Off-Policy Reinforcement Learning. Reinforcement learning (RL) (Sutton et al., 1998) formalizes sequential decision making as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$: at time t an agent observes s_t, selects $a_t \sim \pi(\cdot \mid s_t)$, receives $r_t = r(s_t, a_t)$, and transitions to $s_{t+1} \sim P(\cdot \mid s_t, a_t)$; the goal is to learn a policy maximizing $J(\pi) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t r_t]$ using value functions $V^\pi(s)$ and $Q^\pi(s,a)$ that satisfy Bellman consistency, with $Q^\star$ inducing the optimal policy. Algorithms differ in how they estimate and improve these quantities—value-based learning (Watkins & Dayan, 1992; Hessel et al., 2018; Van Hasselt et al., 2016; Rummery & Niranjan, 1994), actor–critic (Mnih et al., 2016; Schulman et al., 2017; 2015a; Fujimoto et al., 2018), or direct policy optimization (Kakade, 2001; Peters & Schaal, 2008)—while managing exploration vs. exploitation (Sutton et al., 1998). Off-policy RL learns a target policy π from data generated by a (possibly different) behavior policy μ, reusing transitions (s, a, r, s') via replay buffers and bootstrapped Bellman updates; distribution mismatch when evaluating π from μ-data can be corrected (e.g., with IS (Sutton et al., 1998)). This decoupling enables efficient experience reuse and underpins methods like Q-learning (Watkins & Dayan, 1992) and the SAC (Haarnoja et al., 2018a) family.

Soft Actor-Critic Let $\pi_\theta(a \mid s)$ be a stochastic policy with parameters θ. Let $Q_\psi(s,a)$ be the critic with parameters ψ, and let Q_ϕ be its target network (e.g., a Polyak-averaged copy of Q_ψ). SAC (Haarnoja et al., 2018a) maximizes a maximum-entropy objective to improve robustness and exploration:

$$J(\pi_\theta) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \big(r_t + \alpha\, \mathcal{H}(\pi_\theta(\cdot \mid s_t))\big)\Big].$$

where γ is the discount factor.

The Bellman target is

$$y_t = r_t + \gamma\, \mathbb{E}_{a' \sim \pi_\theta(\cdot|s_{t+1})}\big[\, Q_\phi(s_{t+1}, a') - \alpha \log \pi_\theta(a' \mid s_{t+1})\,\big],$$

and the critic minimizes the squared error

$$J_Q(\psi) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\Big[\tfrac{1}{2}\big(Q_\psi(s_t,a_t) - y_t\big)^2\Big].$$

The actor minimizes

$$J_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\; a\sim\pi_\theta(\cdot|s)}\big[\alpha \log \pi_\theta(a \mid s) - Q_\psi(s,a)\big].$$

The temperature α is tuned to match a target entropy $\bar{\mathcal{H}}$ by minimizing

$$J(\alpha) = \mathbb{E}_{s\sim\mathcal{D},\; a\sim\pi_\theta(\cdot|s)}\big[-\alpha\big(\log \pi_\theta(a \mid s) + \bar{\mathcal{H}}\big)\big] \qquad \text{(Haarnoja et al., 2018b)}.$$

N-step Returns and IS Using the same notation as above, on-policy N-step targets speed up credit assignment (Sutton et al., 1998; Schulman et al., 2015b; Mnih et al., 2016), i.e.,

$$y^{(n)}_{t,\text{soft}} = \sum_{k=0}^{n-1} \gamma^k\, \mathbb{E}_{a_{t+k}\sim\pi_\theta}\big[r_{t+k} - \alpha \log \pi_\theta(a_{t+k} \,|\, s_{t+k})\big] + \gamma^n\, \mathbb{E}_{a\sim\pi_\theta(\cdot|s_{t+n})}\big[Q_\phi(s_{t+n}, a)\big].$$

With off-policy data drawn from a behavior policy $\mu \neq \pi_\theta$, per-decision importance ratios

$$\rho_{t+k} = \frac{\pi_\theta(a_{t+k} \mid s_{t+k})}{\mu(a_{t+k} \mid s_{t+k})}$$

can be used to correct the distributional mismatch (Sutton et al., 1998), i.e.,

$$\hat{G}^{(n)}_{t,\text{soft}} = \sum_{k=0}^{n-1} \left(\gamma^k \prod_{j=0}^{k-1} \rho_{t+j}\right) \big[r_{t+k} - \alpha \log \pi_\theta(a_{t+k} \,|\, s_{t+k})\big] + \left(\gamma^n \prod_{j=0}^{n-1} \rho_{t+j}\right) Q_\phi(s_{t+n}, a_{t+n}),$$

with the convention that an empty product equals 1. When $\mu = \pi_\theta$, all ρ's are 1 and $\hat{G}^{(n)}_t$ reduces to the standard N-step target. Pure IS can introduce high variance, therefore the step length n cannot be chosen to be very large (Precup et al., 2000; Sutton et al., 1998; Espeholt et al., 2018).

Averaged N-step Returns for Critic Updates Using N-step returns is a standard way to reduce target bias for the critic (Sutton et al., 1998). For a starting index t and horizon $n \in [1, \texttt{max\_length}]$, following Zhang et al. (2022) we define the N-step target as

$$G^{(n)}_{non-soft}(s_t, a_t, \ldots, a_{t+n-1}) \;=\; \sum_{j=0}^{n-1} \gamma^j r_{t+j} \;+\; \gamma^n\, V_\phi(s_{t+n}), \tag{1}$$

with discount $\gamma \in (0, 1]$ and a *target* network parameterized by ϕ. Here $V_\phi(s) := \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\big[\, Q_\phi(s,a)\big]$ is the (non-entropy) bootstrap value under the current policy. While larger n reduces bootstrapping bias, the variance of $G^{(n)}$ typically grows with n (Precup et al., 2000). A practical variance reduction is to average partial returns (Konidaris et al., 2011; Daley et al., 2024):

$$\bar{G}^{(n)} \;=\; \frac{1}{n} \sum_{i=1}^{n} G^{(i)}. \tag{2}$$

This averaging lowers the variance of the reward-sum component from $\mathcal{O}(n)$ toward roughly $\mathcal{O}(n/4)$–$\mathcal{O}(n/3)$ (decreasing with n, depending on reward correlations), and makes the value-estimation term decay as $1/n$; under the same assumptions as (Daley et al., 2024), the full proof appears in App. C. This motivates using multiple horizons during critic training (see § 4.2). However, in our T-SAC implementation, we do not average N-step returns, as this strategy performs poorly in sparse-reward settings (see App. F).

4 Transformer-based Soft Actor-Critic (T-SAC)

4.1 N-step Returns for Critic Updates

4.1.1 Gradient-Level Averaging of N-Step Returns

Notation. For horizon i, define the prefix-conditioned online critic output $Q_\psi^{(i)} := Q_\psi(s_t, a_t, \dots, a_{t+i-1})$. Directly averaging targets can dilute sparse reward signals (App. F). Instead, we form per-horizon losses

$$L_i(\psi) = \tfrac{1}{2}\big(Q_\psi^{(i)} - G^{(i)}\big)^2, \qquad i = 1, \dots, n, \tag{3}$$

where a shared-weights *online* critic outputs $Q_\psi^{(i)}$ for each prefix $(s_t, a_t, \dots, a_{t+i-1})$ (s_t and a_t use separate embedding layers). We then *average gradients* across horizons:

$$\nabla_\psi \bar{L} = \frac{1}{n}\sum_{i=1}^{n} \nabla_\psi L_i(\psi). \tag{4}$$

Because adjacent horizons have overlapping targets and correspond to adjacent decoder positions in the same network, their per-parameter gradient contributions are positively—but not perfectly—correlated. Averaging therefore reduces update variance while preserving sparse signals (App. D, F; Fig. 2).

4.1.2 Stable Critic Learning Without Importance Sampling

Standard off-policy N-step TD presumes that post-a_t actions are drawn from the current policy π_θ, which mismatches replay generated by a behavior policy μ. Per-decision IS with $\rho_{t+k} = \frac{\pi_\theta(a_{t+k}|s_{t+k})}{\mu(a_{t+k}|s_{t+k})}$ corrects this but injects high variance (Precup et al., 2000; Sutton et al., 1998; Espeholt et al., 2018).

Similarly to Li et al. (2024a), we instead change the target: the critic predicts *prefix-conditioned* values for realized prefixes from replay,

$$\{\, Q_\psi(s_t, a_{t:t+i-1}) \,\}_{i=1}^{n},$$

with i-step targets

$$G^{(i)}(s_t, a_{t:t+i-1}) = \sum_{j=0}^{i-1} \gamma^j r_{t+j} + \gamma^i V_\phi(s_{t+i}), \tag{5}$$

and the loss

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{(s_t, a_{t:t+n-1})\sim\mathcal{D}}\Big[\tfrac{1}{n}\sum_{i=1}^{n}\big(Q_\psi(s_t, a_{t:t+i-1}) - G^{(i)}(s_t, a_{t:t+i-1})\big)^2\Big]. \tag{6}$$

As rewards follow the *recorded* prefix $a_{t:t+i-1}$, no assumption that actions came from π_θ is needed, and hence, no IS is required. Only the bootstrap at $t+i$ depends on π_θ via $V_\phi(s_{t+i})$.

Supervising short windows with multi-horizon targets and averaging their gradients yields stable updates and preserves sparse signals, enabling "multi-step supervision, one-step policy update" *without* IS (Fig. 2, 9b).

4.1.3 Connection to standard N-step TD and theoretical guarantees

Equations 5–6 can be viewed as a standard multi-step TD update in an MDP where each action prefix $a_{t:t+i-1}$ is treated as an extended action. For a fixed horizon i, we define

$$x = (s_t, a_{t:t+i-1}),$$

use equation 5 as the N-step target $G^{(i)}(x)$, and minimize the squared TD error

$$\big(Q_\psi(x) - G^{(i)}(x)\big)^2,$$

exactly as in classical N-step Q-learning.

The key difference to off-policy N-step TD with importance sampling (IS) is what the critic is asked to predict. IS-corrected targets are (in principle) unbiased for Q^π, but have high variance and typically require clipping when behavior and target policies differ. Our critic instead learns the value of realized prefixes under the replay distribution.

From a theoretical perspective, conditioned on a given state s_t and realized prefix $a_{t:t+i-1}$, the distribution over future rewards is fully determined by the environment dynamics and does not depend on how this prefix was generated (behavior versus target policy). Empirically this yields more stable long-horizon learning. See App. E for the formal connection to existing N-step TD theory.

4.2 Critic Network and Objective

Our critic is a causal Transformer that ingests $(s_t, a_t, a_{t+1}, \ldots, a_{t+n-1})$ and outputs the n prefix-conditioned values $\{Q_\psi(s_t, a_t, \ldots, a_{t+i-1})\}_{i=1}^{n}$ (Fig. 1). For a mini-batch of L trajectories, a random start index $t \in [0, N - n]$, and horizons $i \in \{1, \ldots, n\}$ with n sampled uniformly from $\{\texttt{min\_length}, \ldots, \texttt{max\_length}\}$, the training objective is the mean-squared error over all horizons:

$$\mathcal{L}(\psi) = \frac{1}{L\,n} \sum_{k=1}^{L} \sum_{i=1}^{n} \Big(Q_\psi(s_t^k, a_t^k, \ldots, a_{t+i-1}^k) - G^{(i)}(s_t^k, a_t^k, \ldots, a_{t+i-1}^k) \Big)^2 . \tag{7}$$

During backpropagation we apply the gradient-level averaging across $\{L_i\}_{i=1}^{n}$ described above. This construction leverages multi-horizon targets and inherits their variance-reduction benefits without target-level signal dilution.

4.3 Policy Network and Objective

Following Ba et al. (2016); Parisotto et al. (2020) and Plappert et al. (2017), we apply Layer Normalization to the policy's hidden layers (before the nonlinearity); Plappert et al. (2017) report this configuration to be useful for continuous-control actor–critic, especially when exploration noise is injected. The objectives remain

$$J_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\theta} \big[\, \alpha \, \log \pi_\theta(a \mid s) - Q_\psi(s, a) \,\big], \tag{8}$$

$$J(\alpha) = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\theta} \big[\, -\alpha \, (\log \pi_\theta(a \mid s) + \bar{\mathcal{H}}) \,\big], \tag{9}$$

with target entropy $-\bar{\mathcal{H}}$ (typically $-\dim(\mathcal{A})$) and automatic temperature tuning (Haarnoja et al., 2018b). Unlike canonical SAC, our critic does not include entropy in the target; it estimates the standard (non-soft) action-value. The policy is optimized with an entropy-regularized objective, so exploration and regularization are handled entirely by the policy. This "non-soft critic + policy-side regularization" design is also used in MPO (Abdolmaleki et al., 2018), AWR/AWAC (Peng et al., 2019; Nair et al., 2020), and IQL/IDQL (Kostrikov et al., 2021; Hansen-Estruch et al., 2023). Throughout this paper, all value targets are **standard** (non-soft) action-values; the entropy term appears only in the policy objective and is not included in the critic targets.

4.4 Critic–parameter freezing enables target–free training

CrossQ (Bhatt et al., 2019) removes target networks via batch normalization (Ioffe, 2017) and bounded activations. In contrast, we eliminate Polyak updates with a short *critic–freezing* schedule: at the start of each critic segment we snapshot the online critic ($\phi \leftarrow \psi$), precompute and cache bootstrap targets $V_\phi(s)$ for all windows in that segment, and then freeze this snapshot while optimizing the online critic against the cached targets for the next K updates (reusing each segment across N_c windows; Gymnasium MuJoCo (Towers et al., 2024): K=20). This lightweight decoupling curbs target drift without batch renormalization or constrained activations, and on local–motion and sparse–reward tasks (e.g., Box–Pushing–Sparse (Otto et al.)) the resulting *hard–copy* schedule yields stable training that matches or exceeds Polyak updates.

Our scheme introduces a single hyperparameter, the freezing interval K, i.e., the number of critic updates for which we reuse a single value snapshot V_ϕ. Because targets are computed once per segment before we enumerate windows, the minimum effective freezing interval is the segment length

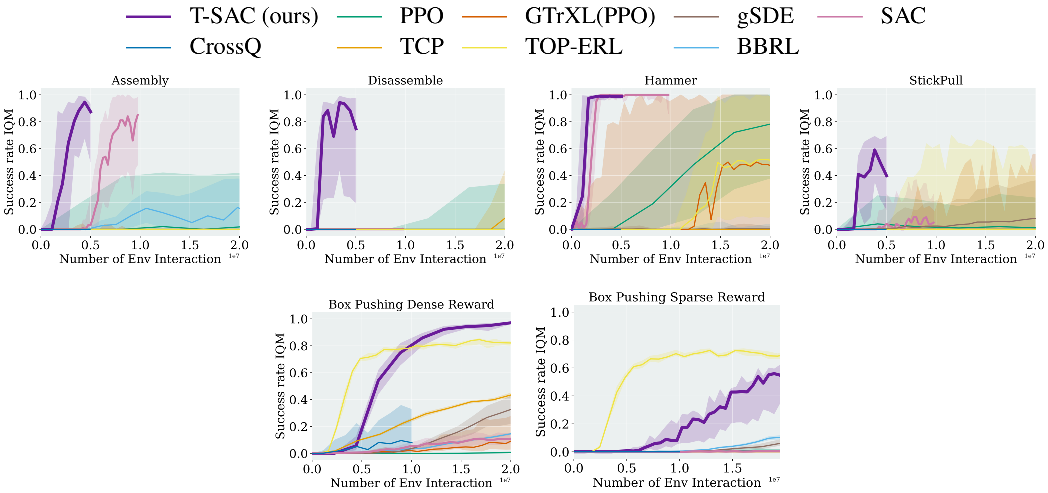


Figure 2: Success-rate IQM vs. environment interactions on challenging Meta-World ML1 tasks and FANCY-GYM Box-Pushing. Panels show Assembly, Disassemble, Hammer, and Stick-Pull, plus Box-Pushing under dense and sparse rewards. Success is counted only at the final timestep.

L_{seg} (for local-motion tasks, $L_{\text{seg}} = 20$). Sweeping $K \in \{20, 100, 1000, 10000\}$ on Gymnasium MuJoCo Walker2d (Fig. 4g), we observe largely stable performance with only mild degradation for the largest K, suggesting that segment-level target caching already provides useful stabilization and that K is not a brittle hyperparameter in our setting.

5 Experiments

Positioning T–SAC. Prior value–based RL largely splits into (i) *step–based* methods (e.g., SAC (Haarnoja et al., 2018a), CrossQ (Bhatt et al., 2019)) that dominate local–motion tasks (e.g., Ant (Towers et al., 2024)), and (ii) *episodic/trajectory–level* methods (e.g., BBRL (Otto et al., 2023a), TOP-ERL (Li et al., 2024a)) that excel on long–horizon problems (e.g., Box–Pushing (Otto et al.), Meta–World (Yu et al., 2020)). T-SAC partially narrows the gap between these regimes: it retains one–step policy updates while using a sequence–conditioned Transformer critic, and empirically matches standard SAC on local–motion benchmarks while outperforming existing Transformer–based approaches (e.g., GTrXL–style policies and TOP-ERL) on our long–horizon tasks.

5.1 Environments and Seeds

We evaluate T–SAC on 57 tasks spanning Meta–World ML1 (50) (Yu et al., 2020), Gymnasium MuJoCo locomotion (5) (Towers et al., 2024), and Box–Pushing (dense/sparse; 2) (Otto et al.). Meta–World probes task generalization; Gymnasium MuJoCo covers standard locomotion; and Box–Pushing stresses precise, contact–rich manipulation. Unless noted otherwise, we report means over 8 seeds (ablations use 4) with 95% bootstrap confidence intervals (Agarwal et al., 2021). Training time per 1M environment steps, compared to off–policy baselines, is shown in App. I. Baseline implementations and hyperparameters are detailed in App. K and App. L, with environment details in App. J.

5.2 Meta–World Results

We run Meta–World ML1 with UTD= 1, policy delay= 5, batch size 512; training time is ∼3 h per 1M env steps. Across 50 tasks, T–SAC solves most within ∼5M steps and yields stronger aggregated IQM than strong baselines (per–task curves in App. A). On the hardest multi–phase tasks (Assembly, Disassemble, Hammer, Stick–Pull) T–SAC is particularly strong (Fig. 2). In contrast, TOP–ERL (Li et al., 2024a) typically requires 20M steps to reach similar aggregates. All comparisons use 5M env steps for T–SAC, while many baselines use larger budgets (Fig. 7, 8). Success is evaluated only

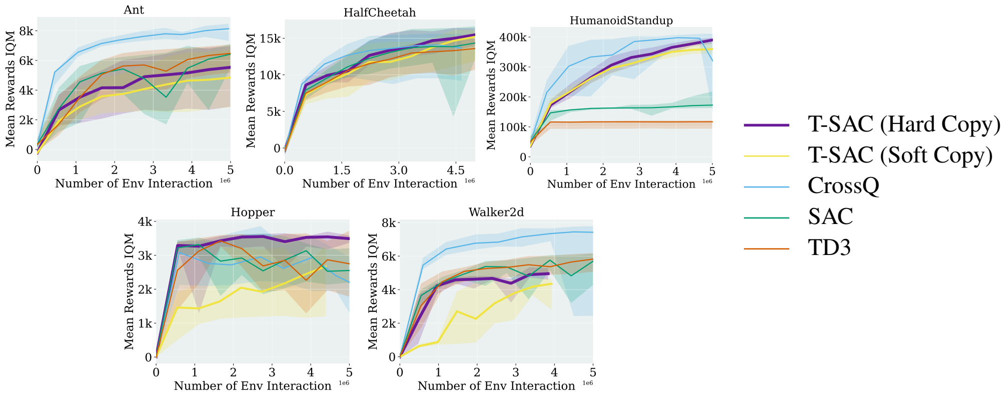


Figure 3: Episode return (IQM) vs. environment interactions for Ant, HalfCheetah, HumanoidStandup, Hopper, and Walker2d. Evaluation follows Gymnasium v4 native shaping/termination (no reward normalization); we report undiscounted return and use deterministic-policy evaluation.

at the final step (App. J), and our aggregates compute IQM *per task* and then average across tasks (unlike pooled–task IQM in TOP–ERL).

5.3 Box Pushing (Dense and Sparse)

We evaluate dense and sparse variants of FancyGym (Otto et al.) Box–Pushing with tight success tolerances (position $\pm 5\,\text{cm}$, orientation $\pm 0.5\,\text{rad}$). Under dense shaping, T–SAC attains **96.8%** success (Fig. 2), exceeding prior step–based baselines ($\leq 85\%$) under the same protocol. Under sparse rewards—where these terms apply only at the terminal step—T–SAC with the hard–copy critic reaches **60%** success, compared to TOP–ERL's **70%**. Thus T–SAC is state-of-the-art on Meta-World ML1 and dense Box-Pushing, and competitive under sparse rewards.

5.4 Gymnasium MuJoCo

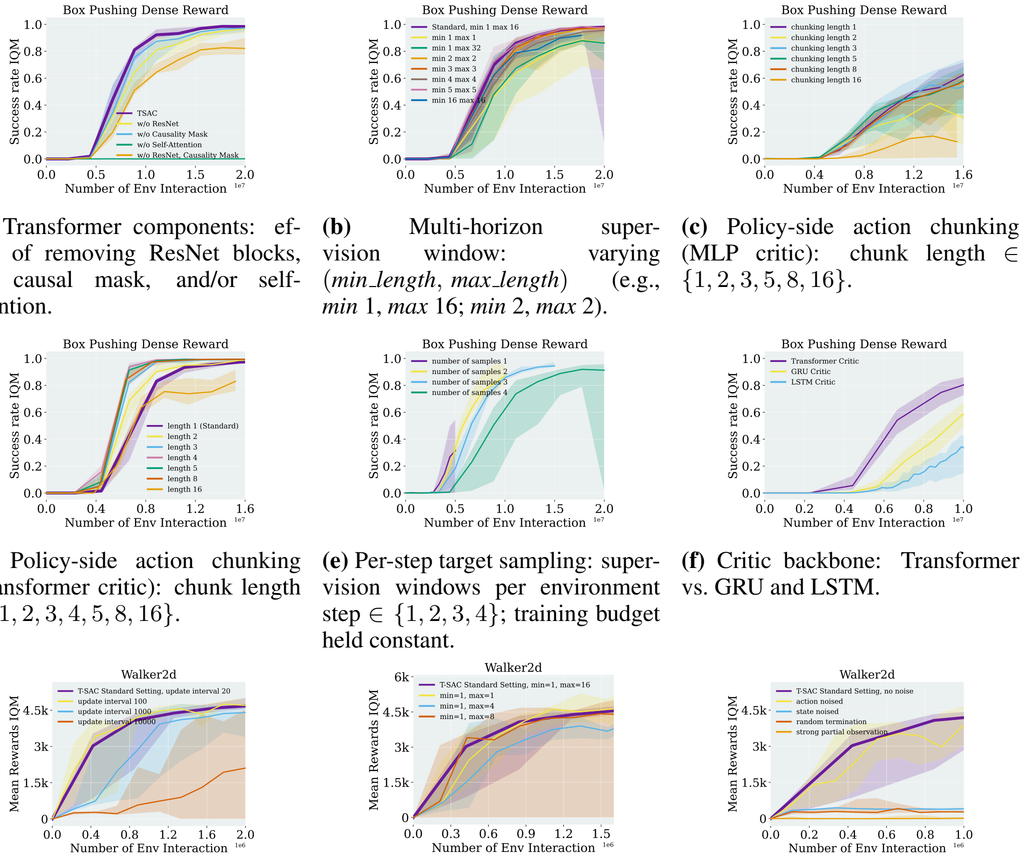
A lightweight *critic–parameter freezing* schedule (§ 4.4, App. B) enables target–free training: we remove the target network while retaining SAC–style stability at low update rates ($\text{UTD} \approx 0.75$) and consistently match or surpass Polyak updates. Across the five Gymnasium MuJoCo tasks, T–SAC is competitive with or better than SAC on *Ant*, *Hopper*, and *Walker2d*, with the largest gains on *HumanoidStandup* and *HalfCheetah* (Fig. 3), and we do not observe slower early convergence despite conditioning the critic on multi–step sequences from an early exploratory policy. Because episodes have variable length, we apply a simple action mask when constructing N-step targets from fixed–length windows to avoid bootstrapping across episode boundaries (App. H); this mask is an implementation detail rather than a core component of T–SAC and does not degrade performance.

5.5 Ablation Study

We conduct targeted ablations on FancyGym Box–Pushing (dense) and MuJoCo Walker2d. These ablations are structured to disentangle the effect of the sequence-conditioned Transformer critic—our main algorithmic contribution—from supporting design choices. Within each ablation group, all settings are identical except for the component under test; across groups, minor differences (e.g., training budget or `step_length`) arise from compute limits and are stated explicitly.

5.5.1 Transformer Components

We ablate three parts of the Transformer critic—ResNet blocks, the causal mask, and self-attention—holding all other settings fixed (Fig. 4a). Removing only self-attention invalidates the segment-conditioned objective and typically diverges. Removing self-attention *together with* the ResNet and causal mask reduces the critic to a plain MLP; we compare this baseline in § 5.5.3 and Fig. 4c.



(a) Transformer components: effect of removing ResNet blocks, the causal mask, and/or self-attention.

(b) Multi-horizon supervision window: varying (*min_length*, *max_length*) (e.g., *min* 1, *max* 16; *min* 2, *max* 2).

(c) Policy-side action chunking (MLP critic): chunk length $\in \{1, 2, 3, 5, 8, 16\}$.

(d) Policy-side action chunking (Transformer critic): chunk length $\in \{1, 2, 3, 4, 5, 8, 16\}$.

(e) Per-step target sampling: supervision windows per environment step $\in \{1, 2, 3, 4\}$; training budget held constant.

(f) Critic backbone: Transformer vs. GRU and LSTM.

(g) Effect of freeze interval K on Walker2d: T-SAC is stable down to $K = 20$, while very large K slows learning.

(h) Effect of reuse factor (`step_length`) on locomotion: moderate values are robust; larger `step_length` yields only mild gains.

(i) Robustness on Walker2d under noise $\mathcal{N}(0, 0.05)$, false terminations ($p = 0.1$), and partial observability (last half of observation removed).

Figure 4: Ablations: transformer-critic design and training settings. Within each panel, methods share the same interaction budget and differ only in the ablated component.

5.5.2 Step Length and Min Length (Reuse factor)

We sweep `step_length` and `min_length`, which bound the multi-horizon (N-step) supervision window. At each update we sample $n \in \{\texttt{min\_length}, \ldots, \texttt{step\_length}\}$; by default $n \sim \text{Unif}\{1, \ldots, 16\}$. Using a *fixed* horizon L smooths optimization but slightly reduces final performance (Fig. 4b), partly because the last $L-1$ states of each segment never serve as starting indices—an effect amplified for large L (e.g., 16). Despite standard guidance to keep $n \leq 5$ (Precup et al., 2000; Sutton et al., 1998; Espeholt et al., 2018), our Transformer critic with gradient-level averaging is stable up to $n=16$ and benefits from longer windows (Fig. 4b). An analogous sweep under the hard-copy scheme on Gymnasium Walker2d shows consistent trends (Fig. 4h).

5.5.3 Comparison to Multi-step MLP (Reinforcement Learning with Action Chunking)

With 16M environment steps (default budget: 20M), a multi-step MLP critic underperforms the Transformer critic; policy-side chunking with an MLP yields only modest gains (Figs. 5, 4c). These policy-side baselines follow the Q-Chunking (QC) architecture of Li et al. (2025): we use the same action-chunking policy but drop the offline behavior-cloning constraint to match our online off-policy setting. In contrast, chunking the Transformer critic helps: the best setting uses chunk length 4, converges by $\approx$ 10M steps, reaches 99.5% final success, and shows no late-stage divergence across seeds (Fig. 4d). Comparing the QC-style MLP critic (Fig. 4c) to the QC-style Transformer critic (Fig. 4d) thus isolates the benefit of the sequence-conditioned critic under the same chunked policy. This gain is not just from richer input features: in Fig. 5(b), with `min_length` = `max_length` = 1 the Transformer critic still outperforms the `chunking_length` = 1 baseline in Fig. 5(c), even though both operate at single-step temporal resolution, indicating that the advantage comes from how it integrates N-step returns and averages gradients over short trajectories. The MLP critic also suffers causal leakage: because it consumes fixed-length segments, $Q(s_t, a_t)$ is effectively conditioned on $(s_t, a_{t:t+n})$, thereby "peeking" at future actions $a_{t+1:t+n}$ (Li et al., 2025). Our Transformer critic applies a causal mask so each token attends only to positions $t' \leq t$ and outperforms an ablation without masking (Fig. 4a). Finally, the MLP critic imposes fixed-length inputs and ties the policy chunk length to the critic, whereas the Transformer critic avoids these constraints.

5.5.4 Number of Samples Generated per Step

We vary the number of supervision windows sampled per environment step ("per-step target samples"). For this study we use `step_length` $= [1, 8]$ (standard: $[1, 16]$). Unlike conventional SAC (one target per step), T-SAC benefits from generating multiple windows (Fig. 4e); our default is 4. In practice, use four vectorized envs or collect a single trajectory of length $4\times$`max_length` and slice it (App. H). Intuitively, multiple windows raise the share of fresh samples in each batch: with one window, once selected there are none left; with four, three remain.

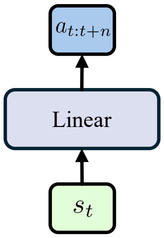


Figure 5: Policy chunking (adapted from Li et al. (2025)).

5.5.5 GRU/LSTM as the Critic

We replace the Transformer critic with GRU and LSTM variants under identical training (10M env steps; standard: 20M). Although recurrent critics can model action sequences, our gradient–level averaging analysis (§ 4.1.1; App. D) does not directly apply, and parallelism is reduced (Fig. 12. Empirically, both GRU and LSTM underperform the Transformer critic on Box–Pushing in our setting (Fig. 4f).

5.5.6 Robustness under noise and partial observability

We evaluate robustness to injected noise on actions and states, stochastic early termination, and partial observability via short observation windows. T-SAC degrades gracefully under action and state noise, retaining a clear performance margin over SAC and CrossQ. Stochastic termination and partial observability lead to larger drops and higher variance, but T-SAC remains at least as stable as these baselines, suggesting that sequence–conditioned critics help mitigate such effects.

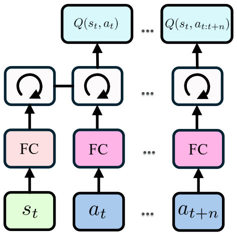


Figure 6: Structure of the recursive network used in our experiments.

6 Conclusion and Future Work

On Meta-World ML1 multiphase and FancyGym box-pushing tasks, T-SAC with a Transformer critic attains state-of-the-art success rates under fixed training budgets (5M and 20M environment steps, respectively) and a common evaluation protocol (success over 8 seeds; see §5). The sequence-conditioned critic provides smoother value estimates and more coherent long-horizon credit assignment than both largely open-loop multiphase pipelines and standard step-based value methods, yielding higher-quality continuous control.

Our study is restricted to online continuous control with low-dimensional observations. Extending T-SAC to discrete-action domains and to pixel-based or strongly partially observable settings (e.g., with visual or belief-state encoders) is nontrivial—preliminary experiments revealed instabilities and high sensitivity to architectural and optimization choices. Applying T-SAC to real-robot tasks and developing theory for when critic-side chunking provably helps, including representation analyses and shared Transformer backbones for actor–critic, are important directions for future work.

Reproducibility Statement

We made substantial efforts to ensure reproducibility. The paper and appendix specify environments, evaluation protocols, and all hyperparameters used. Upon acceptance, we will release a public GitHub repository containing the implementation of the proposed algorithms, experiment scripts, and trained models. Detailed descriptions of the experimental setup, including configuration files, are provided in the appendix to enable independent reimplementation during review.

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.

Pranav Agarwal, Aamer Abdul Rahman, Pierre-Luc St-Charles, Simon JD Prince, and Samira Ebrahimi Kahou. Transformers in reinforcement learning: a survey. *arXiv preprint arXiv:2307.05979*, 2023.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pp. 243–252. PMLR, 2017.

Kavosh Asadi, Yao Liu, Shoham Sabach, Ming Yin, and Rasool Fakoor. Learning the target network in function space. *arXiv preprint arXiv:2406.01838*, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. *arXiv preprint arXiv:1902.05605*, 2019.

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Brett Daley, Martha White, and Marlos C Machado. Averaging n-step returns reduces variance in reinforcement learning. *arXiv preprint arXiv:2402.03903*, 2024.

Ishan Durugkar and Peter Stone. Td learning with constrained gradients. 2018.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Mattie Fellows, Matthew JA Smith, and Shimon Whiteson. Why target networks stabilise temporal difference methods. In *International Conference on Machine Learning*, pp. 9886–9909. PMLR, 2023.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *Advances in neural information processing systems*, 30, 2017.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Shivaram Kalyanakrishnan, Siddharth Aravindan, Vishwajeet Bagdawat, Varun Bhatt, Harshith Goka, Archit Gupta, Kalpesh Krishna, and Vihari Piratla. An analysis of frame-skipping in reinforcement learning. *arXiv preprint arXiv:2102.03718*, 2021.

Seungchan Kim, Kavosh Asadi, Michael Littman, and George Konidaris. Deepmellow: removing the need for a target network in deep q-learning. In *Proceedings of the twenty eighth international joint conference on artificial intelligence*, 2019.

George Konidaris, Scott Niekum, and Philip S Thomas. Td_gamma: Re-evaluating complex backups in temporal difference learning. *Advances in Neural Information Processing Systems*, 24, 2011.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Ge Li, Zeqi Jin, Michael Volpp, Fabian Otto, Rudolf Lioutikov, and Gerhard Neumann. Prodmp: A unified perspective on dynamic and probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 8(4):2325–2332, 2023.

Ge Li, Dong Tian, Hongyi Zhou, Xinkai Jiang, Rudolf Lioutikov, and Gerhard Neumann. Top-erl: Transformer-based off-policy episodic reinforcement learning. *arXiv preprint arXiv:2410.09536*, 2024a.

Ge Li, Hongyi Zhou, Dominik Roth, Serge Thilges, Fabian Otto, Rudolf Lioutikov, and Gerhard Neumann. Open the black box: Step-based policy updates for temporally-correlated episodic reinforcement learning. *arXiv preprint arXiv:2401.11437*, 2024b.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018. URL https://arxiv.org/pdf/1712.09381.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via closed-loop resampling. *arXiv e-prints*, pp. arXiv–2408, 2024.

Borislav Mavrin, Hengshuai Yao, and Linglong Kong. Deep reinforcement learning with decorrelation. *arXiv preprint arXiv:1903.07765*, 2019.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Shota Ohnishi, Eiji Uchibe, Yotaro Yamaguchi, Kosuke Nakanishi, Yuji Yasui, and Shin Ishii. Constrained deep q-learning gradually approaching ordinary q-learning. *Frontiers in neurorobotics*, 13:103, 2019.

Fabian Otto, Onur Celik, Dominik Roth, and Hongyi Zhou. Fancy gym. URL https://github.com/ALRhub/fancy_gym.

Fabian Otto, Philipp Becker, Ngo Anh Vien, Hanna Carolin Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. *arXiv preprint arXiv:2101.09207*, 2021.

Fabian Otto, Onur Celik, Hongyi Zhou, Hanna Ziesche, Vien Anh Ngo, and Gerhard Neumann. Deep black-box reinforcement learning with movement primitives. In *Conference on Robot Learning*, pp. 1244–1265. PMLR, 2023a.

Fabian Otto, Hongyi Zhou, Onur Celik, Ge Li, Rudolf Lioutikov, and Gerhard Neumann. Mp3: Movement primitive-based (re-) planning policy. *arXiv preprint arXiv:2306.12729*, 2023b.

Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Alexandre Piché, Valentin Thomas, Rafael Pardinas, Joseph Marino, Gian Maria Marconi, Christopher Pal, and Mohammad Emtiyaz Khan. Bridging the gap between target networks and functional regularization. *arXiv preprint arXiv:2106.02613*, 2021.

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. *arXiv preprint arXiv:1706.01905*, 2017.

Doina Precup, Richard S Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. 2000.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on robot learning*, pp. 1634–1644. PMLR, 2022.

Thomas Rückstieß, Martin Felder, and Jürgen Schmidhuber. State-dependent exploration for policy gradient methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 234–249. Springer, 2008.

Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn*, 1(1):14–24, 2010.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

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

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

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

Sahil Sharma, Aravind Srinivas, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. *arXiv preprint arXiv:1702.06054*, 2017.

Richard S Sutton, Andrew G Barto, et al. Reinforcement learning: An introduction. 1(1), 1998.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Théo Vincent, Yogesh Tripathi, Tim Faust, Yaniv Oren, Jan Peters, and Carlo D'Eramo. Bridging the performance gap between target-free and target-based reinforcement learning with iterated q-learning. *arXiv preprint arXiv:2506.04398*, 2025.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Haichao Zhang, Wei Xu, and Haonan Yu. Generative planning for temporally coordinated exploration in reinforcement learning. *arXiv preprint arXiv:2201.09765*, 2022.

Jesse Zhang, Haonan Yu, and Wei Xu. Hierarchical reinforcement learning by discovering intrinsic options. *arXiv preprint arXiv:2101.06521*, 2021.

A Appendix: Individual Meta-World Tests results

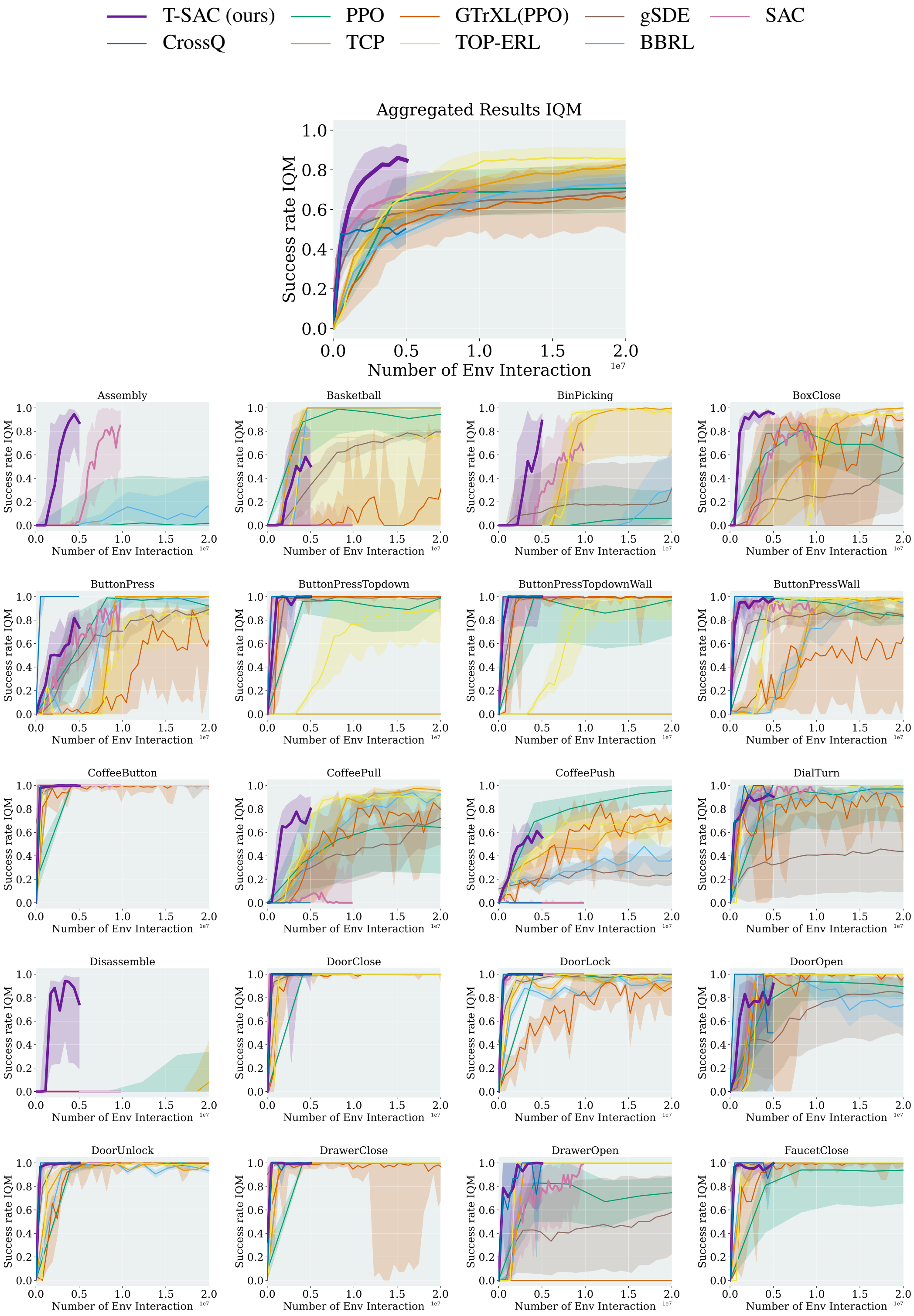


Figure 7: Success Rate IQM of each individual Meta-World tasks. (Part 1)

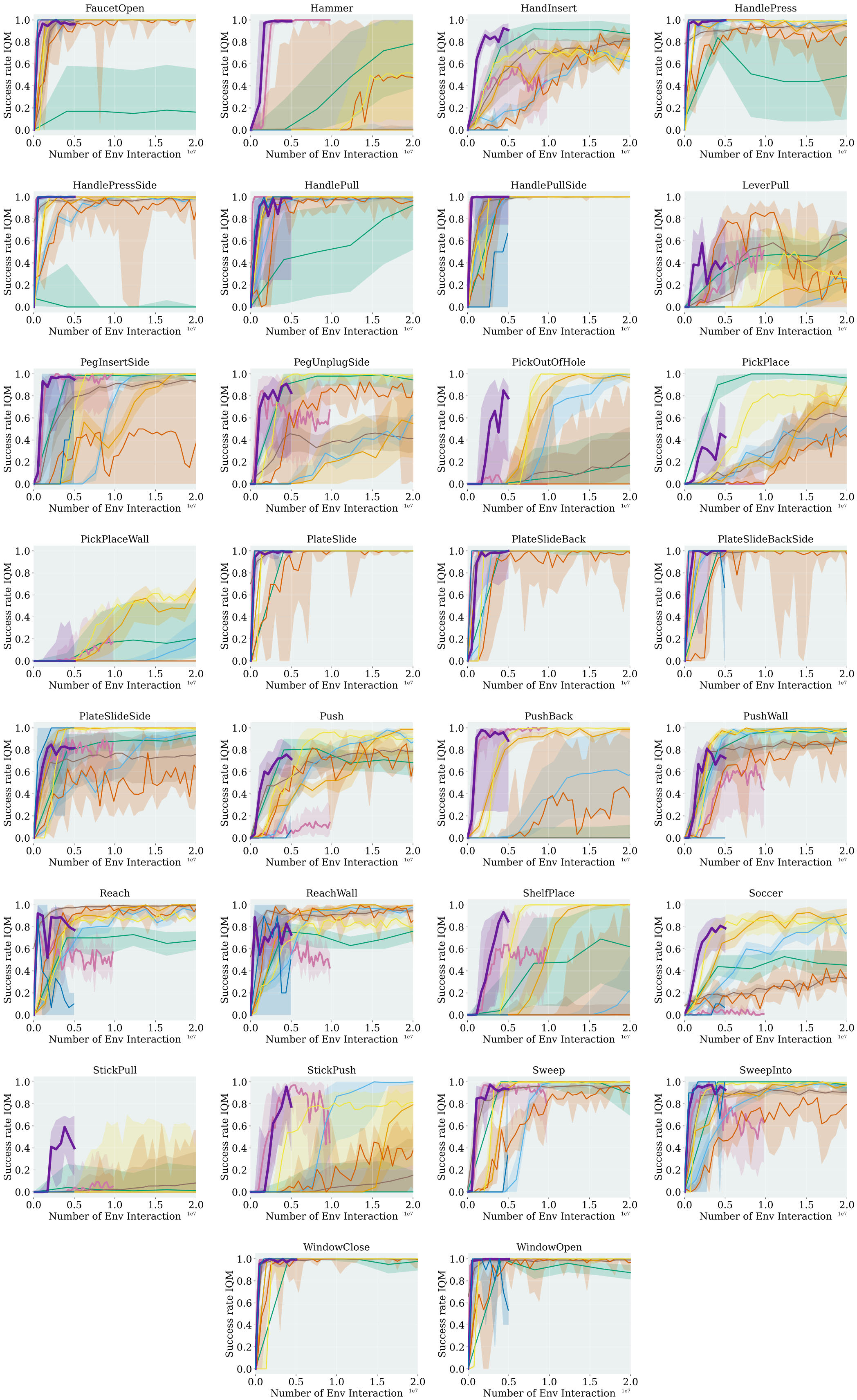


Figure 8: Success Rate IQM of each individual Meta-World tasks. (Part 2)

B Appendix: Detailed Algorithm Flowchart

```
Algorithm 1: T-SAC
Initialize: Critic params $\phi$; target critic $\phi_{\text{target}} \leftarrow \phi$; policy params $\theta$; temperature $\alpha$; replay
            buffer $\mathcal{B}$; environment reset $\to s_0$.
Input: Segment length cap $M$; window length bounds $(\ell_{\min}, \ell_{\max})$; updates per iteration $U$;
       critic steps $N_c$; policy steps $N_p$; soft update $\tau$; warmups: policy / temperature.
repeat
    Collect one segment (length $\leq M$);
    store $s_0$; $t \leftarrow 0$;
    while $t < M$ do
        sample $a_t \sim \pi_\theta(\cdot \mid s_t)$; step env $\to (r_t, s_{t+1}, d_t)$;
        append $(s_t, a_t, r_t, d_t, s_{t+1})$ to a temporary buffer;
        if $d_t$ then break;
        $t \leftarrow t+1$; $s_t \leftarrow s_{t+1}$;
    end
    push the whole segment $(s_{0:M}, a_{0:M-1}, r_{0:M-1}, d_{0:M-1})$ to $\mathcal{B}$;
    if $d_t$ then reset env $\to s_0$;
    else $s_0 \leftarrow s_M$;
    Parameter updates;
    for $u \leftarrow 1$ to $U$ do
        sample a batch of segments $\{(s_{0:M}, a_{0:M-1}, r_{0:M-1}, d_{0:M-1})\}_{b=1}^{B}$ from $\mathcal{B}$;
        precompute segment-wise bootstrapped targets using $Q_{\phi_{\text{target}}}$;
        ▷ reuse across $N_c$ windows
        for $k \leftarrow 1$ to $N_c$ do
            for each sequence in the batch, draw start $i$ uniformly over valid indices and draw
              $\ell \sim \mathcal{U}\{\ell_{\min}, \ldots, \ell_{\max}\}$ s.t. $i+\ell \leq$segment end;
            form windows $(s_{i:i+\ell+1}, a_{i:i+\ell}, r_{i:i+\ell}, d_{i:i+\ell})$ and the corresponding $N$-step
              returns;
            update critic parameters $\phi$ with the Transformer critic on these windows;
            ▷ critic update
            $\phi_{\text{target}} \leftarrow \tau\,\phi + (1-\tau)\,\phi_{\text{target}}$;
            ▷ soft (or hard) target update
        end
        if step > policy warmup then  sample a (fresh) batch of states from $\mathcal{B}$;
        for $k \leftarrow 1$ to $N_p$ do
            update policy $\theta$ by maximizing the SAC objective using $Q_\phi$;
            if step > temperature warmup then update temperature $\alpha$;
        end
        ;
    end
until convergence;
```

C Appendix: Proof of the variance reduction property of averaging of N-step returns

A convenient variance identity. Under the equicorrelation model,

$$\mathbb{E}[X_k] = m, \quad \mathrm{Var}(X_k) = v, \quad \mathrm{Cov}(X_k, X_\ell) = \rho\, v \ \ (k \neq \ell), \qquad \rho \geq 0,$$

any weighted sum $S = \sum_{k=0}^{N-1} a_k X_k$ satisfies

$$\mathrm{Var}(S) = v\left(\sum_{k=0}^{N-1} a_k^2 \ + \ \rho\left[\left(\sum_{k=0}^{N-1} a_k\right)^2 - \sum_{k=0}^{N-1} a_k^2\right]\right). \tag{10}$$

This follows by expanding Var and collecting diagonal/off-diagonal terms.

Reward part with discount $\gamma < 1$. Define the single N-step discounted reward sum and its triangular average by

$$R_N(\gamma) \ \triangleq \ \sum_{k=0}^{N-1} \gamma^k r_k, \qquad \bar{R}_N(\gamma) \ \triangleq \ \frac{1}{N}\sum_{i=1}^{N}\sum_{k=0}^{i-1} \gamma^k r_k \ = \ \sum_{k=0}^{N-1} w_k r_k, \quad w_k \triangleq \frac{N-k}{N}\gamma^k.$$

Let

$$S_0 \equiv S_0(N,\gamma) \triangleq \sum_{k=0}^{N-1} \gamma^{2k} = \frac{1-\gamma^{2N}}{1-\gamma^2}, \qquad T_0 \equiv T_0(N,\gamma) \triangleq \sum_{k=0}^{N-1} \gamma^k = \frac{1-\gamma^N}{1-\gamma}.$$

Also define the (discounted, triangular) weight aggregates

$$A_\gamma \ \triangleq \ \sum_{k=0}^{N-1} w_k^2 \ = \ \frac{1}{N^2}\sum_{k=0}^{N-1}(N-k)^2\gamma^{2k}, \qquad B_\gamma \ \triangleq \ \left(\sum_{k=0}^{N-1} w_k\right)^2 - A_\gamma. \tag{11}$$

Lemma 1 (Variance formulas for the discounted reward part). *Under the reward assumptions stated in the setup,*

$$\mathrm{Var}\big[R_N(\gamma)\big] = \sigma^2\big[S_0 + \rho\,(T_0^2 - S_0)\big], \qquad \mathrm{Var}\big[\bar{R}_N(\gamma)\big] = \sigma^2\big[A_\gamma + \rho\, B_\gamma\big].$$

Proof. Apply equation 10 with weights $a_k = \gamma^k$ for $R_N(\gamma)$ and $a_k = w_k$ for $\bar{R}_N(\gamma)$, and use the definitions of $S_0, T_0, A_\gamma, B_\gamma$. □

Proposition 1 (Reward-side variance ratio and bounds). *Define*

$$R_\gamma(N) \ \triangleq \ \frac{\mathrm{Var}[R_N(\gamma)]}{\mathrm{Var}[\bar{R}_N(\gamma)]} = \frac{S_0 + \rho\,(T_0^2 - S_0)}{A_\gamma + \rho B_\gamma}.$$

Then for all $N \geq 1$, $\rho \geq 0$, and $\gamma \in (0,1]$,

$$1 \ \leq \ R_\gamma(N) \ < \ 4.$$

Moreover, for $\gamma = 1$, $R_\gamma(N) \nearrow 4$ as $N \to \infty$; for any fixed $\gamma \in (0,1)$, $R_\gamma(N) \to 1$. When $\rho > 0$, $R_\gamma(N)$ is strictly increasing in N for $\gamma = 1$; for $\gamma < 1$ it need not be monotone.

Proof (sketch). The $\gamma = 1$ proof carries through verbatim after replacing the unweighted triangular weights j by the discounted weights $w_k = ((N-k)/N)\gamma^k$; the same algebraic positivity arguments yield the bounds. For the limits, when $\gamma < 1$ we have $w_k \to \gamma^k$ pointwise as $N \to \infty$ and dominated convergence gives $A_\gamma \to S_0$ and $\sum_k w_k \to T_0$, hence $R_\gamma(N) \to 1$. For $\gamma = 1$, the standard triangular-sum identities imply $R_1(N) \nearrow 4$. □

Bootstrap value part. Let $Z_i \triangleq V_{\phi_{\text{tar}}}(s_{t+i})$ denote the (target) values used for bootstrapping, and assume

$$\mathbb{E}[Z_i] = \nu, \quad \mathrm{Var}(Z_i) = \tau^2, \quad \mathrm{Cov}(Z_i, Z_j) = \kappa\,\tau^2 \ \ (i \neq j), \qquad \kappa \geq 0,$$

and that $\{r_k\}$ and $\{Z_i\}$ are independent unless stated otherwise. The single N-step bootstrap term and its triangular average are

$$B_N(\gamma) \triangleq \gamma^N Z_N, \qquad \bar{B}_N(\gamma) \triangleq \frac{1}{N}\sum_{i=1}^{N} \gamma^i Z_i.$$

Let

$$S_1 \equiv S_1(N,\gamma) \triangleq \sum_{i=1}^{N} \gamma^{2i} = \frac{\gamma^2(1-\gamma^{2N})}{1-\gamma^2}, \qquad C \equiv C(N,\gamma) \triangleq \sum_{i=1}^{N} \gamma^i = \frac{\gamma(1-\gamma^N)}{1-\gamma}.$$

Lemma 2 (Variance of the averaged bootstrap part)**.** *With* $\mathrm{Cov}(Z_i, Z_j) = \kappa\,\tau^2$ *for* $i \neq j$*,*

$$\mathrm{Var}[\bar{B}_N(\gamma)] = \frac{\tau^2}{N^2}\Big[S_1 + \kappa\,(C^2 - S_1)\Big].$$

Proof. Apply equation 10 with $a_i = \gamma^i/N$. □

Proposition 2 (Bootstrap-side variance ratio, bounds, and condition)**.** *Define*

$$R_B(N,\gamma,\kappa) \;\triangleq\; \frac{\mathrm{Var}[B_N(\gamma)]}{\mathrm{Var}[\bar{B}_N(\gamma)]} = \frac{N^2\,\gamma^{2N}}{S_1 + \kappa(C^2 - S_1)}.$$

Then for any $\kappa \in [0,1]$*,*

$$\frac{N^2\gamma^{2N}}{C^2} \;\leq\; R_B(N,\gamma,\kappa) \;\leq\; \frac{N^2\gamma^{2N}}{S_1}, \qquad \frac{\partial R_B}{\partial \kappa} < 0.$$

In particular, averaging reduces bootstrap variance ($R_B \geq 1$) *whenever*

$$\kappa \;\leq\; \kappa_\star(N,\gamma) \;\triangleq\; \frac{N^2\gamma^{2N} - S_1}{C^2 - S_1}.$$

For the uncorrelated case ($\kappa = 0$)*,* $R_B(N,\gamma,0) = N^2\gamma^{2N}/S_1$*.*

Proof. Monotonicity in κ is immediate from the denominator. The bounds follow from $S_1 \leq S_1 + \kappa(C^2 - S_1) \leq C^2$. Solve $R_B \geq 1$ for κ to get $\kappa_\star$. □

Putting the parts together. With $G_N(\gamma) = R_N(\gamma) + B_N(\gamma)$ and $\bar{G}_N(\gamma) = \bar{R}_N(\gamma) + \bar{B}_N(\gamma)$, and assuming independence between rewards and bootstrap values,

$$\frac{\mathrm{Var}[G_N(\gamma)]}{\mathrm{Var}[\bar{G}_N(\gamma)]} = \frac{\mathrm{Var}[R_N(\gamma)] + \mathrm{Var}[B_N(\gamma)]}{\mathrm{Var}[\bar{R}_N(\gamma)] + \mathrm{Var}[\bar{B}_N(\gamma)]}. \tag{12}$$

Since all terms are nonnegative,

$$\min\!\big\{R_\gamma(N),\, R_B(N,\gamma,\kappa)\big\} \;\leq\; \frac{\mathrm{Var}[G_N(\gamma)]}{\mathrm{Var}[\bar{G}_N(\gamma)]} \;\leq\; \max\!\big\{R_\gamma(N),\, R_B(N,\gamma,\kappa)\big\}.$$

Consequently:

- Because $R_\gamma(N) \geq 1$ (Prop. 1), if $R_B(N,\gamma,\kappa) \geq 1$ (e.g., $\kappa \leq \kappa_\star$), then averaging N-step targets strictly reduces total variance.
- Even if $R_B(N,\gamma,\kappa) < 1$, the overall ratio in equation 12 remains ≥ 1 whenever the reward-side gain dominates:

$$R_\gamma(N) \;\geq\; \frac{\mathrm{Var}[\bar{B}_N(\gamma)]}{\mathrm{Var}[B_N(\gamma)]} = \frac{S_1 + \kappa(C^2 - S_1)}{N^2\gamma^{2N}}.$$

Dependence between rewards and bootstrap values. If $\mathrm{Cov}\big(R_N(\gamma), B_N(\gamma)\big)$ and $\mathrm{Cov}\big(\bar{R}_N(\gamma), \bar{B}_N(\gamma)\big)$ are nonzero, the numerator/denominator of equation 12 each acquire an additional covariance term. The sandwich bound above still applies after inserting these, and a crude control is $|\mathrm{Cov}(X,Y)| \leq \sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}$ (Cauchy–Schwarz), which cannot overturn the above conclusions unless the cross-covariances are pathologically large.

Useful closed forms. Besides S_0, S_1, T_0, C above, one has

$$\sum_{k=0}^{N-1}(N-k)\gamma^k = \frac{1-(N+1)\gamma^N + N\gamma^{N+1}}{(1-\gamma)^2}.$$

A closed form for $\sum_{k=0}^{N-1}(N-k)^2\gamma^{2k}$ (hence A_γ via equation 11) follows from the standard identities for $\sum kx^k$ and $\sum k^2x^k$ after the change $k \mapsto N-1-k$; we omit it as not needed for the bounds above.

Remarks. (i) Setting $\gamma \to 1$ recovers the *undiscounted* results: $S_0, T_0, S_1, C \to N$ and $A_\gamma, B_\gamma \to A/N^2, B/N^2$, where A, B are the non-discounted triangle sums.
(ii) As $N \to \infty$ with fixed $\gamma < 1$, $S_1 \to \gamma^2/(1-\gamma^2)$ and $C \to \gamma/(1-\gamma)$ while $N^2\gamma^{2N} \to 0$; thus $R_B(N,\gamma,\kappa) \to 0$. For typical RL regimes ($\gamma \gtrsim 0.95$, moderate N), $\kappa_\star(N,\gamma)$ is positive and large, so averaging still reduces bootstrap variance over a wide range of κ.
(iii) $R_\gamma(N)$ is horizon- and discount-agnostic in the sense of the bound $1 \le R_\gamma(N) < 4$; for $\gamma < 1$, its large-N limit is 1. It is the principal driver of the overall variance reduction.

On the equicorrelation assumption. We assumed an equicorrelation (exchangeable) model for the reward noise and for the bootstrapped values: identical variances and a common pairwise correlation (ρ and κ, respectively). This is a standard device that yields closed forms while capturing the empirically relevant regime of positively correlated temporal signals in RL trajectories.

The key conclusions above are *robust* to relaxing equicorrelation. Let Σ be any covariance matrix for $(r_0, \ldots, r_{N-1})$ with nonnegative entries (i.e., nonnegative autocovariances). For any nonnegative weight vector w, the variance is $w^T\Sigma w$ and increases monotonically with each off-diagonal entry. Since the triangular weights have strictly smaller ℓ_2 norm and smaller sum than the flat weights of the single N-step sum, the reward-side variance reduction persists under a wide range of stationary, positively correlated processes (including Toeplitz/lag-dependent models such as $\mathrm{Cov}(r_k, r_\ell) = \sigma^2\rho_{|k-\ell|}$ with $\rho_d \ge 0$). The specific constant 4 in the upper bound is tight for the exchangeable model; with general lag structure the same $[1,4)$ bracket continues to hold under mild bounded-correlation conditions (e.g. $\sup_{k\neq\ell}\mathrm{Corr}(r_k, r_\ell) \le 1$), while the uncorrelated case ($\rho_d \equiv 0$) recovers the $[1,3)$ limit.

For the bootstrap part, assuming a common correlation κ across $\{Z_i\}$ is likewise a tractable approximation: the explicit ratio $R_B(N,\gamma,\kappa)$ is decreasing in κ, so weaker dependence only strengthens the variance reduction. More general lag-dependent models $\mathrm{Cov}(Z_i, Z_j) = \tau^2\kappa_{|i-j|}$ with $\kappa_d \ge 0$ lead to the same qualitative behavior (smaller weights and partial averaging reduce variance), with our equicorrelation formulas serving as convenient upper/lower benchmarks.

When to be cautious. If the process exhibits strong *negative* or oscillatory correlations (e.g. alternation effects), equicorrelation overstates the benefit of averaging; in such cases, replacing the common ρ (or κ) by a small set of lag-specific parameters $(\rho_1, \rho_2, \ldots)$ is safer. Empirically, one can estimate the sample autocovariance and plug it into $w^T\hat{\Sigma}w$ to verify the inequalities numerically.

D Appendix: Variance reduction from gradient-level averaging with a shared-weights Transformer critic

Setup. Fix a trajectory position t. A Transformer critic with shared parameters ψ outputs

$$Q_\psi^{(1)},\, Q_\psi^{(2)},\, \dots,\, Q_\psi^{(n)},$$

where $Q_\psi^{(i)}$ predicts the i-step return for the same prefix $(s_t, a_t, \dots, a_{t+i-1})$. Let $G^{(i)}$ denote the i-step target and define the per-horizon MSE

$$L_i(\psi) = \tfrac{1}{2}\big(Q_\psi^{(i)} - G^{(i)}\big)^2, \qquad \bar{L}(\psi) \triangleq \frac{1}{n}\sum_{i=1}^{n} L_i(\psi).$$

In implementation we *average their gradients* during backprop:

$$\nabla_\psi \bar{L}(\psi) \;=\; \frac{1}{n}\sum_{i=1}^{n} \nabla_\psi L_i(\psi) \;=\; \frac{1}{n}\sum_{i=1}^{n} \big(Q_\psi^{(i)} - G^{(i)}\big)\, \nabla_\psi Q_\psi^{(i)}.$$

Local gradient factorization under weight sharing. Let w be any scalar entry of ψ. By the chain rule,

$$g_i(w) \;\triangleq\; \frac{\partial L_i}{\partial w} = \underbrace{\big(Q_\psi^{(i)} - G^{(i)}\big)}_{\varepsilon^{(i)}}\, \underbrace{\frac{\partial Q_\psi^{(i)}}{\partial w}}_{\psi^{(i)}(w)}.$$

For linear modules (affine maps in attention/FFN), the Jacobian has the standard local form $\frac{\partial Q_\psi^{(i)}}{\partial w} = a^{(i)}\,\delta^{(i)}$ (input activation × upstream error). Because the same w is *shared* across decoder positions, the sequence $\{g_i(w)\}_{i=1}^{n}$ are n gradient contributions for the *same* parameter, drawn from adjacent positions of one forward pass, and are therefore generally *positively correlated*.

A convenient covariance model (exchangeable/equicorrelated). For fixed w, we use the standard homoscedastic equicorrelation approximation (also common in mini-batch analyses):

$$\mathrm{Var}[g_i(w)] = \sigma_w^2, \qquad \mathrm{Cov}(g_i(w), g_j(w)) = \rho_w\, \sigma_w^2 \quad (i \neq j), \qquad \rho_w \in [0, 1).$$

This captures the empirically relevant regime where adjacent horizons produce positively correlated gradients and yields tight, closed-form variance expressions.

Lemma 3 (Variance of the averaged per-parameter update)**.** *With the model above, the averaged update $\bar{g}(w) \triangleq \frac{1}{n}\sum_{i=1}^{n} g_i(w)$ satisfies*

$$\mathrm{Var}[\bar{g}(w)] = \frac{1}{n^2}\left(\sum_{i=1}^{n} \mathrm{Var}[g_i] + \sum_{i\neq j} \mathrm{Cov}(g_i, g_j)\right) = \sigma_w^2\, \frac{1 + (n-1)\rho_w}{n}.$$

In particular, $\mathrm{Var}[\bar{g}(w)] < \sigma_w^2$ *for any* $\rho_w < 1$.

Corollary 3 (Effective batch size and asymptotics)**.** *Define the* effective sample size $n_{\text{eff}}(w) \triangleq \dfrac{n}{1 + (n-1)\rho_w}$. *Then* $\mathrm{Var}[\bar{g}(w)] = \sigma_w^2 / n_{\text{eff}}(w)$ *with* $1 \leq n_{\text{eff}}(w) \leq n$*, strictly increasing in* n*, and* $\lim_{n\to\infty} \mathrm{Var}[\bar{g}(w)] = \rho_w\, \sigma_w^2$ *(the correlation-imposed variance floor).*

Proposition 4 (Uniform horizon averaging is optimal under exchangeability)**.** *Among all unbiased linear combinations $\sum_{i=1}^{n} \alpha_i g_i(w)$ with $\sum_i \alpha_i = 1$, the variance is minimized by the* uniform *weights $\alpha_i = \frac{1}{n}$ whenever $\mathrm{Cov}(g_i, g_j)$ is exchangeable (same diagonal/off-diagonal).*

Proof. For an exchangeable covariance $\Sigma_w = \sigma_w^2[(1-\rho_w)I + \rho_w \mathbf{1}\mathbf{1}^T]$, $\mathrm{Var}(\sum_i \alpha_i g_i) = \alpha^T \Sigma_w \alpha$ is minimized under $\mathbf{1}^T\alpha = 1$ by $\alpha^\star = \frac{1}{n}\mathbf{1}$. □

Why $\rho_w \gtrsim 0$ is natural. Both multiplicative factors of $g_i(w)$ vary smoothly with i: (i) the targets $G^{(i)}$ share overlapping reward sums and a common bootstrapped tail; and (ii) the Jacobians $\partial Q_\psi^{(i)}/\partial w$ come from *adjacent* decoder positions of the same Transformer. This induces positive correlation among $\{g_i(w)\}$, putting us squarely in the regime of Lemma 3.

Connection to target-side variance (discounted rewards and bootstrap). Let $\gamma \in (0, 1]$ be the discount. Write the single N-step reward sum and its triangular average as

$$R_N(\gamma) = \sum_{k=0}^{N-1} \gamma^k r_k, \qquad \bar{R}_N(\gamma) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{i-1} \gamma^k r_k = \sum_{k=0}^{N-1} \underbrace{\frac{N-k}{N} \gamma^k}_{w_k} r_k.$$

Under the equicorrelated reward model (mean μ, variance σ^2, pairwise corr. $\rho \geq 0$),

$$\mathrm{Var}\big[R_N(\gamma)\big] = \sigma^2 \big[S_0 + \rho\,(T_0^2 - S_0)\big]\,, \qquad \mathrm{Var}\big[\bar{R}_N(\gamma)\big] = \sigma^2 [A_\gamma + \rho\, B_\gamma]\,,$$

with

$$S_0 = \sum_{k=0}^{N-1} \gamma^{2k} = \frac{1-\gamma^{2N}}{1-\gamma^2}, \quad T_0 = \sum_{k=0}^{N-1} \gamma^k = \frac{1-\gamma^N}{1-\gamma}, \quad A_\gamma = \sum_{k=0}^{N-1} w_k^2, \quad B_\gamma = \Big(\sum_{k=0}^{N-1} w_k\Big)^2 - A_\gamma.$$

The reward-side ratio

$$R_\gamma(N) \;\triangleq\; \frac{\mathrm{Var}[R_N(\gamma)]}{\mathrm{Var}[\bar{R}_N(\gamma)]} = \frac{S_0 + \rho\,(T_0^2 - S_0)}{A_\gamma + \rho B_\gamma}$$

satisfies the *uniform bound* $1 \leq R_\gamma(N) < 4$ for all $N \geq 1$, $\rho \geq 0$, $\gamma \in (0, 1]$, and $R_\gamma(N) \nearrow 4$ as $N \to \infty$.

For the bootstrapped values $Z_i = V_{\phi_{\mathrm{tar}}}(s_{t+i})$ with $\mathrm{Var}(Z_i) = \tau^2$ and $\mathrm{Cov}(Z_i, Z_j) = \kappa\,\tau^2$ $(i \neq j,\ \kappa \in [0, 1])$, define

$$B_N(\gamma) = \gamma^N Z_N, \qquad \bar{B}_N(\gamma) = \frac{1}{N} \sum_{i=1}^{N} \gamma^i Z_i,$$

and let $S_1 = \sum_{i=1}^{N} \gamma^{2i}$, $C = \sum_{i=1}^{N} \gamma^i$. Then

$$\mathrm{Var}[\bar{B}_N(\gamma)] = \frac{\tau^2}{N^2} \big[S_1 + \kappa(C^2 - S_1)\big], \qquad R_B(N, \gamma, \kappa) \triangleq \frac{\mathrm{Var}[B_N(\gamma)]}{\mathrm{Var}[\bar{B}_N(\gamma)]} = \frac{N^2 \gamma^{2N}}{S_1 + \kappa(C^2 - S_1)}.$$

R_B is decreasing in κ and obeys

$$\frac{N^2 \gamma^{2N}}{C^2} \;\leq\; R_B(N, \gamma, \kappa) \;\leq\; \frac{N^2 \gamma^{2N}}{S_1}.$$

In particular, averaging the bootstrap part reduces variance whenever $\kappa \leq \kappa_\star(N, \gamma) \triangleq \dfrac{N^2 \gamma^{2N} - S_1}{C^2 - S_1}$.

Theorem 5 (Main: gradient averaging reduces update variance; compounded by target-side smoothing)**.** *Let w be any scalar parameter of the shared-weights Transformer critic and suppose $\{g_i(w)\}_{i=1}^n$ are homoscedastic and equicorrelated with $\rho_w < 1$. Then*

$$\mathrm{Var}\left[\frac{\partial \bar{L}}{\partial w}\right] = \sigma_w^2\, \frac{1 + (n-1)\rho_w}{n} < \sigma_w^2 = \mathrm{Var}\left[\frac{\partial L_j}{\partial w}\right], \quad \forall j \in \{1, \ldots, n\}.$$

Moreover, writing $g_i(w) = \varepsilon^{(i)} \psi^{(i)}(w)$ and (mildly) assuming $\{\varepsilon^{(i)}\}$ and $\{\psi^{(i)}(w)\}$ are independent across i with bounded second moments, there exist constants $a_w, b_w \geq 0$ (depending only on ψ) such that

$$\mathrm{Var}[g_i(w)] \;\leq\; a_w\, \mathrm{Var}[G^{(i)}] \;+\; b_w.$$

Consequently, replacing a single horizon by the triangularly averaged target across horizons $1{:}N$ reduces the reward-side variance by at least a factor $R_\gamma(N)^{-1} \in (1/4, 1]$, and (when $\kappa \leq \kappa_\star$) also reduces the bootstrap-side variance by a factor $R_B(N, \gamma, \kappa)^{-1}$. Thus, in addition to the across-horizon gradient averaging *gain $\frac{1+(n-1)\rho_w}{n}$, the per-horizon variance term σ_w^2 itself decreases with N via target-side smoothing, yielding a compounded reduction.*

Practical notes. (i) The gradient-level algebra is agnostic to discount γ; only the target-side constants $(S_0, T_0, A_\gamma, B_\gamma)$ and (S_1, C) change with γ. (ii) Under exchangeability, uniform averaging across horizons is *variance-optimal* (Prop. 4); no learned horizon-weights are needed for variance reasons. (iii) As n grows, the residual variance floor is $\rho_w\, \sigma_w^2$ (Cor. 3); lower temporal correlation between horizon-gradients directly improves this floor. (iv) If horizon-gradients are not perfectly exchangeable, the bound $\mathrm{Var}[\bar{g}(w)] \leq \dfrac{\bar{\sigma}_w^2}{n}\big(1 + (n-1)\bar{\rho}_w\big)$ still holds whenever $\mathrm{Var}[g_i] \leq \bar{\sigma}_w^2$ and $\mathrm{Corr}(g_i, g_j) \leq \bar{\rho}_w$ for all $i \neq j$.

E Appendix: Connection to Multi-step TD Theory

Equations 5–6 can be viewed as a standard multi-step TD update in an MDP where each action prefix $a_{t:t+i-1}$ is treated as an extended action. For a fixed horizon i, we define the extended state–action pair

$$x = (s_t, a_{t:t+i-1}),$$

use equation 5 as the N-step target $G^{(i)}(x)$, and minimize the squared TD error

$$\left(Q_\psi(x) - G^{(i)}(x)\right)^2,$$

exactly as in classical N-step Q-learning / multi-step TD on this augmented MDP, with a shared-parameters Transformer implementing Q_ψ for all such prefixes and horizons.

When the behavior policy matches the target policy ($\mu = \pi_\theta$), the replay distribution coincides with the on-policy evaluation distribution, the implicit IS ratios are all 1, and $G^{(i)}$ reduces to the standard on-policy i-step return (Sutton et al., 1998; Precup et al., 2000). In the tabular setting, this yields exactly classical on-policy multi-step TD / TD(λ), for which convergence to Q^π is well understood; with linear function approximation and suitable step sizes, one recovers convergence to the unique projected fixed point of the TD operator under the on-policy distribution.

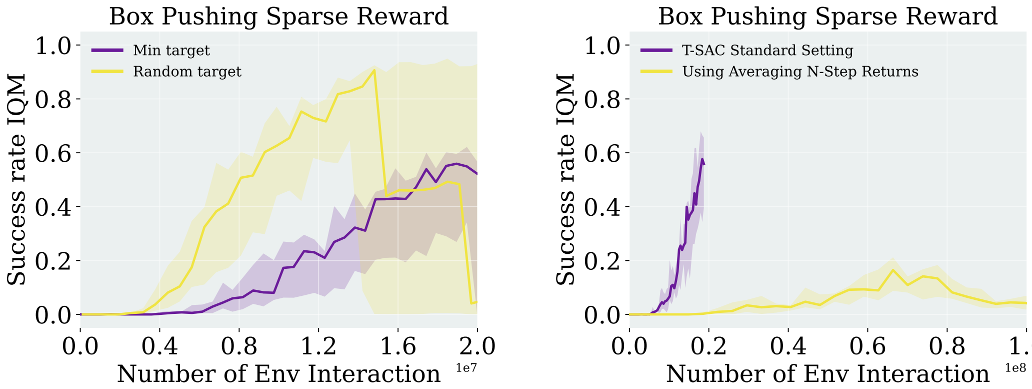
In the off-policy case ($\mu \neq \pi_\theta$) and *without* importance weights, our critic update falls into the "uncorrected off-policy TD" regime analyzed by Munos et al. (2016). In this regime, multi-step TD converges (under suitable assumptions) to the fixed point of a projected Bellman operator defined with respect to the behavior distribution, yielding a bias relative to Q^π but admitting error-propagation bounds that relate this bias to distribution mismatch and approximation error. Our choice to learn values of realized prefixes

$$Q_\psi(s_t, a_{t:t+i-1})$$

under μ is precisely an instance of this uncorrected regime, with the benefit that we avoid the high variance associated with long-horizon IS products (Precup et al., 2000).

The key difference to standard off-policy N-step TD with IS is therefore *what* the critic is asked to predict. IS-corrected targets are (in principle) unbiased for Q^π, but their variance scales poorly with the length of the IS product and typically requires aggressive clipping when behavior and target policies differ. Our critic instead learns the value of realized prefixes under the replay distribution, trading some asymptotic bias for substantially reduced variance and improved numerical stability.

From a theoretical perspective, conditioned on a given state s_t and realized prefix $a_{t:t+i-1}$, the distribution over future rewards is fully determined by the environment dynamics and the continuation policy, and does not depend on whether this prefix was generated by the behavior or target policy. This viewpoint underpins our sequence-conditioned critic and helps explain the empirically observed stability of long-horizon learning.



(a) Effect of target-value construction under sparse rewards. Randomly selecting one of two double-Q targets leads to high-variance updates and occasional collapse, whereas the conservative minimum-based target yields stable learning curve with the hard–copy critic.

(b) Naively averaging N-step targets across horizons destabilizes learning and can erase progress, confirming the need for gradient–level averaging.

Figure 9: Meta-World Box Pushing (Sparse Reward). Ablations on (a) target-value construction and (b) return-propagation schemes for T–SAC.

F Appendix: Additional Experimental Results

Post-figure summary. Figure 9a shows that the instability originally observed on Box–Pushing–Sparse is explained by the high–variance target estimator: with the conservative minimum–based target, hard–copy T–SAC is stable and reaches the best success rates. Figure 9b further illustrates that naive N-step target averaging can derail optimization, motivating our choice of gradient–averaged multi–horizon losses. **Seeds: 4. Results under IQM with 95% confidence intervals.**

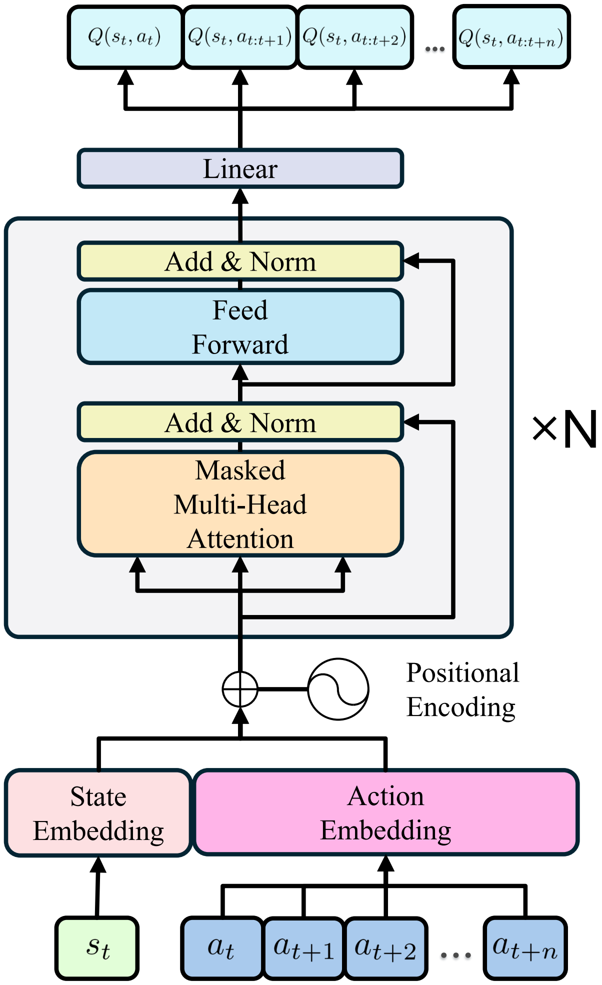
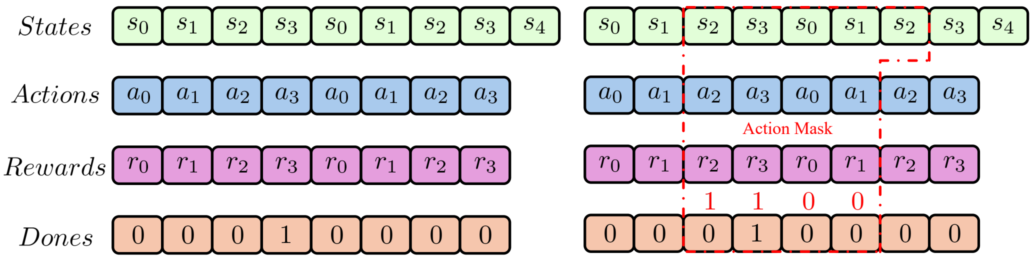


Figure 10: T-SAC Critic Detailed Structure: a causal Transformer over short state–action segments. Given $(s_t, a_t, \ldots, a_{t+n})$, the network produces n scalar outputs $\{Q_\psi(s_t, a_t, \ldots, a_{t+i})\}_{i=1}^{n}$. Colors and block styling follow the Transformer diagram conventions of Vaswani et al. (2017).

G APPENDIX: TRANSFORMER CRITIC DETAILED STRUCTURE

Implementation details. We follow the TOP–ERL–style Transformer critic design adopted in this work (see Li et al. (2024a) for the schematic), i.e., a masked multi–head self–attention stack with positional encodings and residual Add&Norm blocks; the critic ingests $(s_t, a_t, \ldots, a_{t+n})$ and jointly predicts all $1 \ldots n$ step returns. State and action tokens use *separate* one–layer linear embeddings (no bias), consistent with our training objective that conditions on realized action prefixes; the output head is a linear map without bias that emits one scalar per decoder position. No dropout is used anywhere in the critic. The causal mask ensures each position i only attends to $\leq i$ tokens, aligning the network outputs with the i-step targets used for learning.



(a) Fixed-length segments stored in the replay buffer. Each sample is a window $(s_{t:t+L}, a_{t:t+L-1}, r_{t:t+L-1}, d_{t:t+L-1})$. If a terminal $d_\tau = 1$ appears before the window is full, we immediately continue saving from the start of the next trajectory until the segment length L is reached.

(b) Action mask for a sampled segment. The binary mask m marks valid action positions: $m_\tau = 1$ for steps before or at the first terminal in the window and $m_\tau = 0$ after any $d_\tau = 1$. Entries that occur after a terminal (including those filled from a new trajectory) are masked out so losses/targets and attention never cross episode boundaries.

Figure 11: T-SAC segment construction and masking.

H Appendix: Trajectories saved in Replay Buffer and Action Mask Design

The benefit of chunking on the critic side

We analyze chunking under sparse rewards and target–value reuse.

Setup. Consider a fixed segment of N transitions $(s_t, a_t, r_t, \dots, s_{t+N})$. A *chunked* sample is obtained by: (i) drawing a start index p uniformly from $\{t+1, \dots, t+N\}$; (ii) drawing a window length L uniformly from the integers $\{\ell_{\min}, \dots, \ell_{\max}\}$ (with $\ell_{\min} \geq 1$); and defining the window end

$$q \;=\; \min\{\, p + L - 1,\; t+N \,\}.$$

For every $i \in \{p, \dots, q-1\}$ we form the truncated multi-step target that bootstraps at q:

$$G_i \;=\; \sum_{j=i}^{q-1} \gamma^{\,j-i} r_j \;+\; \gamma^{\,q-i} V_\phi(s_q). \tag{13}$$

Hence the *same* value $V_\phi(s_q)$ is reused across the $(q-p)$ TD updates inside the window.

How often is a particular value reused? Write $\Delta = \ell_{\max} - \ell_{\min} + 1$, and (for clarity) index the segment states by $1, \dots, N$. Let reuse_j denote the number of updates in a sampled window whose target bootstraps at $V_\phi(s_j)$ (i.e., with $q = j$). Then, for all interior states $j < N$,

$$\mathbb{E}[\text{reuse}_j] = \frac{1}{N\,\Delta} \sum_{k=\ell_{\min}-1}^{\min(\ell_{\max}-1,\, j-1)} k = \begin{cases} 0, & j < \ell_{\min}, \\[4pt] \dfrac{(j - \ell_{\min} + 1)(j + \ell_{\min} - 2)}{2N\,\Delta}, & \ell_{\min} \leq j < \ell_{\max}, \\[4pt] \dfrac{\ell_{\min} + \ell_{\max} - 2}{2N}, & \ell_{\max} \leq j \leq N-1. \end{cases} \tag{14}$$

Thus, away from the left boundary, the expected reuse $\mathbb{E}[\text{reuse}_j]$ *plateaus* at $\frac{\ell_{\min}+\ell_{\max}-2}{2N}$ for all $j \in [\ell_{\max},\, N-1]$.

The right boundary $j = N$ is special because of truncation ($q = \min\{p+L-1, N\}$). In this case,

$$\mathbb{E}[\text{reuse}_N] = \frac{1}{N} \sum_{h=1}^{\ell_{\min}} (h-1) \;+\; \frac{1}{N\,\Delta} \sum_{h=\ell_{\min}+1}^{\ell_{\max}} (h-1)(\ell_{\max} - h + 1), \tag{15}$$

which exceeds the interior plateau and concentrates more bootstrap reuse at the end of the segment. A transparent special case is $\ell_{\min} = 1$:

$$\mathbb{E}[\text{reuse}_N] \;=\; \frac{\ell_{\max}^2 - 1}{6N}. \tag{16}$$

Averaging equation 14 across all j yields the compact relation

$$\frac{1}{N}\sum_{j=1}^{N}\mathbb{E}[\text{reuse}_j] \;=\; \frac{\mathbb{E}[L]-1}{N}, \tag{17}$$

i.e., per sampled window the expected reuse scales linearly with the average window length.

Connection to state coverage (selection) probability. The probability that a given state j is *covered* by the sampled window (i.e., $j \in [p,\ q]$) is

$$\Pr(j \text{ covered}) = \frac{1}{N\,\Delta}\sum_{p=1}^{j}\big[\ell_{\max} - \max\{\ell_{\min},\, j-p+1\} + 1\big]_{+}, \tag{18}$$

where $[\cdot]_+ = \max\{\,\cdot, 0\,\}$. For the common case $\ell_{\min} = 1$ and writing $m = \ell_{\max}$, this simplifies to

$$\Pr(j \text{ covered}) = \begin{cases} \dfrac{1}{N}\left(j - \dfrac{j(j-1)}{2m}\right), & 1 \le j \le m, \\ \dfrac{m+1}{2N}, & m < j \le N, \end{cases} \tag{19}$$

i.e., a ramp near the start followed by a flat plateau. This higher coverage (vs. 1-step sampling) underlies the critic-side gains below.

Sparse rewards: how far does a single reward propagate? Assume only the terminal transition carries non-zero reward (the sparse-reward setting). An update's target contains that reward iff the sampled window reaches the segment end ($q = N$), in which case *all* $(N-p)$ updates inside the window include it. Therefore, the expected number of reward-bearing updates per sampled window equals

$$\mathbb{E}\big[\#\text{updates including terminal reward}\big] = \frac{1}{N}\sum_{h=1}^{\ell_{\min}}(h-1) \;+\; \frac{1}{N\,\Delta}\sum_{h=\ell_{\min}+1}^{\ell_{\max}}(h-1)(\ell_{\max}-h+1), \tag{20}$$

which coincides with equation 15. In particular, for $\ell_{\min} = 1$,

$$\mathbb{E}\big[\#\text{updates including terminal reward}\big] \;=\; \frac{\ell_{\max}^2 - 1}{6N}, \tag{21}$$

representing a $\approx \ell_{\max}^2/6$-fold amplification over uniform 1-step TD (which touches the terminal reward only in the single $(N-1) \to N$ update, i.e., $1/N$ of samples).

Takeaways. Chunking yields two critic-side benefits: (i) *Target–value reuse:* each sampled window reuses a single bootstrap $V_\phi(s_q)$ across $\mathbb{E}[\text{reuse}_j]$ updates, reaching a plateau of $\frac{\ell_{\min}+\ell_{\max}-2}{2N}$ for interior states and an even larger value at the terminal state due to truncation equation 15. This may help explain why, in our setting, training remains stable even *without* a target network in locomotion tasks.
(ii) *Sparse-reward propagation:* when only the last transition is rewarded, chunking increases—often quadratically in $\ell_{\max}$ when $\ell_{\min} = 1$—the share of updates that incorporate the true reward, substantially shortening effective credit-assignment horizons. This mechanism helps explain why T-SAC performs well under sparse-reward settings.

I Appendix: Computational costs and Sample Efficiency

Training time is reported for 1M environment steps (UTD= 1), unless otherwise stated. All benchmarks were run on an NVIDIA A100 (40 GB) GPU and an Intel Xeon Platinum 8368 CPU.

T -SAC: MLP policy with two 256-unit hidden layers; Transformer critic with 2 layers × 256 units.
Gru/Lstm: same policy; 2-layer RNN critic (256 units).
Sac and CrossQ: default configurations.

Table 1: Sample efficiency on long-horizon benchmarks, measured as the number of environment steps (in millions) required to reach a fixed performance threshold on each task. Thresholds are defined as 90% of SAC's final return on Box-Pushing (dense) and ML1, and 90% of T-SAC's final return on Box-Pushing (sparse). Lower is better. Values are means over seeds.

Task	SAC	CrossQ	GTrXL policy	TOP-ERL	T-SAC (ours)
Box-Pushing (dense)	15M	10M	20M	2M	4M
Box-Pushing (sparse)	N.A.	N.A.	N.A.	4M	17M
ML1	4M	N.A.	N.A.	4M	1M

Table 2: Effect of minimum and maximum sequence length on T-SAC performance and wall-clock training time on *Box-Pushing (dense)*. All runs use the same number of environment steps 1 M for standard setting.

min_length	max_length	Return (mean ± s.e.)	Wall-clock time	Peak GPU memory (GB)
1	1	-78.45 ± 6.89	2h35m03s	2.37
4	4	-66.63 ± 3.09	2h39m23s	2.37
1	4	-74.80 ± 7.69	2h38m56s	2.37
1	16	-65.05 ± 0.20	3h06m58s	2.37

Table 3: Computational cost comparison for different methods on *Box-Pushing (dense)* for a fixed number of environment steps and matched (or explicitly stated) update-to-data (UTD) ratios.

Method	Params (M)	UTD	Wall-clock time (hours)
SAC	0.2	1	1.47
CrossQ	10	1	2.44
T-SAC (ours)	3.3	1/4	0.77
TOP-ERL	3.3	1/10	0.35

Table 4: Performance at fixed data budgets on long-horizon tasks. Entries are success rate ± standard error over seeds after a given number of environment steps. All methods are trained with the same number of transitions.

Task	Method	100k steps	300k steps	500k steps	1M steps
Box-Pushing (dense)	SAC	0.0 ± 0.1	0.05 ± 0.12	0.05 ± 0.15	0.01 ± 0.2
	CrossQ	0.0 ± 0.0	0.0 ± 0.0	0.01 ± 0.01	0.07 ± 0.02
	GTrXL policy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.3 ± 0.01
	T-SAC (ours)	0.0 ± 0.0	0.05 ± 0.02	0.2 ± 0.01	0.8 ± 0.08
Box-Pushing (sparse)	SAC	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	CrossQ	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	GTrXL policy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	T-SAC (ours)	0.0 ± 0.0	0.0 ± 0.0	0.01 ± 0.01	0.1 ± 0.05
ML1	SAC	0.1 ± 0.01	0.35 ± 0.02	0.42 ± 0.02	0.5 ± 0.05
	CrossQ	0.12 ± 0.01	0.37 ± 0.02	0.5 ± 0.05	0.5 ± 0.08
	GTrXL policy	0.05 ± 0.01	0.07 ± 0.02	0.1 ± 0.02	0.28 ± 0.02
	T-SAC (ours)	0.1 ± 0.01	0.35 ± 0.05	0.41 ± 0.08	0.58 ± 0.12

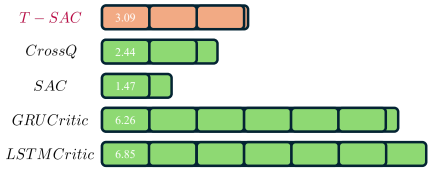


Figure 12: Training time is reported for 1M environment steps (UTD= 1), unless otherwise stated. All benchmarks were run on an NVIDIA A100 (40 GB) GPU and an Intel Xeon Platinum 8368 CPU.

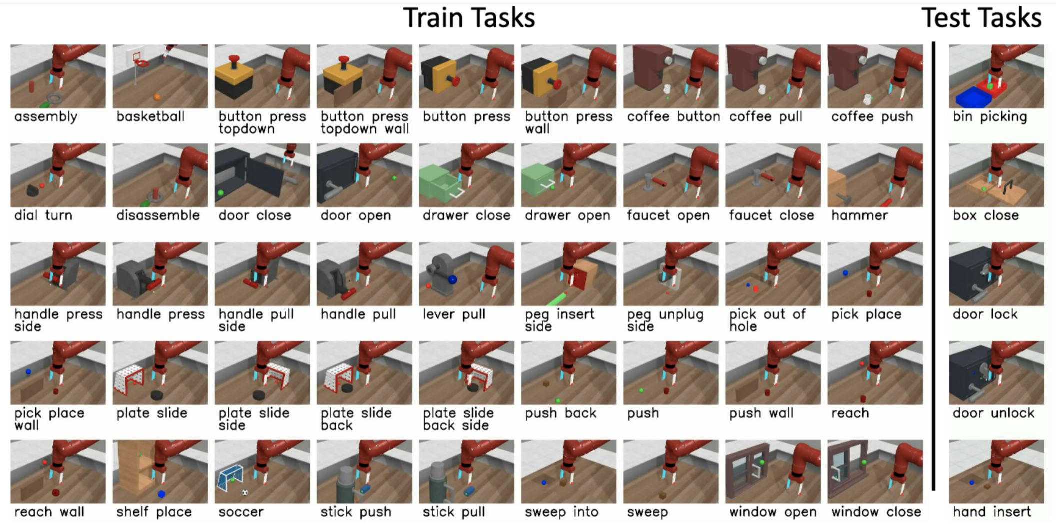


Figure 13: Meta-World tasks (Yu et al., 2020).

J Experiment Description

J.1 Meta-World ML1

Meta-World (Yu et al., 2020) is an open-source simulated benchmark for meta-reinforcement learning and multi-task learning in robotic manipulation. It comprises 50 distinct tasks spanning skills such as grasping, pushing, and object placement, each posing different perception–control challenges. By covering a broader skill spectrum than narrowly scoped benchmarks, Meta-World is well-suited for evaluating algorithms that aim to generalize across diverse behaviors. Figure 13 enumerates all 50 tasks and illustrates their variety and difficulty.

Success criterion. To better approximate real-world deployment, we adopt a stringent evaluation rule: an episode is counted as successful only if the environment's success condition is satisfied at the *final* timestep; intermediate achievements do not count toward success.

J.2 Box Pushing

Setup. A 7-DoF Franka Emika Panda arm with a rod pushes a box on a table to a target pose. At episode start, initial and target box poses are sampled with a minimum 0.2 m separation:

$$x_i \in [0.3, 0.6], \quad y_i \in [-0.4, 0.4], \quad \theta \in [0, 2\pi].$$

Success (for evaluation) is position error $\leq$ 0.05 m and orientation error $\leq$ 0.5 rad.

Observations & Actions. Observations: robot joint positions/velocities $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, box position/orientation $(\boldsymbol{p}, \boldsymbol{r})$, and target $(\boldsymbol{p}_{\text{target}}, \boldsymbol{r}_{\text{target}})$. Actions: joint torques $\boldsymbol{a}_t \in \mathbb{R}^7$.

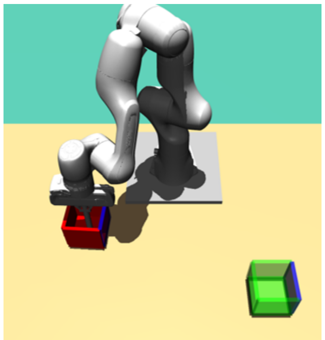
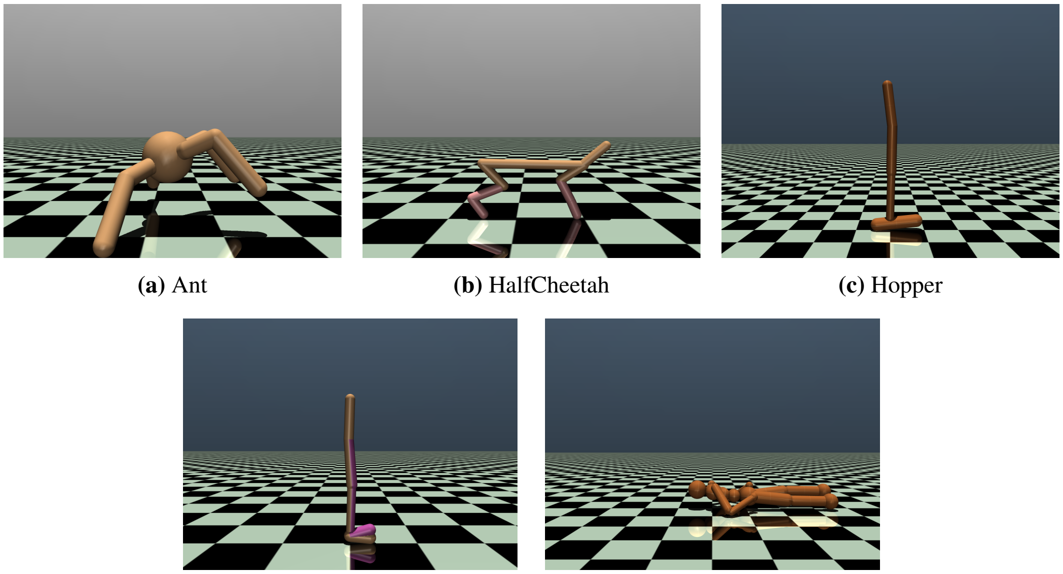


Figure 14: Box Pushing task (Otto et al.).



(a) Ant

(b) HalfCheetah

(c) Hopper

(d) Walker2d

(e) HumanoidStandup

Figure 15: MuJoCo (Towers et al., 2024) tasks used in our experiments.

Termination. Fixed horizon $T = 100$ steps; no early termination.

Dense reward. At each step,

$$R_{\text{total}} = -R_{\text{rod}} - 0.02\,\tau_t - \text{err}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - 350\,R_{\text{position}} - 200\,R_{\text{rotation}}.$$

Subterms are

$$R_{\text{rod}} = \text{clip}(\|\boldsymbol{p} - \boldsymbol{h}_{\text{pos}}\|,\, 0.05,\, 10) + \text{clip}\big(\tfrac{2}{\pi}\arccos|\boldsymbol{h}_{\text{rot}}\cdot\boldsymbol{h}_0|\,,\, 0.25,\, 2\big)\,, \tag{22}$$

$$\tau_t = \sum_{i=1}^{7}(a_t^i)^2, \tag{23}$$

$$\text{err}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_{i:\,|q_i|>|q_i^b|}\big(|q_i| - |q_i^b|\big) + \sum_{j:\,|\dot{q}_j|>|\dot{q}_j^b|}\big(|\dot{q}_j| - |\dot{q}_j^b|\big)\,, \tag{24}$$

$$R_{\text{position}} = \|\boldsymbol{p} - \boldsymbol{p}_{\text{target}}\|, \tag{25}$$

$$R_{\text{rotation}} = \tfrac{1}{\pi}\arccos|\boldsymbol{r}\cdot\boldsymbol{r}_{\text{target}}|\,. \tag{26}$$

Here, $\boldsymbol{h}_{\text{pos}}$ is the rod tip position, and $\boldsymbol{h}_{\text{rot}}, \boldsymbol{h}_0$ are rod orientations (quaternions).

Sparse reward. Only the task terms are applied at the final step:

$$R_{\text{total}} = \begin{cases} -R_{\text{rod}} - 0.02\,\tau_t - \text{err}(\boldsymbol{q}, \dot{\boldsymbol{q}}), & t < T, \\ -R_{\text{rod}} - 0.02\,\tau_t - \text{err}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - 350\,R_{\text{position}} - 200\,R_{\text{rotation}}, & t = T. \end{cases} \tag{27}$$

J.3 Gymnasium MuJoCo

We evaluate on the Gymnasium MuJoCo **v4** suite—`Ant-v4`, `HalfCheetah-v4`, `Hopper-v4`, `Walker2d-v4`, and `HumanoidStandup-v4` (Fig. 15). We use the default observation and action spaces and the native v4 reward shaping and termination rules (no reward normalization). Performance is reported as undiscounted episode return. Unless noted otherwise, evaluation uses the deterministic policy over 152 episodes and aggregates results across multiple random seeds using the IQM with 95% bootstrap confidence intervals; full hyperparameters and seeds are provided in App. L.

K Appendix: Algorithm Implementations

PPO Proximal Policy Optimization (PPO) (Schulman et al., 2017) is an on–policy, step–based method that constrains policy updates to remain close to the behavior policy. Two variants are common: *PPO–Penalty* (KL regularization) and *PPO–Clip* (clipped surrogate). We evaluate PPO–Clip given its prevalence and robustness, following the reference implementation in Raffin et al. (2021). **Seeds: 20.**

SAC Soft Actor–Critic (SAC) (Haarnoja et al., 2018a;b) is an off–policy actor–critic with twin Q–networks to mitigate overestimation and an entropy term to encourage exploration. We use the Bhatt et al. (2019) implementation, which includes SAC. **Seeds: 20 (Meta–World ML1), 5 (Gym MuJoCo).**

TD3 Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) addresses overestimation and instability via (i) clipped double Q–learning, (ii) delayed policy updates, and (iii) target policy smoothing. Our TD3 follows standard practice adapted from Raffin et al. (2021), including Polyak averaging and action noise for exploration. **Seeds: 5.**

GTrXL Gated Transformer–XL (GTrXL) (Parisotto et al., 2020) stabilizes Transformer training for partially observable control. We build on the PPO + GTrXL implementation from Liang et al. (2018) and add minibatch advantage normalization plus a state–independent log–standard–deviation head following Huang et al. (2022). **Seeds: 4.**

gSDE Generalized State–Dependent Exploration (gSDE) (Raffin et al., 2022; Rückstieß et al., 2008; Rückstiess et al., 2010) replaces i.i.d. Gaussian action noise with state–dependent, temporally smooth exploration. Concretely, disturbances are generated as $\epsilon_t = \Theta s$, where s is the last hidden layer's activation and Θ is resampled from a Gaussian every n steps according to the SDE sampling frequency. We evaluate gSDE with PPO using the reference implementation of Raffin et al. (2022); for stability on some tasks we employ a linear schedule for the PPO clipping range. **Seeds: 20.**

BBRL Black–Box Reinforcement Learning (BBRL) (Otto et al., 2023a;b) performs episodic, trajectory–level search by parameterizing policies with ProMPs (Paraschos et al., 2013). This handles sparse and non–Markovian rewards but can reduce sample efficiency. We consider both diagonal–covariance (**BBRL–Std**) and full–covariance (**BBRL–Cov**) Gaussian policies, paired with ProDMP (Li et al., 2023). **Seeds: 20.**

TCP Temporally–Correlated Episodic RL (TCP) (Li et al., 2024b) augments episodic policy updates with step–level signals, narrowing the gap between episodic and step–based RL while retaining smooth, parameter–space exploration. **Seeds: 20.**

TOP–ERL Trajectory–Optimized Policy for Episodic RL (TOP–ERL) optimizes a distribution over motion–primitive parameters with (i) a KL–constrained trust region and (ii) a temporally structured covariance that induces smooth, correlated exploration across the episode. Our instantiation uses ProDMP (Li et al., 2023) as the trajectory generator; unless stated, we adopt an adaptive scale (entropy) schedule and per–dimension normalization of primitive parameters. **Seeds: 8.**

CrossQ CrossQ (Bhatt et al., 2019) is an off–policy SAC variant that removes target networks and applies BRN in the critic, enabling strong sample efficiency at an update–to–data ratio of $\mathrm{UTD} = 1$. We follow the authors' reference implementation: a single batch–normalized critic (no target networks), default temperature tuning, and recommended hyperparameters unless stated otherwise. **Seeds: 4 (Meta–World ML1), 5 (Gym MuJoCo and Box–Pushing).** Training on Box–Pushing was capped at 10 M steps due to the experiment budget; by that point, wall–clock time exceeded 24 h.

L Appendix: Hyperparameters of the algorithms

Baseline provenance. For **BBRL**, **TCP**, **PPO**, **gSDE**, **GTrXL**, **TOP–ERL**, and **SAC** on Meta–World ML1, we report numbers from prior publications and/or official released runs/configurations under settings comparable to ours; we did not perform additional large–scale sweeps for these baselines in this paper (see citations in the main text and Appendix K).

Methods tuned in this work. We tuned **SAC** on Gym/FancyGym, the full **CrossQ** implementation, and **TD3**, including optimizer selection and hyperparameters (e.g., learning rates).

Our tuning for T–SAC. For **T–SAC**, we conducted a targeted grid search over Transformer–critic depth (number of attention layers), number of heads, dimensions per head, learning rates (policy/critic/α), supervision–window settings (fixed vs. variable horizons; `min_length`, `step_length`), number of per–step target windows, and policy–side chunk length (for the compatibility study). Where appropriate, we initialized choices from publicly reported configurations: Transformer hyperparameter ranges from TOP–ERL (Li et al., 2024a), the entropy–temperature term from Celik et al. (2025), and the optimizer family from CrossQ (Bhatt et al., 2019). Final settings and search grids are listed in the appendix tables.

Table 5: Hyperparameters for the Meta-World experiments. Episode Length $T = 500$

	PPO	gSDE	GTrXL	SAC	CrossQ	TCP	BBRL	TOP-ERL	T-SAC
number samples	16000	16000	19000	1000	1	16	16	2	4 * 125
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.	n.a.	n.a.
discount factor	0.99	0.99	0.99	0.99	0.99	1	1	1.0	0.99
ϵ_μ	n.a.	n.a.	n.a.	n.a.	n.a.	0.005	0.005	0.005	n.a.
ϵ_Σ	n.a.	n.a.	n.a.	n.a.	n.a.	0.0005	0.0005	0.0005	n.a.
trust region loss coef.	n.a.	n.a.	n.a.	n.a.	n.a.	1	10	1.0	n.a.
optimizer	adam	adam	adam	adam	adam	adam	adam	adam	adamw
epochs	10	10	5	1000	1	50	100	15	20
learning rate	3e-4	1e-3	2e-4	3e-4	3e-4	3e-4	3e-4	1e-3	2.5e-4
use critic	True	True	True	True	True	True	True	True	True
epochs critic	10	10	5	1000	1	50	100	50	100
learning rate critic	3e-4	1e-3	2e-4	3e-4	3e-4	3e-4	3e-4	5e-5	2.5e-5
number minibatches	32	n.a.	n.a	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	500	1024	256	256	n.a.	n.a.	256	512
buffer size	n.a.	n.a.	n.a.	1e6	1e6	n.a.	n.a.	3000	5000 * 125
learning starts	0	0	n.a.	10000	5000	0	0	2	200
temperature warmup	0	0	0	0	0	0	0	0	10000
polyak_weight	n.a.	n.a.	n.a.	5e-3	1.0	n.a.	n.a.	5e-3	5e-3
SDE sampling frequency	n.a.	4	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0	0	auto	auto	0	0	n.a.	auto
normalized observations	True	True	False	False	False	True	False	False	False
normalized rewards	True	True	0.05	False	False	False	False	False	False
observation clip	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	lin_0.3	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	lin_0.3	0.1	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[128, 128]	[128, 128]	n.a.	[256, 256]	[256, 256]	[128, 128]	[32, 32]	[128, 128]	[128, 128]
hidden layers critic	[128, 128]	[128, 128]	n.a.	[256, 256]	[2048, 2048]	[128, 128]	[32, 32]	n.a.	n.a.
hidden activation	tanh	tanh	relu	relu	relu	relu	relu	leaky_relu	leaky_relu
orthogonal initialization	Yes	No	xavier	fanin	fanin	Yes	Yes	Yes	fanin
initial std	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0
number of heads	-	-	4	-	-	-	-	8	4
dims per head	-	-	16	-	-	-	-	16	32
number of attention layers	-	-	4	-	-	-	-	2	2

Task-specific settings (Meta-World). For **T–SAC**, we initialize the policy's log standard deviation as $\log \sigma = -5$. The replay buffer stores $5{,}000$ segments of length 125 (i.e., $5{,}000 \times 125 = 625{,}000$ transitions). The sampler retrieves 4 segments of length 125 (i.e., $4 \times 125 = 500$ transitions).

Table 6: Hyperparameters for the Box Pushing Dense, Episode Length $T = 100$

	PPO	gSDE	GTrXL	SAC	CrossQ	TCP	BBRL	TOP-ERL	T-SAC
number samples	48000	80000	8000	8	1	152	152	4	4 * 100
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.	n.a.	n.a.
discount factor	1.0	1.0	0.99	0.99	0.99	1.0	1.0	1.0	0.99
ϵ_μ	n.a.	n.a.	n.a.	n.a.	n.a.	0.05	0.1	0.005	n.a.
ϵ_Σ	n.a.	n.a.	n.a.	n.a.	n.a.	0.0005	0.00025	0.0005	n.a.
trust region loss coef.	n.a.	n.a.	n.a.	n.a.	n.a.	1	10	1.0	n.a.
optimizer	adam	adam	adam	adam	adam	adam	adam	adam	adamw
epochs	10	10	5	1	1	50	20	15	20
learning rate	5e-5	1e-4	2e-4	3e-4	3e-4	3e-4	3e-4	3e-4	2.5e-4
use critic	True	True	True	True	True	True	True	True	True
epochs critic	10	10	5	1	1	50	10	30	100
learning rate critic	1e-4	1e-4	2e-4	3e-4	3e-4	1e-3	3e-4	5e-5	2.5e-5
number minibatches	40	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	2000	1000	512	256	n.a.	n.a.	512	256
buffer size	n.a.	n.a.	n.a.	2e6	1e6	n.a.	n.a.	7000	20000 * 100
learning starts	0	0	0	1e5	5000	0	0	8000	5000
temperature warmup	0	0	0	0	0	0	0	0	0
polyak_weight	n.a.	n.a.	n.a.	5e-3	1.0	n.a.	n.a.	5e-3	5e-3
SDE sampling frequency	n.a.	4	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0.01	0	auto	auto	0	0	0	0
normalized observations	True	True	False	False	False	True	False	False	False
normalized rewards	True	True	0.1	False	False	False	False	False	False
observation clip	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	10.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	0.2	10.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	0.2	0.1	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[512, 512]	[256, 256]	n.a.	[256, 256]	[256, 256]	[128, 128]	[128, 128]	[256, 256]	[4 layers × 512]
hidden layers critic	[512, 512]	[256, 256]	n.a.	[256, 256]	[256, 256]	[256, 256]	[256, 256]	n.a.	n.a.
hidden activation	tanh	tanh	relu	tanh	tanh	leaky_relu	leaky_relu	leaky_relu	relu
orthogonal initialization	Yes	No	xavier	fanin	fanin	Yes	Yes	Yes	fanin
initial std	1.0	0.05	1.0	1.0	1.0	1.0	1.0	1.0	1.0
number of heads	-	-	4	-	-	-	-	8	4
dims per head	-	-	16	-	-	-	-	16	64
number of attention layers	-	-	4	-	-	-	-	2	2
MP type	n.a.	n.a.	value	n.a.	n.a.	ProDMP	ProDMP	ProDMP	n.a.
number basis functions	n.a.	n.a.	value	n.a.	n.a.	8	8	8	n.a.
weight scale	n.a.	n.a.	value	n.a.	n.a.	0.3	0.3	0.3	n.a.
goal scale	n.a.	n.a.	value	n.a.	n.a.	0.3	0.3	0.3	n.a.

Table 7: Hyperparameters for the Box Pushing Sparse, Episode Length $T = 100$

	PPO	gSDE	GTrXL	SAC	CrossQ	TCP	BBRL	TOP-ERL	T-SAC
number samples	48000	80000	8000	8	1	76	76	4	4 * 100
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.	n.a.	n.a.
discount factor	1.0	1.0	1.0	0.99	0.99	1.0	1.0	1.0	1.0
ϵ_μ	n.a.	n.a.	n.a.	n.a.	n.a.	0.05	0.1	0.005	n.a.
ϵ_Σ	n.a.	n.a.	n.a.	n.a.	n.a.	0.0005	0.00025	0.0005	n.a.
trust region loss coef.	n.a.	n.a.	n.a.	n.a.	n.a.	1	10	1.0	n.a.
optimizer	adam	adam	adam	adam	adam	adam	adam	adam	adamw
epochs	10	10	5	1	1	50	20	15	20
learning rate	5e-4	1e-4	2e-4	3e-4	3e-4	3e-4	3e-4	3e-4	2.5e-4
use critic	True	True	True	True	True	True	True	True	True
epochs critic	10	10	5	1	1	50	10	30	100
learning rate critic	1e-4	1e-4	2e-4	3e-4	3e-4	3e-4	3e-4	5e-5	3.0e-4
number minibatches	40	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
batch size	n.a.	2000	1000	512	512	n.a.	n.a.	512	256
buffer size	n.a.	n.a.	n.a.	2e6	2e6	n.a.	n.a.	7000	20000 * 100
learning starts	0	0	0	1e5	1e5	0	0	400	2000
temperature warmup	0	0	0	0	0	0	0	0	0
polyak_weight	n.a.	n.a.	0	5e-3	1.0	n.a.	n.a.	5e-3	5e-3
SDE sampling frequency	n.a.	4	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
entropy coefficient	0	0.01	0	auto	auto	0	0	0	0
normalized observations	True	True	False	False	False	True	False	False	False
normalized rewards	True	True	0.1	False	False	False	False	False	False
observation clip	10.0	n.a.	False	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
reward clip	10.0	10.0	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
critic clip	0.2	0.2	10.0	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
importance ratio clip	0.2	0.2	0.1	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
hidden layers	[512, 512]	[256, 256]	n.a.	[256, 256]	[256, 256]	[128, 128]	[128, 128]	[256, 256]	[4 layers × 512]
hidden layers critic	[512, 512]	[256, 256]	n.a.	[256, 256]	[2048, 2048]	[256, 256]	[256, 256]	n.a.	n.a.
hidden activation	tanh	tanh	relu	tanh	relu	leaky_relu	leaky_relu	leaky_relu	leaky_relu
orthogonal initialization	Yes	No	xavier	fanin	fanin	Yes	Yes	Yes	fanin
initial std	1.0	0.05	1.0	1.0	1.0	1.0	1.0	1.0	1.0
number of heads	-	-	4	-	-	-	-	8	4
dims per head	-	-	16	-	-	-	-	16	64
number of attention layers	-	-	4	-	-	-	-	2	2
MP type	n.a.	n.a.	value	n.a.	n.a.	ProDMP	ProDMP	ProDMP	n.a.
number basis functions	n.a.	n.a.	value	n.a.	n.a.	8	8	8	n.a.
weight scale	n.a.	n.a.	value	n.a.	n.a.	0.3	0.3	0.3	n.a.
goal scale	n.a.	n.a.	value	n.a.	n.a.	0.3	0.3	0.3	n.a.

Table 8: Hyperparameters for the Gymnasium MuJoCo, Episode Length $T = 1000$

	TD3	CrossQ	SAC	T-SAC (Soft Copy)	T-SAC (Hard Copy)
number samples	1	1	1	4 * 20	4 * 20
GAE λ	n.a.	n.a.	n.a.	n.a.	n.a.
discount factor	0.99	0.99	0.99	0.99	0.99
optimizer	adam	adam	adam	adamw	adamw
epochs	1	1	1	12	12
learning rate	3e-4	1e-3	3e-4	3e-4	3e-4
use critic	True	True	True	True	True
epochs critic	1	3	1	60	60
learning rate critic	3e-4	1e-3	3e-4	3e-4	3e-4
batch size	256	256	256	256	256
buffer size	1e6	1e6	1e6	1e5 * 20	1e5 * 20
learning starts	5000	5000	5000	10000	10000
temperature warmup	0	0	0	10000	10000
polyak_weight	5e-3	1.0	5e-3	5e-3	1.0
entropy coefficient	auto	auto	auto	auto	auto
hidden layers	[256, 256]	[256, 256]	[256, 256]	[256, 256]	[256, 256]
hidden layers critic	[256, 256]	[2048, 2048]	[256, 256]	n.a.	n.a.
hidden activation	relu	relu	relu	relu	relu
orthogonal initialization	fanin	fanin	fanin	fanin	fanin
initial std	1.0	1.0	1.0	1.0	1.0
number of heads	-	-	-	4	4
dims per head	-	-	-	64	64
number of attention layers	-	-	-	2	2

Task-specific settings (Gymnasium MuJoCo). For **T–SAC**, the initial policy log-standard deviation is set to -5 for Ant, HumanoidStandup, and HalfCheetah, and to -10 for Hopper and Walker2d. For Hopper and Walker2d only, the target entropy is $H_{\text{target}} = -4 \cdot \dim(\mathcal{A})$; unless otherwise noted, other tasks use the SAC default $H_{\text{target}} = -\dim(\mathcal{A})$.

APPENDIX: USE OF LARGE LANGUAGE MODELS (LLMS)

We used large language models (LLMs) as a general-purpose assistant during writing and development. Its roles included:

- grammar and spell-checking, language polishing, and minor stylistic edits;
- drafting and rewriting multi-paragraph text (e.g., introductions, preliminaries, and parts of experimental write-ups) based on author-provided outlines and results;
- high-level suggestions for debugging strategies and hyperparameter choices;
- assistance with literature search (proposing search queries and surfacing candidate papers).

All BibTEX entries were copied from Google Scholar; the LLMs did not generate or edit bibliographic entries. The LLMs did *not* originate the paper's main idea, problem formulation, algorithmic design, or experimental plan, and it was not used to generate or alter data, results, or figures. All citations were selected and verified by the authors against the original sources. All LLMs outputs were reviewed and, when necessary, edited or discarded. No confidential or proprietary data were shared with the LLMs.