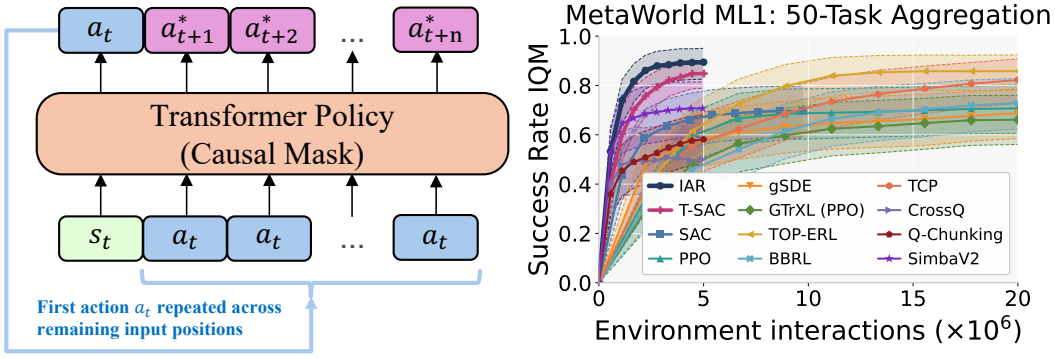
# IAR: IMPLICIT ACTION REPETITION FOR POLICY OPTIMIZATION WITH SEQUENCE CRITICS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In actor–critic learning, policy updates depend on both the learned value function and the action contexts in which it is queried. We study action repetition as a mechanism for constructing these contexts during policy optimization. We introduce implicit-action repetition (IAR), an actor-update mechanism for continuous control with sequence-conditioned critics. For each replay state, IAR samples an initial action and constructs prefixes that repeat it for different durations. The policy then samples a continuation action conditioned on each prefix, and the actor is optimized using the resulting sequence values. Repeated prefixes and critic parameters are held fixed during differentiation, allowing the actor to learn under hypothetical sustained-action contexts without requiring those repetitions to be executed in the environment. Once the initial action is sampled, all prefixes are available, allowing a causal Transformer to evaluate the continuation distributions in parallel. This removes sequential feedback among continuation predictions during actor updates, while environment interaction and critic bootstrapping retain autoregressive sampling. Across all 50 manipulation tasks under the Meta-World ML1 protocol, the resulting method achieves an aggregate success rate of approximately $90\%$, compared with approximately $84\%$ for T-SAC. These results support repeated-action conditioning as an effective mechanism for policy optimization with sequence critics.




(a) IAR policy overview

(b) Aggregate Meta-World ML1 performance

Figure 1: Overview of IAR and Meta-World ML1 performance. (a) During actor updates, IAR repeats the initial action as conditioning input, enabling parallel prediction of continuation actions for different prefix lengths. Environment interaction and critic bootstrapping remain autoregressive. (b) IAR is trained for 5 million environment interactions; available baseline curves extend to 20 million. Matched-budget comparisons are made at 5 million interactions. Curves report the unweighted mean across 50 tasks of within-task IQMs of seed-level success rates. Shading indicates 95% bootstrap confidence intervals.

## 1 INTRODUCTION

Action repetition offers a simple way to improve exploration in reinforcement learning (Lakshminarayanan et al., 2017; Sharma et al., 2017; Biedenkapp et al., 2021). By sustaining an action over



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

multiple timesteps, an agent can reduce dithering and reach regions of the state space that are difficult to visit through temporally uncorrelated exploration. Prior work on temporally extended exploration, action persistence, and novelty-guided repetition demonstrates that this temporal structure can increase state-space coverage and improve learning efficiency (Dabney et al., 2020; Sabbioni et al., 2023; Hu et al., 2025). These findings motivate a complementary question for actor–critic learning: can repetition also help the actor make better use of the information already represented by its critic?

In off-policy actor–critic methods for continuous control, the return-maximization component of the actor objective takes the form

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\theta(\cdot|s)}\left[-\widehat{Q}_\phi(s,a)\right], \tag{1}$$

where $\mathcal{D}$ is the replay buffer and $\widehat{Q}_\phi$ denotes the critic estimate used for actor optimization. This is the return-maximization component of the soft actor–critic objective (Haarnoja et al., 2018a). With the critic parameters held fixed, reparameterized actor updates propagate gradients through the sampled actions. The policy's learning signal therefore depends not only on the learned critic but also on the action inputs at which its gradients are evaluated.

Exploration and actor optimization address distinct but coupled aspects of learning. Exploration shapes the data used to train the critic and the replay-state distribution used for actor updates. Our focus is the actor update itself: how to construct the action contexts used to query a learned critic, with the sampled replay states and critic parameters held fixed during the update.

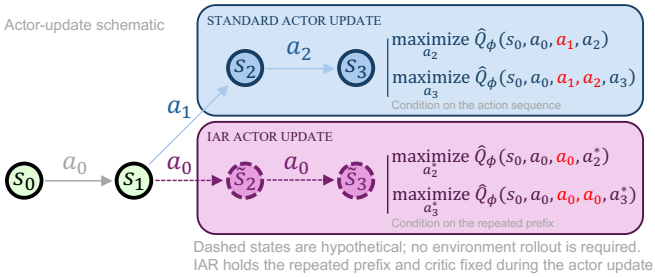



Figure 2: Schematic comparison of standard and IAR actor updates. Blue arrows depict transitions under successive actions; the purple branch illustrates a hypothetical repeated-action trajectory. IAR evaluates continuation actions with a sequence-conditioned critic while holding the repeated prefixes fixed. Hypothetical states are shown only for illustration and need not be explicitly generated or observed.

A sequence-conditioned critic evaluates an initial state together with an action sequence, making it possible to evaluate policy-generated continuation actions under different action prefixes. T-SAC (Tian et al., 2026) learns such sequence values while retaining a single-step actor objective. This motivates an actor-update mechanism that uses sequence-conditioned values across different prefix lengths while allowing the corresponding continuation actions to be sampled efficiently.

We introduce **implicit-action repetition (IAR)**, which constructs repeated-action prefixes exclusively during actor optimization, without requiring those repetitions to be executed in the environment. Building on the sequence-value formulation of T-SAC, IAR first samples an initial action at each replay state and constructs a family of prefixes that repeat this action for different durations. The policy then samples a continuation action conditioned on each prefix. The sequence critic estimates the return from executing the specified prefix and continuation action and then following the interaction policy. The actor objective averages these sequence values together with the value of the initial action. During differentiation, the repeated-action prefixes are detached from the computation graph and the critic parameters are held fixed, so each term updates the actor through its newly sampled action. IAR thus uses hypothetical sustained-action contexts at multiple temporal horizons to provide learning signals for policy improvement.

This choice of conditioning contexts also removes the sequential dependency between continuation predictions during actor optimization. Our causal Transformer policy conditions each action on the initial state and the preceding action sequence. Direct autoregressive sampling requires each generated action to become an input to the next prediction. Under IAR, all repeated-action prefixes are known once the initial action has been sampled, allowing the policy to evaluate the continuation distributions for all prefix lengths in a single parallel forward pass. Environment interaction and critic bootstrap generation retain autoregressive sampling, so bootstrap generation continues to use the deployed action-generation procedure.

We evaluate the resulting learning procedure on all 50 manipulation tasks under the Meta-World ML1 protocol (Yu et al., 2020). Our method achieves an aggregate success rate of approximately $90\%$, compared with approximately $84\%$ for T-SAC, the previous state-of-the-art baseline in our comparison. These results support repeated-action conditioning as an effective approach to policy optimization with sequence critics.

Our contributions are threefold:

- We introduce IAR, an actor-update mechanism that uses sequence-critic estimates to optimize policy actions conditioned on repeated-action prefixes of different lengths.
- We develop a causal Transformer policy that evaluates these contexts in parallel during actor optimization while retaining autoregressive generation for interaction and critic bootstrapping.
- We demonstrate improved performance over T-SAC, the previous state-of-the-art (SOTA), across the 50 Meta-World ML1 tasks.

## 2 PRELIMINARIES

**Off-Policy Actor–Critic Learning.** For an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, let $\mathcal{D}$ denote a replay buffer and $\pi_\theta$ a stochastic actor. SAC (Haarnoja et al., 2018a;b) learns twin critics and uses $\widehat{Q}_\phi = \min_{i=1,2} Q_{\phi_i}$. Its actor loss and sampled TD target are

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\theta(\cdot|s)}[\alpha \log \pi_\theta(a \mid s) - \widehat{Q}_\phi(s, a)], \tag{2}$$

$$y = r + \gamma(1-d)[\widehat{Q}_{\bar{\phi}}(s', a') - \alpha \log \pi_\theta(a' \mid s')], \tag{3}$$

where $(s, a, r, s', d)$ is a replay transition, $d$ indicates termination, $a' \sim \pi_\theta(\cdot \mid s')$, $\bar{\phi}$ denotes target critic parameters, and $\alpha$ controls entropy regularization. The scalar critics in SAC minimize squared TD error. During actor optimization, critic parameters are held fixed and gradients propagate through reparameterized policy actions. Our method sets $\alpha = 0$ in both the actor objective and critic targets.

**Sequence-Conditioned Value Functions.** Let $\mathbf{a}_{t:t+\ell-1} = (a_t, \ldots, a_{t+\ell-1})$ denote an action sequence of length $\ell$. For a continuation policy $\nu$, define the sequence value as

$$Q^\nu(s_t, \mathbf{a}_{t:t+\ell-1}) = \mathbb{E}\left[\sum_{j=0}^{\ell-1} \gamma^j r_{t+j} + \gamma^\ell V^\nu(s_{t+\ell})\right], \tag{4}$$

where the expectation is over transitions from $s_t$ under the specified action sequence, and $V^\nu$ is the expected discounted return from subsequently following $\nu$. Terminal states are treated as absorbing, with zero subsequent rewards and value. For $\ell = 1$, this reduces to the ordinary action value. The continuation policy is determined by the interaction rule and may differ from the raw action proposal. A sequence critic $Q_\phi$ approximates this value using the initial state and action sequence as inputs; intermediate states are not required as conditioning inputs.

T-SAC (Tian et al., 2026) learns such values for multiple prefix lengths using a causal Transformer critic. For valid replay prefixes, its TD targets take the form

$$y_t^{(\ell)} = \sum_{j=0}^{\ell-1} \gamma^j r_{t+j} + \gamma^\ell (1 - d_{t+\ell-1}) \bar{V}(s_{t+\ell}), \tag{5}$$

where $\bar{V}(s) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\widehat{Q}_{\bar{\phi}}(s, a)]$ uses the single-action critic output, with $\bar{\phi}$ denoting the critic parameters used for bootstrap evaluation. Valid prefixes do not cross episode boundaries. T-SAC averages per-horizon TD losses without importance sampling and retains an entropy-regularized, single-step actor.

**Autoregressive Action-Sequence Policies.** An autoregressive policy factorizes the joint distribution of an action sequence as

$$\pi_\theta(\mathbf{a}_{t:t+\ell-1} \mid s_t) = \prod_{j=0}^{\ell-1} \pi_{\theta,j}(a_{t+j} \mid s_t, \mathbf{a}_{t:t+j-1}). \tag{6}$$

For $j = 0$, the prefix is empty and $\pi_{\theta,0}(\cdot \mid s_t) = \pi_\theta(\cdot \mid s_t)$. The position-dependent conditional distributions share parameters $\theta$ and can be represented by a causal Transformer (Vaswani et al., 2017). When all conditioning prefixes are given, the distributions at different positions can be evaluated in parallel. Sampling a complete sequence remains sequential, however, because each generated action becomes an input to subsequent predictions. This distinction motivates constructing known conditioning prefixes for parallel actor updates.

## 3 Related Work

**Sequence critics and policy improvement.** TOP-ERL and T-SAC use Transformer critics to learn action-sequence values (Li et al., 2025; Tian et al., 2026). CQN-AS extends value-based control to action sequences (Seo & Abbeel, 2026), while SEAR combines multi-horizon prefix-value learning with replanning (Nagy et al., 2026). Multi-step actor optimization is also explored by DoMo-AC (Tang et al., 2023). Building on sequence-conditioned values, IAR optimizes fresh continuation actions under fixed repeated-action prefixes, providing multiple temporal contexts for actor updates.

**Action chunking and candidate selection.** Learning Routines and Q-chunking perform off-policy learning over temporally extended action representations (Cetin & Celiktutan, 2021; Li et al., 2026b). Decoupled Q-chunking separates policy and critic chunk lengths (Li et al., 2026a); DEAS decouples offline value learning from the actor (Kim et al., 2026); and AC3 learns continuous chunks using success-conditioned actor updates (Yang et al., 2026). ACSAC jointly selects action candidates and execution lengths using a causal Transformer critic (Chen et al., 2026). IAR adopts critic-guided candidate selection while focusing on the conditioning contexts used during actor optimization. Its detached prefixes fix action inputs within an actor update, a different mechanism from DEAS's detached value learning.

**Action repetition and temporal persistence.** Options formalize temporal abstraction (Sutton et al., 1999), while repetition methods introduce persistence through sampled or learned durations (Dabney et al., 2020; Sharma et al., 2017; Biedenkapp et al., 2021). Prior work also studies control-frequency adaptation (Metelli et al., 2020), policy-gradient robustness to time discretization (Park et al., 2021), and persistence-aware value updates (Yu et al., 2021; Sabbioni et al., 2023). IAR constructs hypothetical repeated prefixes during actor optimization without requiring their execution. Each prefix conditions a fresh continuation action, through which the corresponding sequence value supplies the actor gradient. These known prefixes also enable parallel continuation predictions, while interaction and bootstrapping remain autoregressive.

## 4 Method

Our method combines implicit-action repetition (IAR), best-of-$N$ selection, and categorical distributional critics. IAR uses repeated-action contexts for actor updates, while interaction and bootstrapping remain autoregressive (Fig. 3). Let $H_s$, $H_t$, and $H_\pi$ denote the respective interaction, bootstrap, and actor-update horizons, and $\mathbf{a} = f_\theta^{(H)}(s, \boldsymbol{\epsilon})$ an autoregressive chunk of length $H$ sampled with noise $\boldsymbol{\epsilon}$. Each $Q_{\phi_i}(s, \mathbf{a})$ estimates the return from executing $\mathbf{a}$ and then following the interaction policy; we write $\widehat{Q}_\phi = \min_{i=1,2} Q_{\phi_i}$.

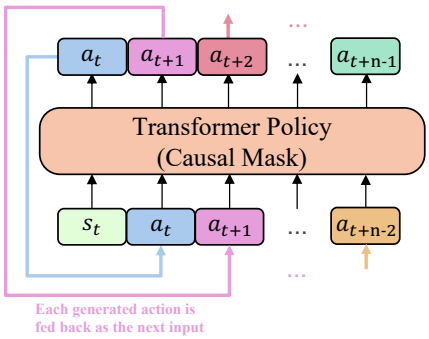



Figure 3: Autoregressive generation for interaction and bootstrapping: each sampled action conditions subsequent predictions.

### 4.1 Policy Updates with Implicit-Action Repetition

For $s_t \sim \mathcal{D}$, IAR samples an initial action and detaches it to form the repeated context:

$$a_t = g_{\theta,0}(s_t, \varnothing, \epsilon_0), \qquad c_t = \mathrm{sg}(a_t), \tag{7}$$

where $g_{\theta,n}$ is the reparameterized sampler at position $n$, all positions share $\theta$, and sg denotes stop-gradient. Define $\mathbf{c}_t^{(n)} = (c_t, \ldots, c_t)$ with $n$ copies and $\mathbf{c}_t^{(0)} = \varnothing$. Continuation actions are

$$a^*_{t+n} = g_{\theta,n}(s_t, \mathbf{c}_t^{(n)}, \epsilon_n), \qquad n = 1, \ldots, H_\pi - 1, \tag{8}$$

with $a_t^* = a_t$; the star denotes a fresh sample. Once $a_t$ is sampled, all prefixes are known, enabling one parallel Transformer forward pass for the remaining conditional distributions.

The actor minimizes

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_{s_t\sim\mathcal{D},\,\boldsymbol{\epsilon}}\left[\frac{1}{H_\pi}\sum_{n=0}^{H_\pi-1}\widehat{Q}_\phi\left(s_t, \mathbf{c}_t^{(n)}, a_{t+n}^*\right)\right], \tag{9}$$

where $Q(s, \mathbf{c}, a)$ scores the concatenated sequence $(\mathbf{c}, a)$. Term $n$ evaluates repeating $a_t$ for $n$ steps, taking $a_{t+n}^*$, and then following the interaction policy. With prefixes and critic parameters fixed, gradients flow only through each term's fresh action; the $n = 0$ term directly trains $a_t$. These are separate sequence queries, not a chunk formed by concatenating continuation samples. Their shared prefixes enable the packed evaluation in Sec. 4.4.

### 4.2 Best-of-$N$ Action Selection

Following Q-chunking (Li et al., 2026b), we draw $N$ independent candidates $\mathbf{a}^{(j)} = f_\theta^{(H)}(s, \boldsymbol{\epsilon}^{(j)})$ and select using the online critics:

$$j^\star = \arg\max_{1\le j\le N}\widehat{Q}_\phi(s, \mathbf{a}^{(j)}), \qquad \mathbf{a}^{\text{sel}} = \mathbf{a}^{(j^\star)}. \tag{10}$$

For interaction, $H = H_s$ and the selected chunk is executed until completion or episode termination. For bootstrapping, $H = H_t$ and the target critics evaluate the selected chunk. The actor loss does not differentiate through selection, and $N$ is independent of the three horizon hyperparameters.

### 4.3 Distributional Critic

Following categorical value learning (Bellemare et al., 2017; Lee et al., 2025), each critic represents $Z_{\phi_i}(s, \mathbf{a})$ on fixed, uniformly spaced atoms $\mathcal{Z} = \{z_m\}_{m=1}^M$. Its probabilities define

$$Q_{\phi_i}(s, \mathbf{a}) = \sum_{m=1}^{M} z_m p_{\phi_i,m}(s, \mathbf{a}), \tag{11}$$

whose twin-critic minimum is used for actor updates and candidate ranking.

For a replay prefix $\mathbf{a} = a_{t:t+\ell-1}$ that does not cross an episode boundary, let $R = \sum_{j=0}^{\ell-1}\gamma^j r_{t+j}$, $s^+ = s_{t+\ell}$, and $\Gamma = \gamma^\ell(1 - d_{t+\ell-1})$. At $s^+$, Eq. (10) selects an autoregressive bootstrap chunk $\mathbf{a}^+$ of length $H_t$. We average the target critics' probabilities to obtain $\bar{Z}$ and project the resulting return:

$$\bar{p}_m(s^+, \mathbf{a}^+) = \frac{1}{2}\sum_{i=1}^{2} p_{\bar{\phi}_i,m}(s^+, \mathbf{a}^+),$$
$$q = \Pi_{\mathcal{Z}}\left[R + \Gamma\bar{Z}(s^+, \mathbf{a}^+)\right], \tag{12}$$

where $\bar{\phi}_i$ are target parameters and $\Pi_{\mathcal{Z}}$ projects onto the fixed support, clipping values outside its bounds. This target uses an equal mixture of the two distributions, rather than the minimum of their expected returns; terminal prefixes have $\Gamma = 0$. Each critic minimizes cross-entropy over valid replay prefixes and bootstrap samples:

$$\mathcal{L}_Z(\phi_i) = -\mathbb{E}\left[\sum_{m=1}^{M}\text{sg}(q_m)\log p_{\phi_i,m}(s_t, \mathbf{a})\right]. \tag{13}$$

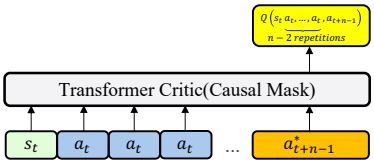



Figure 4: **Repeated-prefix critic query.** The critic scores a repeated-action prefix and one fresh continuation. With the prefix and critic fixed, gradients flow through the fresh action.

### 4.4 Critic Evaluation with Repeated-Action Prefixes

We compute the $H_\pi$ values in Eq. (9) together using shared prefix representations. Set $H = H_\pi$, $x_0 = s_t$, $x_i = c_t$ for $i = 1, \ldots, H-1$, and $y_n = a_{t+n}^*$ for $n = 0, \ldots, H-1$. The packed input is

$$(x_0, \ldots, x_{H-1}, y_0, \ldots, y_{H-1}),$$

with logical positions $i$ for $x_i$ and $n+1$ for $y_n$. At every Transformer layer, $x_i$ attends only to $(x_0, \ldots, x_i)$, and $y_n$ only to $(x_0, \ldots, x_n, y_n)$. Thus, context tokens cannot depend on sampled actions, and each $y_n$ independently evaluates $Q_{\phi_i}(s_t, \mathbf{c}_t^{(n)}, a_{t+n}^*)$ with the same attention paths and positions as its separate causal query (Figs. 4 and 5). Only the $y_n$ outputs contribute to the actor loss, using the twin-min expected values and gradient rule above.

### 4.5 Batched Bootstrap Computation

Each round performs $N_c$ critic updates followed by $N_p$ actor updates (100 and 40 in Meta-World). At the critic-phase boundary, we pool valid bootstrap states from $B$ sampled segments, each with $T$ bootstrap positions, yielding at most $BT$ states. Candidate generation is batched across states and candidates, but remains autoregressive within each length-$H_t$ chunk.

| | $s_t$ | $a_t$ | $a_t$ | $a_t$ | $a_{t+1}$ | $a_{t+2}$ |
|---|---|---|---|---|---|---|
| $Q_\phi(s_t, a_t)$ | 1 | 0 | 0 | 1 | 0 | 0 |
| $Q_\phi(s_t, a_t, a_{t+1})$ | 1 | 1 | 0 | 0 | 1 | 0 |
| $Q_\phi(s_t, a_t, a_t, a_{t+2})$ | 1 | 1 | 1 | 0 | 0 | 1 |

Figure 5: Sampled-action rows of the packed attention mask for $H = 3$. 1 allows attention, while 0 means mask-out; the two $a_t$ columns before the divider are detached.

Generation, online-critic ranking, and target evaluation use the parameters at the phase boundary. For each bootstrap state, we cache the selected chunk's averaged target distribution $\bar{Z}$, before adding rewards or discounting (Fig. 6). Each replay window combines its endpoint's cached distribution with its own $R$ and $\Gamma$ via Eq. (12). The cache stays fixed for all $N_c$ updates, including intervening target-network updates, and is refreshed at the next critic phase. During actor updates, fresh initial and continuation actions are sampled at every optimizer step.

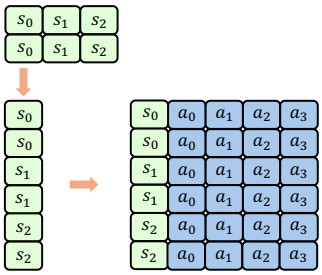



Figure 6: Batched bootstrap computation. Pooling states (e.g., $256 \times 100 \to 25{,}600 \times 1$) parallelizes autoregressive rollouts across states; cached target distributions serve all critic updates in the round.

## 5 Experimental Results

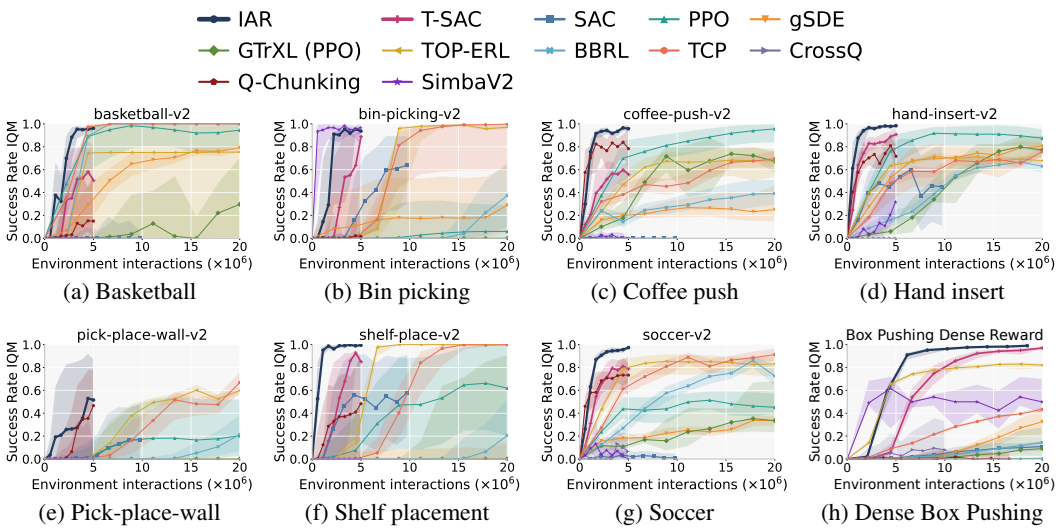
### 5.1 Main Results

**Meta-World.** Across all 50 Meta-World ML1 tasks, IAR achieves an aggregate terminal success rate of approximately 90% after $5 \times 10^6$ environment interactions, compared with approximately 84% for T-SAC at the same budget (Figure 1b). The aggregate is the unweighted mean of the 50 task-wise IQMs, as defined above. IAR learns faster than T-SAC and attains the highest plotted aggregate central estimate at this budget. Longer-budget baseline curves are included for context; their final values should not be interpreted as matched-budget comparisons.

Figure 7(a)–(g) highlights strong performance across several manipulation settings. On *bin-picking*, *hand-insert*, and *shelf-place*, IAR approaches unit success-rate IQM within roughly one to two million interactions. It also learns rapidly on *basketball*, *coffee-push*, and *soccer*, reaching close to 100% by $5 \times 10^6$ interactions. At this common budget, T-SAC reaches approximately 50%, 60%, and 80%, respectively. The obstacle-constrained *pick-place-wall* task provides another clear example: IAR reaches approximately 85–90%, whereas the other plotted methods remain at or below approximately 50% at the same budget.

These selected tasks illustrate IAR's sample efficiency rather than uniform dominance across the benchmark. For example, SimbaV2 and CrossQ also learn rapidly on *bin-picking*, and the full results contain tasks on which IAR remains below unit success or other methods achieve higher central estimates, such as *disassemble*. The complete task-wise curves are reported in Figures 9 and 10; the aggregate comparison uses all 50 tasks, not only the seven selected panels. The main results evaluate the complete IAR configuration, while Section 5.2 examines its individual design choices.

**Box Pushing.** Figure 7(h) reports success-rate IQM on dense Box Pushing, where a 7-DoF Franka Emika Panda arm uses a rod to move a box to a target pose. Episodes have a fixed horizon of 100 steps without early termination. Evaluation defines success by a position error of at most 0.05 m and an orientation error of at most 0.5 rad. The main IAR results use five training seeds, and T-SAC uses




(a) Basketball (b) Bin picking (c) Coffee push (d) Hand insert

(e) Pick-place-wall (f) Shelf placement (g) Soccer (h) Dense Box Pushing

Figure 7: Selected task-level results. Panels (a)–(g) show seven Meta-World ML1 tasks spanning object placement, grasping, pushing, and insertion; panel (h) shows dense Box Pushing. The horizontal axes give environment interactions in millions. In panels (a)–(g), the vertical axis gives the within-task IQM of seed-level terminal success rates, and shading indicates 95% bootstrap confidence intervals. For these Meta-World tasks, IAR is trained for $5 \times 10^6$ interactions, while available baseline curves extend to $2 \times 10^7$. Panel (h) reports success-rate IQM with shaded uncertainty bands. The shared legend applies to all panels. Comparisons in the text use a common interaction budget unless stated otherwise.

eight (Appendix F). The plotted baselines are T-SAC, TOP-ERL, SAC, PPO, gSDE, GTrXL (PPO), BBRL, TCP, CrossQ, Q-Chunking, and SimbaV2.

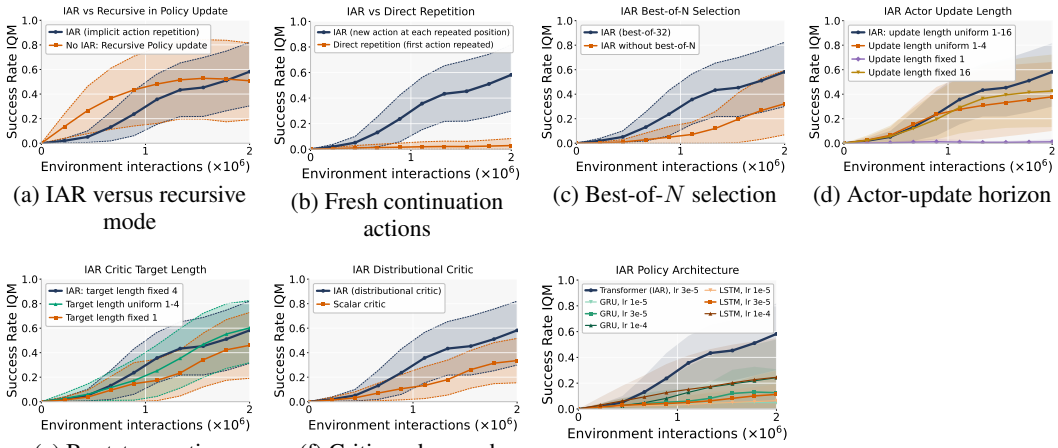
At a common budget of $10^7$ environment interactions, IAR achieves approximately 96% success-rate IQM, compared with approximately 80% and 79% for T-SAC and TOP-ERL, respectively. At $1.8 \times 10^7$ interactions, IAR reaches approximately 99%, while T-SAC and TOP-ERL reach approximately 95% and 83%, respectively. IAR reaches a 90% success-rate IQM at roughly $6.1 \times 10^6$ interactions, compared with $1.25 \times 10^7$ for T-SAC; the other plotted methods do not reach this threshold within their displayed budgets. IAR's final plotted value is approximately 99% at $1.85 \times 10^7$ interactions. Most baseline curves extend to $2 \times 10^7$ interactions, whereas CrossQ ends at approximately $10^7$ due to compute limit. These results show faster learning and a higher late-training central estimate for IAR on this task, although T-SAC narrows the gap as training proceeds.

Table 1: Approximate success rates (%) at 5M environment interactions.

| Algorithm | Meta-World | Box Pushing |
|---|---|---|
| **IAR (ours)** | **89.4** | **73.3** |
| T-SAC | 84.8 | 18.0 |
| SimbaV2 | 70.8 | 58.7 |
| SAC | 67.2 | 1.0 |
| TOP-ERL | 64.3 | 67.8 |
| PPO | 61.3 | 0.0 |
| gSDE | 58.5 | 0.1 |
| TCP | 57.1 | 10.2 |
| GTrXL (PPO) | 50.1 | 0.3 |
| CrossQ | 50.0 | 4.0 |
| BBRL | 47.9 | 1.0 |
| Q-Chunking | 58.0 | 0.3 |

Meta-World: unweighted mean of 50 task-wise IQMs. Box Pushing: within-task success-rate IQM.

**Computational Efficiency** Repeated-action conditioning makes all continuation contexts available before the actor update, allowing the Transformer to evaluate their action distributions in parallel. Environment interaction and critic bootstrapping retain autoregressive sampling. The reported IAR Meta-World configuration uses a UTD ratio of 0.5; however, training cost also includes action generation, candidate selection, and critic updates. Per $10^6$ environment interactions, the reported training times are 2.10 h for CrossQ, 3.25 h for SimbaV2, approximately 5.8 h for T-SAC, and 5.26 h for IAR. The SimbaV2 run uses UTD 2, matching its cluster configuration. Thus, IAR takes about 9.3% less time than T-SAC, but about 2.50× and 1.62× as long as CrossQ and SimbaV2, respectively. IAR has 81,668 actor parameters and 1,086,720 critic parameters (1,168,388 in total). The low update ratio and sample-efficiency gains




(a) IAR versus recursive mode (b) Fresh continuation actions (c) Best-of-$N$ selection (d) Actor-update horizon

(e) Bootstrap-action horizon (f) Critic and reward scaling (g) Policy architecture

Figure 8: Seven ablations on six Meta-World tasks. The horizontal axis gives environment interactions and the vertical axis gives aggregate success-rate IQM. The shaded bands show the uncertainty displayed in the plots. Panel (f) jointly changes the critic representation and reward scaling; panel (g) includes three learning rates for each recurrent architecture.

therefore do not imply the lowest wall-clock cost. Appendix E reports the hardware configuration, parameter counts, and run identifiers. These run-level comparisons do not isolate the cost of the actor update mechanism or establish time to a common success threshold.

### 5.2 Ablation Study

We evaluate seven design choices on six Meta-World tasks: `assembly-v2`, `basketball-v2`, `coffee-push-v2`, `disassemble-v2`, `pick-place-v2`, and `pick-place-wall-v2`. Figure 8 shows aggregate success-rate IQM curves over $2 \times 10^6$ environment interactions; Appendix B provides the corresponding task-wise results. Unless varied in the relevant comparison, the reference configuration uses best-of-32 selection, an actor-update horizon sampled uniformly from $\{1, \ldots, 16\}$, and its critic is distributional with scaled rewards. All numerical comparisons below use the common budget of $2 \times 10^6$ environment interactions.

**Repeated-action conditioning and continuation sampling.** Replacing IAR with the recurssive mode at $C = 1$ produces faster early learning, but its aggregate curve levels off near $51\%$, while IAR continues to improve to approximately $58\%$. The task-wise ordering is mixed, and the plotted uncertainty bands overlap substantially. In contrast, directly repeating the first action instead of sampling fresh continuation actions yields only about $3\%$ final aggregate IQM. This comparison supports the role of the newly sampled continuations in IAR's actor-update construction, beyond repetition alone.

**Best-of-$N$ selection.** Best-of-32 chunk selection reaches approximately $58\%$ aggregate IQM, versus $32\%$ for a single sampled chunk. The largest final task-wise gains occur on basketball, disassemble, and pick-place-wall. Selection also accelerates learning on `pick-place-v2`, although the single-sample variant catches up by the end of training.

**Actor-update horizon.** Sampling $H_\pi$ uniformly from $\{1, \ldots, 16\}$ yields the highest final aggregate IQM among the tested settings: approximately $58\%$, versus $43\%$ for $H_\pi = 16$, $38\%$ for uniform sampling from $\{1, \ldots, 4\}$, and $1\%$ for $H_\pi = 1$. These results favor training with a range of multi-step contexts, although the ordering varies across tasks. In particular, $H_\pi = 1$ removes all nonempty repeated-action prefixes from the actor objective.

**Bootstrap-action horizon.** A fixed horizon $H_t = 4$ improves aggregate learning over $H_t = 1$, with final IQMs of approximately $58\%$ and $46\%$, respectively. Sampling $H_t$ uniformly from $\{1, \ldots, 4\}$ reaches a similar, slightly higher final central estimate of approximately $60\%$. The task-wise results show substantial differences in which setting performs best. Here $H_t$ is the length of the action

chunk used for bootstrapping, distinct from both the best-of-$N$ candidate count and the replay-prefix length used to accumulate rewards.

**Critic representation and reward scaling.** The distributional critic with scaled rewards reaches approximately $58\%$ aggregate IQM, compared with $34\%$ for the scalar critic with unscaled rewards.

**Policy architecture.** The Transformer at learning rate $3 \times 10^{-5}$ achieves a higher final aggregate IQM than the tested GRU and LSTM policies, each evaluated at learning rates $10^{-5}$, $3 \times 10^{-5}$, and $10^{-4}$. Its final aggregate IQM is approximately $58\%$, whereas the best tested GRU and LSTM settings, both at $10^{-4}$, reach approximately $24\%$ and $25\%$, respectively. This result favors the tested Transformer configuration.

## 6 Conclusion and Future Work

We introduce implicit-action repetition (IAR), a policy-update mechanism that simplifies autoregressive action feedback in Transformer policies. By repeating the first sampled action as subsequent conditioning inputs during actor optimization, IAR preserves autoregressive sampling for environment interaction and critic bootstrapping without imposing repeated executed actions. Combined with best-of-$N$ action selection and categorical distributional critics, the method learns without explicit entropy regularization in either the actor objective or critic targets. On Meta-World ML1, it achieves a mean of task-wise success-rate IQMs of $90\%$ across all 50 tasks within $5 \times 10^6$ environment interactions at a UTD ratio of $0.5$, yielding the highest aggregate success among the evaluated baselines. These results support implicit repetition within policy optimization as a promising direction for learning effective continuous-control policies with a low update budget. The computational comparison also shows a runtime trade-off: IAR takes $5.26\,\mathrm{h}$ per $10^6$ interactions, below T-SAC's reported time of approximately $5.8\,\mathrm{h}$, but above the times for CrossQ and SimbaV2.

Our evaluation currently focuses on simulated manipulation, and the effect of implicit repetition on exploration remains an open question. Future work will examine how repeated conditioning changes policy gradients and interacts with candidate selection and distributional value learning. Profiling component-wise and end-to-end training costs across sequence lengths and update budgets will further clarify the trade-off between parallel actor updates and the autoregressive sampling retained elsewhere in training. Extending IAR to visual observations, partially observable environments, and real-robot control will also help establish the scope and limitations of repetition within policy optimization.

## AI USE STATEMENT

We used ChatGPT to assist with manuscript drafting and editing, organization, LaTeX formatting, and figure-layout preparation. We also used it to help clarify mathematical descriptions, refine the presentation and interpretation of experimental results, and identify inconsistencies between reported results and their textual descriptions. The authors are responsible for verifying AI-assisted content against the implementation, experimental evidence, and cited sources, and take full responsibility for the accuracy, originality, and integrity of the final manuscript.

## ETHICS STATEMENT

This work investigates reinforcement learning for robotic manipulation using simulated Meta-World ML1 and FANCY GYM benchmarks. The experiments evaluate learning performance and sample efficiency in simulation. Deployment on physical robots would require additional validation of robustness and safety, including appropriate operational constraints and human oversight. The reported simulation results should therefore be interpreted within the evaluated tasks and experimental conditions.

## REPRODUCIBILITY STATEMENT

Section 4 describes the IAR architecture, action-generation procedure, and actor–critic updates. Section 5 and Appendix F describe the experimental configuration, baseline comparisons, evaluation metrics, and aggregation procedures. Appendix B provides the ablation configurations and per-task results. Upon acceptance, we will publicly release the IAR source code, training curves and their underlying evaluation logs, and trained model checkpoints. The release will include training and evaluation configurations and instructions for running the experiments and reproducing the reported figures and aggregate results.

## REFERENCES


Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. Pmlr, 2017.

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. In *International conference on learning representations*, volume 2024, pp. 55293–55311, 2024.

André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. Temporl: Learning when to act. In *International Conference on Machine Learning*, pp. 914–924. PMLR, 2021.

Edoardo Cetin and Oya Celiktutan. Learning routines for effective off-policy reinforcement learning. In *International Conference on Machine Learning*, pp. 1384–1394. PMLR, 2021.

Qian Chen, Junqiao Zhao, Hongtu Zhou, Hang Yu, Yanping Zhao, Chen Ye, and Guang Chen. Acsac: Adaptive chunk size actor-critic with causal transformer q-network. *arXiv preprint arXiv:2605.11009*, 2026.

Will Dabney, Georg Ostrovski, and André Barreto. Temporally-extended {\epsilon}-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

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.

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.

Jianshu Hu, Paul Weng, and Yutong Ban. State-novelty guided action persistence in deep reinforcement learning. *Machine Learning*, 114(2):45, 2025.

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/.

Changyeon Kim, Haeone Lee, Younggyo Seo, Kimin Lee, and Yuke Zhu. Deas: Detached value learning with action sequence for scalable offline rl. In *International Conference on Learning Representations*, volume 2026, pp. 64035–64058, 2026.

Aravind Lakshminarayanan, Sahil Sharma, and Balaraman Ravindran. Dynamic action repetition for deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025.

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, 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. In *International Conference on Learning Representations*, volume 2024, pp. 18986–19018, 2024.

Ge Li, Dong Tian, Hongyi Zhou, Xinkai Jiang, Rudolf Lioutikov, and Gerhard Neumann. Top-erl: Transformer-based off-policy episodic reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 44534–44563, 2025.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled q-chunking. In *International Conference on Learning Representations*, volume 2026, pp. 145315–145389, 2026a.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. *Advances in Neural Information Processing Systems*, 38:55518–55553, 2026b.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International conference on machine learning*, pp. 3053–3062. PMLR, 2018.

Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. Control frequency adaptation via action persistence in batch reinforcement learning. In *International Conference on Machine Learning*, pp. 6862–6873. PMLR, 2020.

CF Nagy, Onur Celik, Emiliyan Gospodinov, Florian Seligmann, Weiran Liao, Aryan Kaushik, and Gerhard Neumann. Sear: Sample efficient action chunking reinforcement learning. *arXiv preprint arXiv:2603.01891*, 2026.

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

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.

Seohong Park, Jaekyeom Kim, and Gunhee Kim. Time discretization-invariant safe action repetition for policy gradient methods. *Advances in Neural Information Processing Systems*, 34:267–279, 2021.

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 `https://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.

Luca Sabbioni, Luca Al Daire, Lorenzo Bisi, Alberto Maria Metelli, and Marcello Restelli. Simultaneously updating all persistence values in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9668–9676, 2023.

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

Younggyo Seo and Pieter Abbeel. Coarse-to-fine q-network with action sequence for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 38:109915–109940, 2026.

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, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Yunhao Tang, Tadashi Kozuno, Mark Rowland, Anna Harutyunyan, Rémi Munos, Bernardo Avila Pires, and Michal Valko. Domo-ac: Doubly multi-step off-policy actor-critic algorithm. In *International Conference on Machine Learning*, pp. 33657–33673. PMLR, 2023.

Dong Tian, Onur Celik, and Gerhard Neumann. Chunking the critic: A transformer-based soft actor-critic with n-step returns. In *International Conference on Learning Representations*, volume 2026, pp. 120056–120094, 2026.

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.

Jiarui Yang, Bin Zhu, Jingjing Chen, and Yu-Gang Jiang. Actor-critic for continuous action chunks: A reinforcement learning framework for long-horizon robotic manipulation with sparse reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18692–18700, 2026.

Haonan Yu, Wei Xu, and Haichao Zhang. Taac: Temporally abstract actor-critic for continuous control. *Advances in neural information processing systems*, 34:29021–29033, 2021.

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.

# A FULL META-WORLD ML1 RESULTS

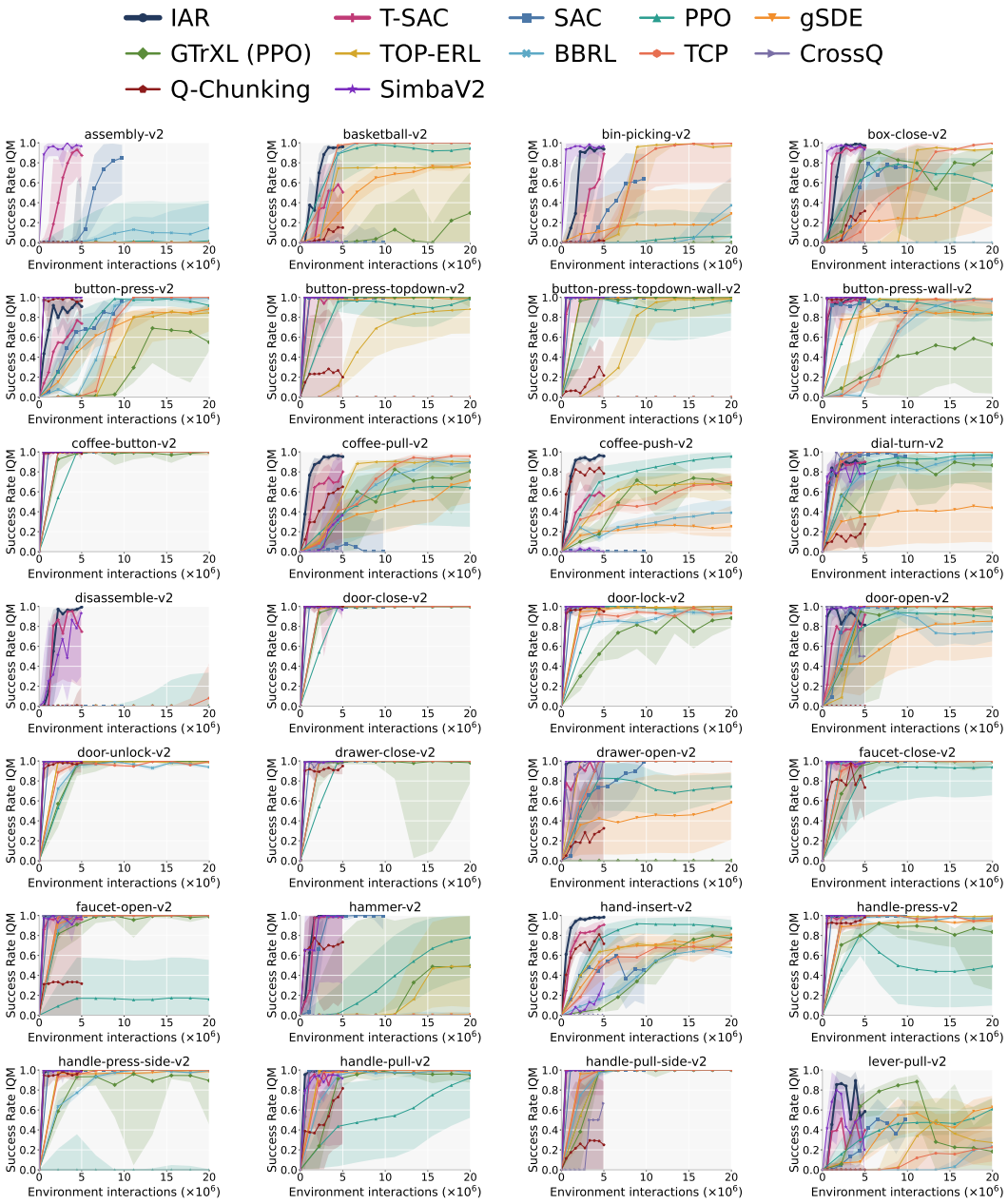



Figure 9: Task-wise terminal-success IQMs on Meta-World ML1 (tasks 1–28; Part 1 of 2).

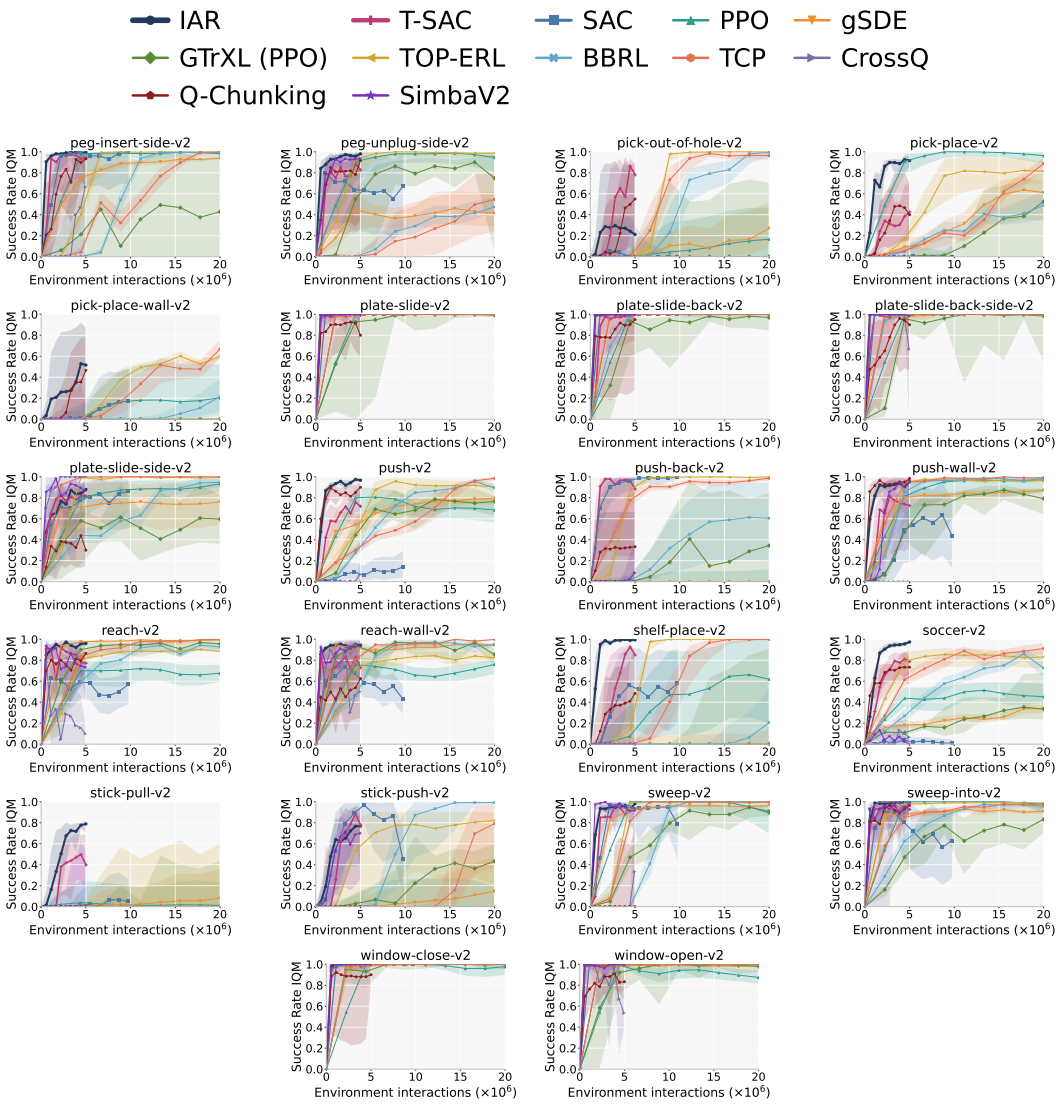



Figure 10: Task-wise terminal-success IQMs on Meta-World ML1 (tasks 29–50; Part 2 of 2).

# B Full Ablation Results

This appendix presents the six-task curves for the seven comparisons in Section 5.2, up to $2 \times 10^6$ environment interactions. Differences are descriptive and do not establish statistical significance.

## B.1 Implicit-Action Repetition versus recurssive Mode

Figure 11 compares IAR with the recurssive mode at $C = 1$. The recurssive mode learns faster initially and finishes higher on `basketball-v2` and `pick-place-v2`. IAR overtakes it near the end of training, reaching approximately $58\%$ aggregate IQM versus $51\%$. IAR finishes higher on `disassemble-v2` and `pick-place-wall-v2`; both variants finish similarly on `coffee-push-v2` and near zero on `assembly-v2`.

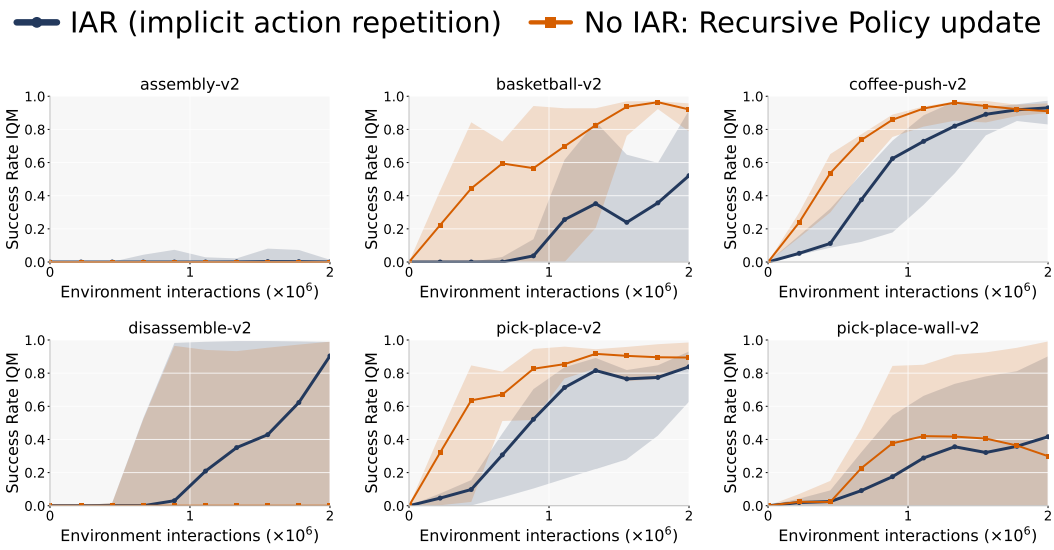



Figure 11: IAR versus the recurssive mode at $C = 1$. Task-wise success-rate IQM versus environment interactions; shading shows the uncertainty bands in the plots.

### B.2 Fresh Continuation Actions versus Direct Repetition

Figure 12 contrasts IAR's newly sampled continuation actions with direct repetition of the first action. Direct repetition remains close to zero on most tasks and reaches only approximately $3\%$ final aggregate IQM, compared with $58\%$ for IAR. Its clearest nonzero task-wise central curve is on `coffee-push-v2`, but it remains far below IAR there. IAR also learns on `basketball-v2`, `disassemble-v2`, `pick-place-v2`, and `pick-place-wall-v2`, whereas neither variant makes substantial progress on `assembly-v2`. These results support combining repeated contexts with freshly sampled continuation actions in the actor update.

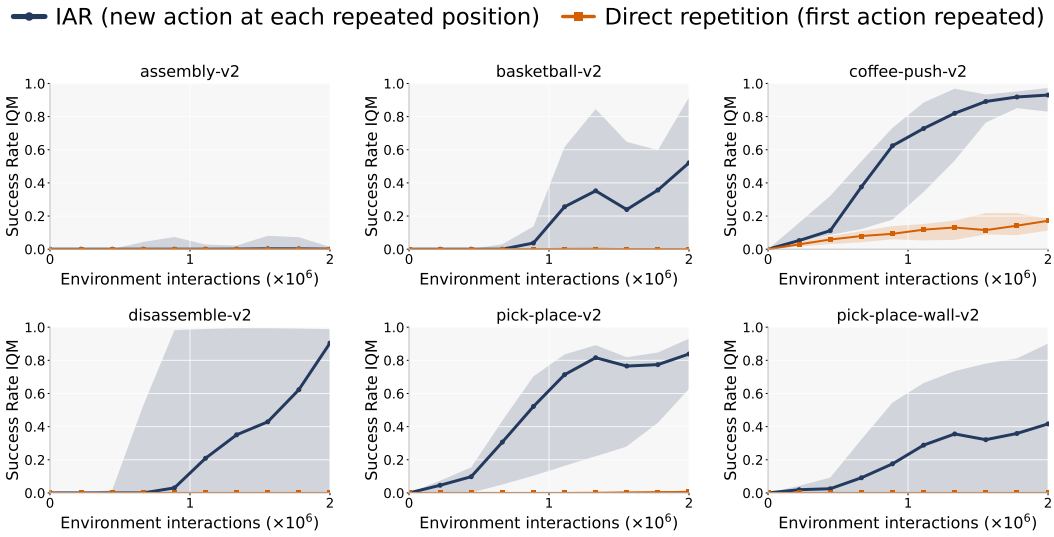



Figure 12: IAR with fresh continuation actions versus direct repetition of the first action. Task-wise success-rate IQM versus environment interactions; shading shows the uncertainty bands in the plots.

### B.3 BEST-OF-$N$ SELECTION

Figure 13 compares best-of-32 chunk selection with a single sampled chunk without critic-based ranking. The final aggregate IQM falls from approximately $58\%$ to $32\%$ when selection is removed. The largest final differences occur on `basketball-v2`, `disassemble-v2`, and `pick-place-wall-v2`. Selection also accelerates learning on `coffee-push-v2`, with a final IQM of about $93\%$ versus $70\%$. On `pick-place-v2`, selection improves early learning, but the single-sample variant eventually catches up, with both reaching about $84\%$. Both variants remain near zero on `assembly-v2`. Thus, selection provides a substantial aggregate benefit, with its effect concentrated in a subset of tasks.

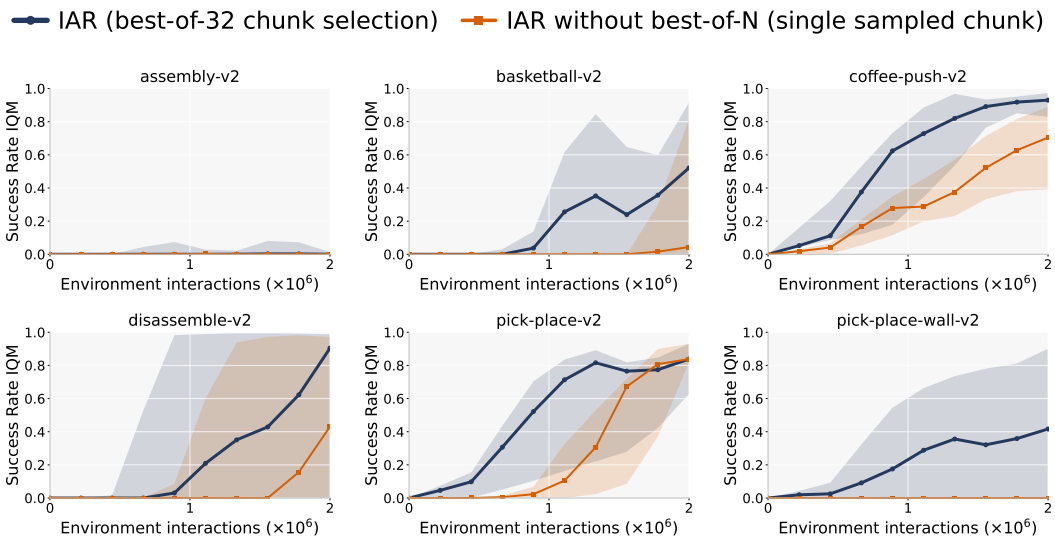



Figure 13: Best-of-32 chunk selection versus a single sampled chunk. Task-wise success-rate IQM versus environment interactions; shading shows the uncertainty bands in the plots.

### B.4 ACTOR-UPDATE HORIZON

Figure 14 compares $H_\pi \sim \text{Unif}\{1, \ldots, 16\}$, $H_\pi \sim \text{Unif}\{1, \ldots, 4\}$, $H_\pi = 1$, and $H_\pi = 16$. The respective final aggregate IQMs are approximately $58\%$, $38\%$, $1\%$, and $43\%$. The largest task-wise separation is on `disassemble-v2`: uniform sampling up to 16 reaches about $90\%$, while the other central curves remain near zero. However, shorter randomized updates and fixed-16 updates learn `coffee-push-v2` faster initially, and fixed-16 updates finish slightly higher on `pick-place-v2`. Single-action updates produce little progress beyond `coffee-push-v2`; they omit all nonempty repeated-action prefixes from the actor objective. The aggregate results favor a broad range of multi-step contexts, but do not establish that longer or randomized horizons are always better on individual tasks.

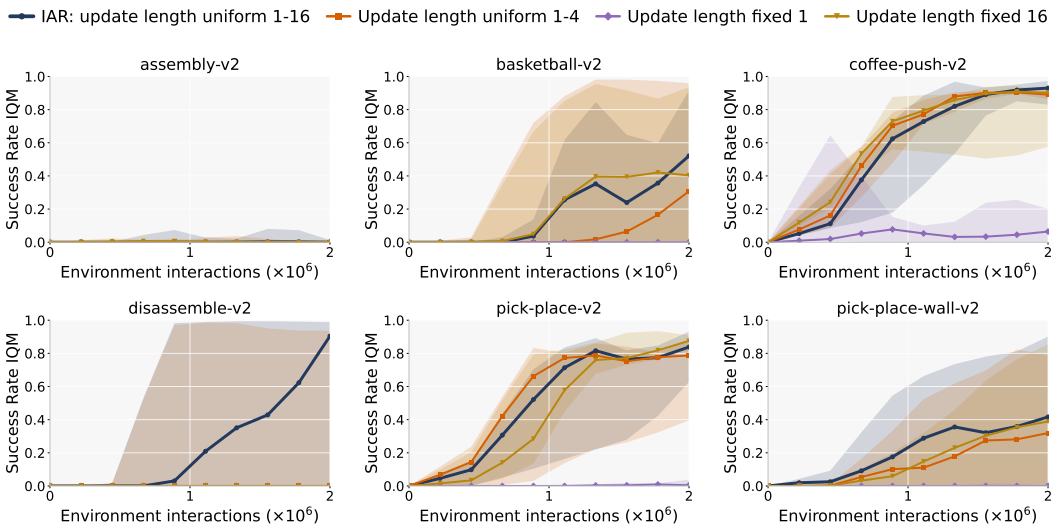



Figure 14: Actor-update horizons: uniform 1–16 (IAR), uniform 1–4, fixed 1, and fixed 16. Task-wise success-rate IQM versus environment interactions; shading shows the uncertainty bands in the plots.

### B.5 BOOTSTRAP-ACTION HORIZON

Figure 15 varies the bootstrap-action chunk length $H_t$: fixed $4$, uniform sampling from $\{1, \ldots, 4\}$, and fixed $1$. Their final aggregate IQMs are approximately $58\%$, $60\%$, and $46\%$, respectively. This is a horizon comparison, not a change in the number $N$ of candidate chunks or in the replay-prefix length used to accumulate rewards. Randomized target lengths raise the final central estimate on `assembly-v2` to about $45\%$, where both fixed settings remain near zero. Conversely, fixed-$4$ targets reach about $42\%$ on `pick-place-wall-v2`, where the other central curves are near zero. The single-action target attains the highest final central estimate on `basketball-v2`, approximately $83\%$, despite its lower aggregate result. These contrasting outcomes indicate task-dependent sensitivity to the bootstrap-action horizon.

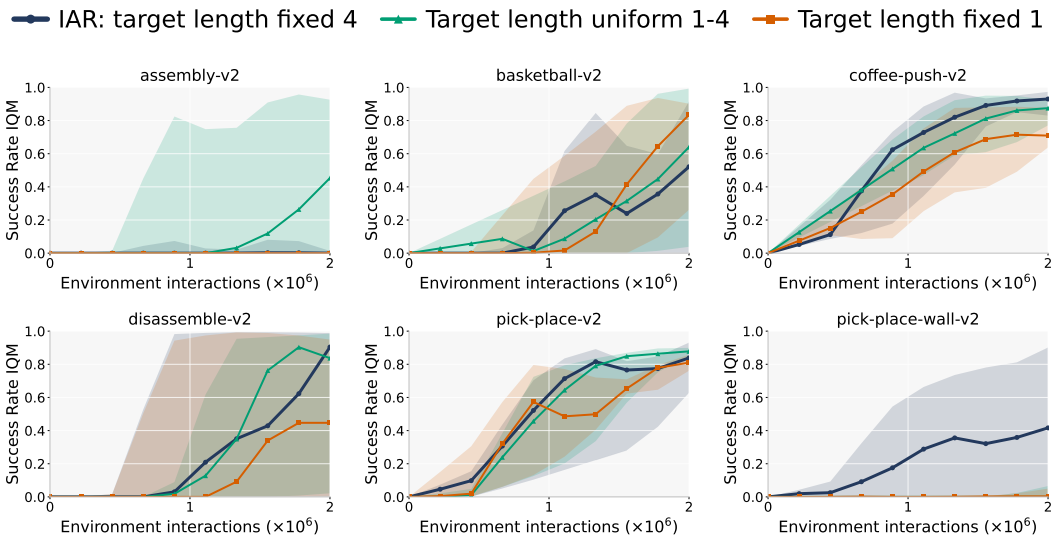



Figure 15: Bootstrap-action horizons: fixed $4$ (IAR), uniform $1$–$4$, and fixed $1$. Task-wise success-rate IQM versus environment interactions; shading shows the uncertainty bands in the plots.

### B.6 Critic Representation and Reward Scaling

Figure 16 compares the distributional critic with scaled rewards against a scalar critic with unscaled rewards. The former reaches approximately $58\%$ final aggregate IQM, compared with $34\%$ for the latter. The largest task-wise differences occur on `coffee-push-v2`, `disassemble-v2`, and `pick-place-wall-v2`. On `disassemble-v2`, the final central estimates are about $90\%$ and $47\%$, respectively; on `pick-place-wall-v2`, the scalar variant's IQM remains near zero. The final difference on `basketball-v2` is much smaller, approximately $52\%$ versus $46\%$, and the scalar variant initially learns faster on that task. Both variants remain near zero on `assembly-v2`. Because the critic representation and reward scaling change together, the observed difference cannot be attributed solely to distributional value learning.

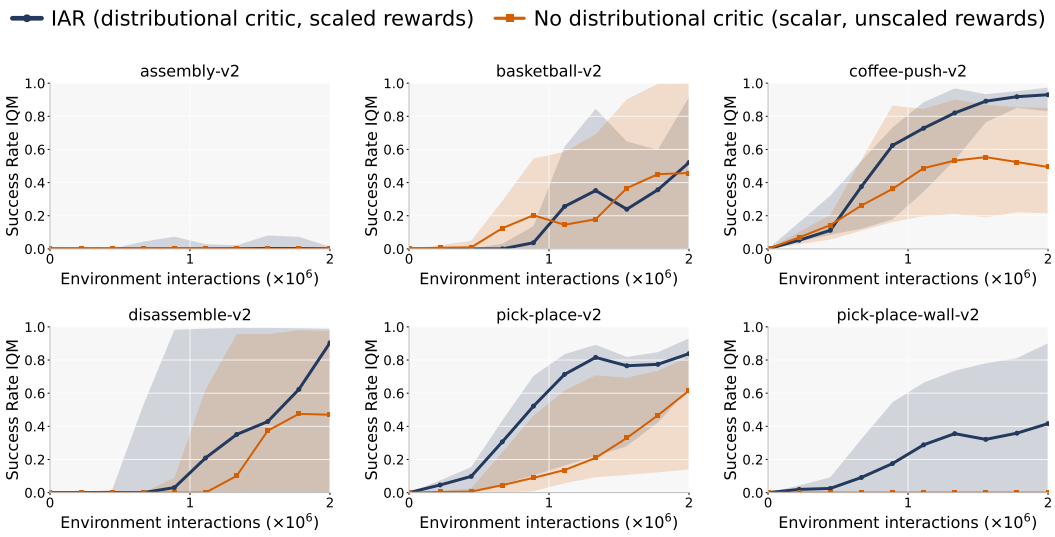



Figure 16: Distributional critic with scaled rewards versus scalar critic with unscaled rewards. Task-wise success-rate IQM versus environment interactions; shading shows the uncertainty bands in the plots.

### B.7 Policy Architecture and Learning Rate

Figure 17 compares the Transformer policy at learning rate $3 \times 10^{-5}$ with GRU and LSTM policies, each tested at $10^{-5}$, $3 \times 10^{-5}$, and $10^{-4}$. The Transformer reaches approximately $58\%$ final aggregate IQM. For both recurrent architectures, the largest tested learning rate gives the highest final aggregate result: approximately $24\%$ for GRU and $25\%$ for LSTM. At $3 \times 10^{-5}$, these values are only about $13\%$ and $11\%$, respectively. The recurrent policies' learning is concentrated on `coffee-push-v2` and `pick-place-v2`, whereas the Transformer also achieves substantial central estimates on `basketball-v2`, `disassemble-v2`, and `pick-place-wall-v2`. All architectures remain near zero on `assembly-v2`. The comparison favors the tested Transformer configuration across the recurrent learning-rate sweep, without establishing equal parameter counts, compute costs, or equally exhaustive tuning.

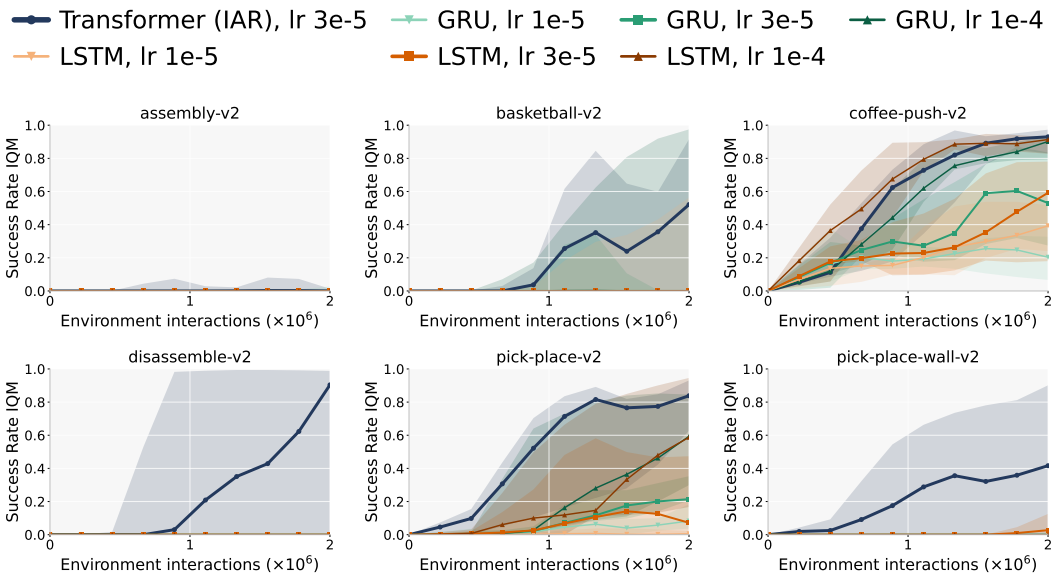



Figure 17: Transformer at learning rate $3 \times 10^{-5}$ versus GRU and LSTM at $10^{-5}$, $3 \times 10^{-5}$, and $10^{-4}$. Task-wise success-rate IQM versus environment interactions; shading shows the uncertainty bands in the plots.

**Algorithm 1:** IAR (Implicit Action Repetition)

**Input:** Horizons $(H_s, H_t, H_\pi)$; candidate count $N$; collection and update settings $(K_{\text{col}}, M, U, N_c, N_p)$; discount $\gamma$; target update rate $\tau$.

Initialize $\theta$, twin critics $\phi$, targets $\bar{\phi} \leftarrow \phi$, and replay buffer $\mathcal{D}$

**repeat**

- Collect $K_{\text{col}}$ transitions using $\text{Select}(s, H_s)$; store length-$M$ segments and clear pending actions at episode boundaries
- **if** *the replay buffer is ready* **then**
  - **for** $u \leftarrow 1$ **to** $U$ **do**
    - **Critic updates**
    - Sample replay segments and pool valid bootstrap states
    - Without gradients, select bootstrap chunks using $\text{Select}(s^+, H_t)$ and cache their averaged target-critic distributions $\bar{Z}_{\text{cache}}(s^+)$
    - Keep the cache fixed for the following $N_c$ updates
    - **for** $k \leftarrow 1$ **to** $N_c$ **do**
      - Sample valid replay windows within segment and episode boundaries; compute their $R$ and $\Gamma$
      - Form detached targets $\mathbf{q} = \Pi_{\mathcal{Z}}[R + \Gamma \bar{Z}_{\text{cache}}(s^+)]$; terminal windows use the point mass at $R$
      - Update both critics by cross-entropy with $\mathbf{q}$
      - $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$
    - **if** *policy warmup is complete* **then**
      - **Actor updates**
      - Sample a fresh batch of replay segments
      - **for** $k \leftarrow 1$ **to** $N_p$ **do**
        - Select replay states $\mathcal{S}$ and choose $H_\pi$
        - For each $s_t \in \mathcal{S}$, sample $a_t = g_{\theta,0}(s_t, \varnothing, \epsilon_0)$; set $a_t^* = a_t$ and $c_t = \text{sg}(a_t)$
        - Form $c_t^{(n)}$ from $n$ copies of $c_t$, with $c_t^{(0)} = \varnothing$
        - **if** $H_\pi > 1$ **then**
          - Evaluate continuation distributions in one parallel policy pass and sample $a_{t+n}^* = g_{\theta,n}(s_t, c_t^{(n)}, \epsilon_n)$ for $n = 1, \ldots, H_\pi - 1$
        - Evaluate separate repeated-prefix queries using the packed critic computation and form

$$\widehat{\mathcal{L}}_\pi = -\frac{1}{|\mathcal{S}| H_\pi} \sum_{s_t \in \mathcal{S}} \sum_{n=0}^{H_\pi - 1} \min_{i \in \{1,2\}} Q_{\phi_i}(s_t, c_t^{(n)}, a_{t+n}^*).$$

        - Update $\theta$ with critic parameters fixed; each term differentiates only through its fresh action

**until** *the training budget is exhausted*

## C PSEUDOCODE

Algorithm 1 summarizes the learning procedure with categorical critics described in Section 4. Both actor updates and critic targets omit entropy regularization ($\alpha = 0$). For a state $s$ and horizon $H$, let $\text{Select}(s, H)$ draw $N$ independent autoregressive candidates $\mathbf{a}^{(j)} = f_\theta^{(H)}(s, \epsilon^{(j)})$ and return

$$\text{Select}(s, H) = \mathbf{a}^{(j^\star)}, \qquad j^\star = \arg \max_{1 \le j \le N} \min_{i \in \{1,2\}} Q_{\phi_i}(s, \mathbf{a}^{(j)}).$$

Setting $N = 1$ recovers sampling without candidate ranking. We denote the configured total number of new primitive transitions collected per iteration by $K_{\text{col}}$, summed across collection environments, and distinguish it from the stored segment length $M$.

Here $Q(s, c, a)$ evaluates the concatenation of the prefix $c$ and the single fresh action $a$. Continuation samples from different positions are not concatenated into a shared action chunk. The stop-gradient operator is applied only to the repeated context; the $n = 0$ term retains the gradient through $a_t$. Fixing critic parameters during actor optimization preserves differentiation through the critic with respect to each fresh action.

The cached random variable $\bar{Z}_{\mathrm{cache}}(s^+)$ has probabilities $\bar{\mathbf{p}}_{\mathrm{cache}}(s^+)$ on $\mathcal{Z}$. This distribution is cached before adding rewards, applying discounts, or projecting. Each replay window combines the cached distribution with its own $R$ and $\Gamma$; target-network updates do not modify the cache within the current critic phase.

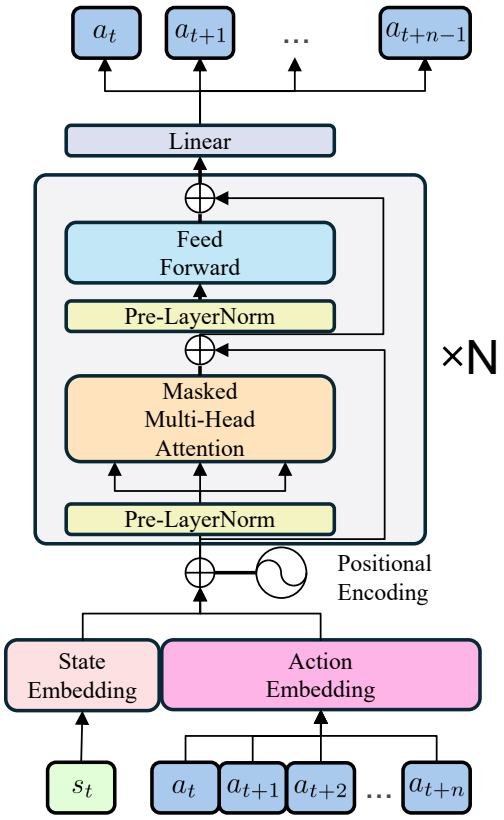



Figure 18: **Architecture of the IAR Transformer policy.** State and action embeddings are combined with positional encodings and processed by a stack of causal, pre-layer-normalized Transformer blocks. A shared linear readout provides the action-prediction output at each position; stochastic sampling is omitted from the schematic. Predictions are shifted by one position: the state token predicts the first action, and each action token predicts the next action. We denote the stack depth by $L_\pi$; the $N$ inside the schematic denotes this depth, independently of the best-of-$N$ candidate count.

## D Transformer Policy Architecture

**Tokenization and prediction alignment.** The policy represents the conditional distributions $\pi_{\theta,n}(\cdot \mid s_t, \mathbf{a}_{t:t+n-1})$ in Eq. (6). For a prediction horizon $H$, the input sequence is $(s_t, a_t, \ldots, a_{t+H-2})$, containing one state token and $H-1$ action tokens. Separate state and action embeddings map these inputs to a common feature dimension:

$$h_0^{(0)} = E_s(s_t) + p_0, \qquad h_n^{(0)} = E_a(a_{t+n-1}) + p_n, \quad n = 1, \ldots, H-1, \tag{14}$$

where $p_n$ denotes the positional encoding at position $n$. Output position 0 parameterizes the prediction of $a_t$ from $s_t$, while position $n \geq 1$ predicts $a_{t+n}$ conditioned on $s_t$ and the preceding $n$ actions. Thus, the action being predicted is not included in its own conditioning prefix. No intermediate environment states are required as policy inputs.

**Causal Transformer blocks.** As shown in Fig. 18, each block applies layer normalization before masked multi-head self-attention and before the position-wise feed-forward network, with a residual connection around each sublayer. Writing $\mathbf{h}^{(\ell)}$ for the token representations entering block $\ell$, we have

$$\begin{aligned} \widetilde{\mathbf{h}}^{(\ell)} &= \mathbf{h}^{(\ell)} + \mathrm{MHA}_\ell\Big(\mathrm{LN}_{\ell,1}(\mathbf{h}^{(\ell)}); \mathcal{C}\Big), \\ \mathbf{h}^{(\ell+1)} &= \widetilde{\mathbf{h}}^{(\ell)} + \mathrm{FFN}_\ell\Big(\mathrm{LN}_{\ell,2}(\widetilde{\mathbf{h}}^{(\ell)})\Big), \quad \ell = 0, \ldots, L_\pi - 1. \end{aligned} \tag{15}$$

The causal mask $\mathcal{C}$ allows position $n$ to attend only to positions $0, \ldots, n$. Consequently, its output depends only on the initial state and the corresponding action prefix. The linear readout is shared across positions. We use $g_{\theta,n}$ for the resulting reparameterized action sampler, consistent with Sec. 4.1; all positions share the policy parameters $\theta$.

**Autoregressive interaction and bootstrapping.** During environment interaction and critic bootstrapping, the policy first samples $a_t$ from the state-only conditional distribution. Each sampled action is then appended to the input prefix to obtain the next conditional distribution. This produces autoregressive chunks of lengths $H_s$ and $H_t$, respectively. Although the network can evaluate all positions in parallel when their input prefixes are available, generating a new chunk remains sequential because later prefixes contain earlier sampled actions.

**Parallel predictions for IAR updates.** For an actor-update state $s_t$, we first sample $a_t$ and form $c_t = \mathrm{sg}(a_t)$. We then provide the fixed token sequence $(s_t, c_t, \ldots, c_t)$, containing $H_\pi - 1$ copies of $c_t$. Under the same causal mask, position $n$ conditions on exactly $n$ copies, yielding

$$a^*_{t+n} = g_{\theta,n}\left(s_t, \mathbf{c}_t^{(n)}, \epsilon_n\right), \qquad n = 1, \ldots, H_\pi - 1, \tag{16}$$

with $a^*_t = a_t$ and $\mathbf{c}_t^{(n)} = (c_t, \ldots, c_t)$, taking $\mathbf{c}_t^{(0)} = \varnothing$. Once the initial action is available, all remaining conditional distributions can be evaluated in one parallel Transformer forward pass. The fresh continuation samples are not fed back into this pass. Positional encodings distinguish different prefix lengths even though their action tokens have identical values.

Each sample is evaluated by the critic as $\widehat{Q}_\phi(s_t, \mathbf{c}_t^{(n)}, a^*_{t+n})$, as in Eq. (9); the continuation samples are not concatenated into a single chunk for this objective. The repeated context is detached and critic parameters are fixed, while gradients flow through each freshly sampled action. In particular, the $n = 0$ term retains the gradient through $a_t$. Implicit repetition therefore changes the conditioning inputs used for actor optimization while preserving autoregressive generation for interaction and bootstrapping.

**Architecture settings.** Table 4 reports the reference policy depths, attention-head counts, embedding widths, and separate variance-MLP sizes. The variance network and stochastic sampling step are not expanded in Fig. 18. The listed configurations use layer normalization and zero Transformer dropout. The policy remains stochastic.

# E COMPUTATIONAL COSTS

We compare the training time and model size of IAR, CrossQ, SimbaV2, and T-SAC for Meta-World. The timing platform is a Lenovo IdeaCentre Gaming 5 14ACN6 desktop with a single NVIDIA GeForce RTX 3060 GPU (12 GB of GPU memory, Ampere architecture, and 3,584 CUDA cores), a 6-core, 12-thread CPU with a maximum frequency of 4.46 GHz, and 16 GB of RAM. The system runs Ubuntu 24.04 with Linux kernel `6.8.0-136-generic` and glibc 2.39. The NVIDIA driver reports support for CUDA 12.2.

Table 2 reports training time per $10^6$ environment interactions and the actor and critic parameter counts recorded for each run. Each environment interaction corresponds to one primitive environment transition, irrespective of the action-chunk length; transitions are summed across parallel environments. The SimbaV2 timing run uses a UTD ratio of 2, matching its cluster configuration.

Table 2: Meta-World training times and model sizes. Times are reported per $10^6$ primitive environment interactions. Actor and critic parameter counts are reported as exact integers.

| Method | Time (h/1M) | Actor parameters | Critic parameters |
|---|---|---|---|
| CrossQ | 2.10 | 79,190 | 8,593,582 |
| SimbaV2 | 3.25 | 171,016 | 4,536,933 |
| T-SAC | 5.80 | 45,320 | 1,061,120 |
| IAR (ours) | 5.26 | 81,668 | 1,086,720 |

IAR has 1,168,388 actor and critic parameters in total, compared with 8,672,772 for CrossQ, 4,707,949 for SimbaV2, and 1,106,440 for T-SAC. Despite its smaller model than CrossQ and SimbaV2, IAR takes approximately $2.50\times$ and $1.62\times$ as long, respectively. Relative to the reported T-SAC time, IAR takes approximately $9.3\%$ less time, with $5.6\%$ more actor and critic parameters in total. Time comparisons use the unrounded second counts where available.

These run-level timings characterize the reported configurations; the methods are not matched in UTD ratio or parameter budget. No component-wise timing breakdown or cross-run timing variability is available. Parameter counts are used as recorded for the actor and critic, without an additional multiplier for critic networks. Model FLOPs are not reported and cannot be inferred from wall-clock time or parameter counts. Corresponding cost measurements for SAC and dense Box Pushing are unavailable. The reported times alone do not establish the time needed to reach a common success threshold.

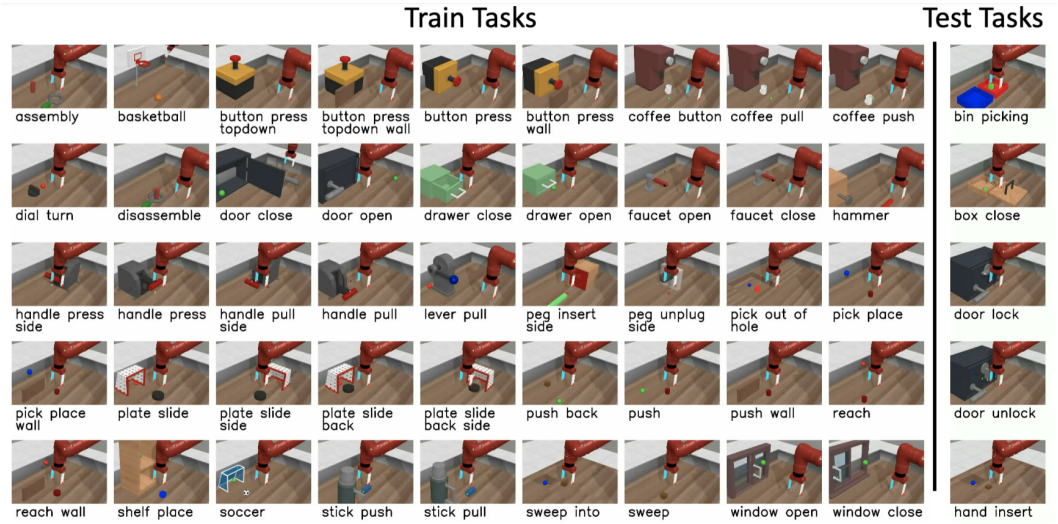



Figure 19: Meta-World tasks (Yu et al., 2020).

### E.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 19 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.

### E.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.

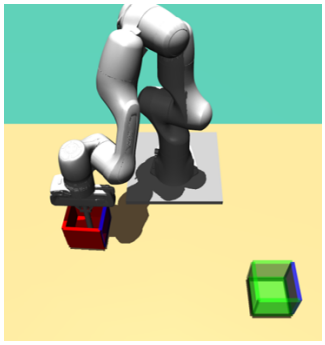


Figure 20: Box Pushing task (Otto et al.).

**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 $\boldsymbol{a}_t \in \mathbb{R}^7$.

**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

$$R_{\text{rod}} = \text{clip}(\|\boldsymbol{p} - \boldsymbol{h}_{\text{pos}}\|,\, 0.05,\, 10) + \text{clip}\big(\tfrac{2}{\pi}\arccos|\boldsymbol{h}_{\text{rot}} \cdot \boldsymbol{h}_0|\,,\, 0.25,\, 2\big)\,, \tag{17}$$

$$\tau_t = \sum_{i=1}^{7} (a_t^i)^2, \tag{18}$$

$$\text{err}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_{i:\,|q_i| > |q_i^b|} \big(|q_i| - |q_i^b|\big) + \sum_{j:\,|\dot{q}_j| > |\dot{q}_j^b|} \big(|\dot{q}_j| - |\dot{q}_j^b|\big)\,, \tag{19}$$

$$R_{\text{position}} = \|\boldsymbol{p} - \boldsymbol{p}_{\text{target}}\|, \tag{20}$$

$$R_{\text{rotation}} = \tfrac{1}{\pi}\arccos|\boldsymbol{r} \cdot \boldsymbol{r}_{\text{target}}|\,. \tag{21}$$

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

Table 3: Selected T-SAC baseline settings.

| Setting | Meta-World ML1 | Dense Box Pushing |
|---|---|---|
| Critic embedding dimension | 128 | 256 |
| Actor hidden layers | $2 \times 128$ | $4 \times 512$ |
| Batch size | 512 | 256 |
| Entropy coefficient | Automatic tuning | 0 |
| Update-to-data ratio | 1 | 0.25 |

# F ALGORITHM IMPLEMENTATIONS

This appendix describes the implementation choices and random-seed counts for IAR and the comparison methods. Benchmark-specific settings are specified below; detailed architecture and optimization hyperparameters are reported in Appendix G.

**IAR** Implicit Action Repetition (IAR) constructs repeated-action contexts during actor optimization. For each replay state, the policy samples an initial action and repeats a stop-gradient copy of it to form prefixes of different lengths. The policy then predicts the continuation distributions in parallel and draws a fresh action for each prefix. Each prefix and its fresh continuation are evaluated as a separate sequence-critic query, and the actor objective averages these values together with the value of the initial action. Critic parameters and repeated prefixes are held fixed during differentiation, so each term updates the actor only through its fresh action; the zero-length-prefix term retains the gradient through the initial action. Environment interaction and bootstrap action generation remain autoregressive, with critic-guided candidate selection where enabled. We set the entropy coefficient to zero in both the actor objective and critic targets. **Seeds: 5 for the main results; 4 for the ablation studies.**

**T-SAC** Transformer-based Soft Actor–Critic (T-SAC) retains a one-step Gaussian actor and learns sequence-conditioned values with a causal Transformer critic using multi-step return targets. The critic has two layers and four attention heads. For Meta-World ML1 and dense Box Pushing, we use AdamW with actor and critic learning rates of $2.5 \times 10^{-4}$ and $2.5 \times 10^{-5}$, respectively, return discount $\gamma = 0.99$, and target update coefficient $\tau = 0.005$. Table 3 summarizes the benchmark-specific settings. **Seeds: 8.**

**PPO** Proximal Policy Optimization (PPO) (Schulman et al., 2017) is an on-policy, step-based method that discourages large changes from the behavior policy. Two common variants are *PPO-Penalty*, which uses KL regularization, and *PPO-Clip*, which uses a clipped surrogate objective. We evaluate PPO-Clip using 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 method with twin Q-networks to mitigate overestimation and entropy regularization to encourage exploration. We use the SAC implementation distributed with CrossQ (Bhatt et al., 2024). **Seeds: 20 (Meta-World ML1), 5 (Gym MuJoCo).**

**TD3** Twin Delayed DDPG (TD3) (Fujimoto et al., 2018) addresses overestimation and instability through clipped double Q-learning, delayed policy updates, and target policy smoothing. Our implementation is adapted from Raffin et al. (2021), with 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 with GTrXL implementation in RLlib (Liang et al., 2018), adding minibatch advantage normalization and 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 independent Gaussian action noise with state-dependent, temporally coherent exploration. The perturbation is $\epsilon_t = \Theta h_t$, where $h_t$ is the policy's last hidden-layer activation and $\Theta$ is a Gaussian noise matrix resampled according to the SDE sampling frequency. We evaluate gSDE with PPO using the reference implementation of Raffin et al. (2022),

resampling the noise matrix every four steps. For stability on some tasks, we use 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 over movement-primitive parameters, building on probabilistic movement primitives (ProMPs) (Paraschos et al., 2013). This formulation supports sparse and non-Markovian rewards. We use ProDMP (Li et al., 2023) as the trajectory generator and consider both diagonal-covariance (**BBRL-Std**) and full-covariance (**BBRL-Cov**) Gaussian policies over primitive parameters. **Seeds: 20.**

**TCP** TCP denotes the temporally correlated episodic RL baseline of Li et al. (2024) in our comparisons. It augments episodic policy updates with step-level learning signals while retaining smooth, parameter-space exploration, combining trajectory-level structure with finer-grained credit assignment. **Seeds: 20.**

**TOP-ERL** Transformer-based Off-Policy Episodic Reinforcement Learning (TOP-ERL) uses a Transformer critic to evaluate action subsequences and learns from multi-step return targets for off-policy episodic updates. Our implementation uses ProDMP (Li et al., 2023) as the trajectory generator and optimizes a distribution over movement-primitive parameters with a KL-constrained trust region and temporally structured exploration. Unless stated otherwise, we use an adaptive scale (entropy) schedule and per-dimension normalization of primitive parameters. **Seeds: 8.**

**CrossQ** CrossQ (Bhatt et al., 2024) is an off-policy SAC variant that removes target networks and uses batch renormalization (BRN), while retaining twin critics. We follow the authors' reference implementation with an update-to-data ratio of $\mathrm{UTD} = 1$, automatic temperature tuning, and the recommended hyperparameters unless stated otherwise. **Seeds: 4 (Meta-World ML1), 5 (Gym MuJoCo and Box Pushing).** Training on Box Pushing was capped at $10^7$ environment steps due to the experiment budget; by that point, wall-clock training time exceeded 24 hours.

**RLAC (Q-Chunking)** Reinforcement Learning with Action Chunking (Li et al., 2026b) introduces Q-chunking, denoted here as RLAC, which performs reinforcement learning in a temporally extended action space. The policy generates a sequence of actions for open-loop execution, and the critic evaluates the entire sequence conditioned on the initial state. Conditioning the critic on all actions in the chunk supports multi-step temporal-difference backups without the intermediate-action off-policy bias of conventional multi-step targets for a single-action critic. In its original offline-to-online formulation, the method also uses a behavior constraint learned from prior data to encourage temporally coherent exploration. **Seeds: 5 for MetaWorld, 4 for Box-Pushing.**

**SimbaV2** SimbaV2 (Lee et al., 2025) is an architecture built on SAC. It uses hyperspherical normalization to control the norms of features and weights, and combines a categorical distributional critic with reward scaling to stabilize optimization across reward scales. These design choices support scaling the model size and training computation. **Seeds: 5 for MetaWorld, 4 for Box-Pushing.**

# G HYPERPARAMETERS OF THE ALGORITHM

This appendix covers Meta-World and dense Box Pushing. The IAR tables report the shared reference configurations of the supplied implementation; experiment-specific launcher overrides are not included. T-SAC is treated as a baseline, with its published settings reported separately.

**IAR architecture and action horizons.** Table 4 summarizes the policy, critic, and action-generation settings. Here $H_s$, $H_t$, and $H_\pi$ denote the action horizons for environment interaction, bootstrapping, and actor updates, respectively. IAR repeats a stop-gradient action token during actor updates; environment interaction and bootstrap action generation use autoregressive chunks.

Table 4: IAR architecture and action-generation reference settings. Transformer dimensions are listed as layers / heads / embedding width.

| Hyperparameter | Meta-World | Dense Box Pushing |
|---|---|---|
| Policy Transformer | $2/1/32$ | $4/4/256$ |
| Critic Transformer | $2/4/128$ | $2/4/256$ |
| Variance MLP hidden layers | $2 \times 128$ | $2 \times 128$ |
| Layer normalization | Enabled | Enabled |
| Transformer dropout | 0 | 0 |
| Actions per policy output position | 1 | 1 |
| Rollout horizon $H_s$ | 4 | 4 |
| Bootstrap horizon $H_t$ | 4 | 4 |
| Actor-update horizon $H_\pi$ | Uniform on $\{1, \ldots, 16\}$ | Uniform on $\{1, \ldots, 16\}$ |
| Best-of-$N$ chunk selection | $N = 32$ | Disabled |
| Critic output | Categorical | Scalar |
| Categorical atoms / support | 101 / $[-5, 5]$ | Not applicable |
| Return-based reward scaling | Enabled | Disabled |
| Bootstrap twin-critic aggregation | Mean | Minimum |

**IAR optimization and replay.** Both configurations use AdamW, weight decay $10^{-5}$ on decayed parameter groups, and actor optimizer coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$. The actor's temporal discount factor is one. Learning-rate schedules, observation normalization, gradient clipping, and trust-region updates are disabled. Table 5 lists the remaining settings.

Table 5: IAR optimization reference settings. Replay capacity is measured in segments; warmup is measured in learner iterations.

| Hyperparameter | Meta-World | Dense Box Pushing |
|---|---|---|
| Actor learning rate | $3 \times 10^{-5}$ | $3 \times 10^{-5}$ |
| Critic learning rate | $2.5 \times 10^{-5}$ | $3 \times 10^{-5}$ |
| Critic optimizer $(\beta_1, \beta_2)$ | $(0.85, 0.95)$ | $(0.9, 0.999)$ |
| Return discount $\gamma$ | 0.99 | 0.98 |
| Batch size | 512 | 256 |
| Replay capacity (segments) | 100,000 | 20,000 |
| Stored segment length $M$ | 100 | 100 |
| Critic window length | Uniform on $\{1, \ldots, 16\}$ | Uniform on $\{1, \ldots, 16\}$ |
| Update rounds per iteration $U$ | 5 | 2 |
| Critic window updates per round $N_c$ | 100 | 100 |
| Actor updates per round $N_p$ | 40 | 20 |
| Actor warmup (iterations) | 200 | 100 |
| Critic warmup (iterations) | 0 | 0 |
| Target-copy coefficient $\tau$ | 1 | 1 |

The values of $N_c$ and $N_p$ above are the effective counts for 100-step segments. In dense Box Pushing, the critic-start count is bounded by the episode horizon and segment length, while actor starts are sampled at a ratio of 0.2. Each update round samples a critic batch and a fresh actor batch. Critic windows are truncated at segment ends. Target critics are hard-copied after each critic update, and learning also requires the replay buffer to be ready.

**T-SAC baseline.** Table 6 records the published T-SAC configuration Tian et al. (2026). These values are distinct from the IAR settings above. Unlisted T-SAC hyperparameters follow the original paper.

Table 6: Published T-SAC baseline hyperparameters.

| Hyperparameter | Meta-World | Dense Box Pushing |
|---|---|---|
| Optimizer | AdamW | AdamW |
| Actor learning rate | $2.5 \times 10^{-4}$ | $2.5 \times 10^{-4}$ |
| Critic learning rate | $2.5 \times 10^{-5}$ | $2.5 \times 10^{-5}$ |
| Return discount $\gamma$ | 0.99 | 0.99 |
| Target update coefficient $\tau$ | 0.005 | 0.005 |
| Actor hidden layers | $2 \times 128$ | $4 \times 512$ |
| Critic layers / heads / width | $2/4/128$ | $2/4/256$ |
| Batch size | 512 | 256 |
| Replay capacity (segments) | 5,000 | 20,000 |
| Stored segment length | 125 | 100 |
| Samples collected per iteration | $4 \times 125$ | $4 \times 100$ |
| Entropy coefficient | Automatic tuning | 0 |
| Update-to-data ratio | 1 | 0.25 |

**Other baselines.** For PPO, SAC, GTrXL, gSDE, BBRL, TCP, TOP-ERL, and CrossQ, we use the same benchmark-specific baseline configurations as those used in the T-SAC study Tian et al. (2026). For RLAC (Q-Chunking) Li et al. (2026b) and SimbaV2 Lee et al. (2025), we use the authors' officially released code and configurations.