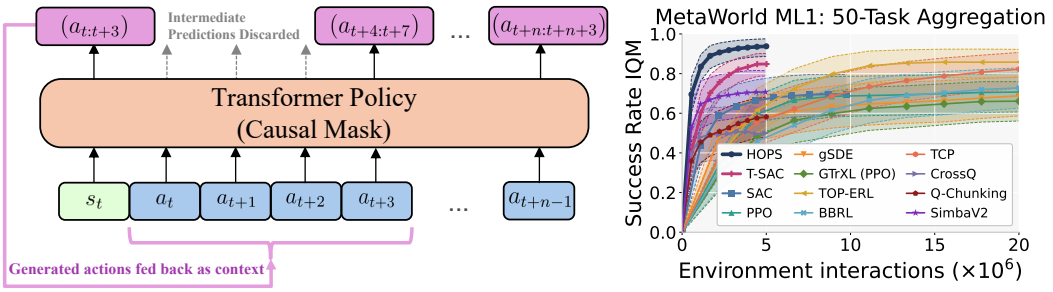
CHUNKING THE POLICY: EXTENDING THE POLICY-UPDATE HORIZON WITH A COMPACT TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce HOPS (Horizon-Extended Optimization over Policy-Generated Sequences), an off-policy actor–critic method that independently controls the policy-update horizon while keeping interaction-time prediction and execution horizons fixed. A compact Transformer autoregressively composes action chunks of length C, a task-dependent hyperparameter, into longer sequences for actor optimization while retaining short prediction and execution horizons during interaction. In the default Meta-World configuration, $C = 4$: actor updates optimize sequences of up to sixteen actions, while interaction predicts and executes H_{exec} (default: 4) actions before replanning. Twin prefix-conditioned critics provide value gradients across the generated sequence. We train separate agents online from scratch on all 50 Meta-World ML1 tasks and dense-reward FANCY GYM Box Pushing, using final-timestep success for evaluation. HOPS achieves a mean task-wise success IQM of 0.938 on Meta-World at 5×10^6 environment interactions, compared with 0.848 for T-SAC, the previous State-of-the-art baseline. On Box Pushing, HOPS reaches a success IQM of 0.941 at 8×10^6 interactions, compared with 0.637 for T-SAC. Controlled ablations on a 20-task Meta-World subset vary the maximum policy-update horizon while holding the interaction horizons and critic training fixed. Extending the maximum update horizon from four to sixteen improves aggregate performance, with task-dependent effects. These results support extending the temporal scope of actor optimization through chunk-autoregressive generation while retaining short action chunks for interaction.



(a) Policy Transformer overview

(b) Metaworld ML1 aggregation results

Figure 1: Overview of HOPS and aggregate performance on all 50 Meta-World ML1 tasks. (a) At each actor update, HOPS autoregressively composes action chunks of length C into an H_{upd}-step sequence using causal masking, where C is a task-dependent hyperparameter. The schematic illustrates the Meta-World default $C = 4$. Chunk-boundary representations condition subsequent chunks, while twin prefix-conditioned critics evaluate the resulting sequence. (b) Unweighted mean of the 50 task-wise terminal-success IQMs. Across all tasks, initial object positions are randomized, while the robot's initial configuration is fixed. Success is determined solely at the final timestep of each episode; success achieved only at earlier timesteps is not counted. Each task's IQM is computed across seed-level terminal-success rates, with all tasks weighted equally in the aggregate. Shading denotes the reported 95% confidence intervals.

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.

1 INTRODUCTION

Temporally coordinated actions are essential for robotic manipulation. Maintaining contact, aligning an object, and completing a sequence of manipulation stages all require actions whose effects remain compatible over time (Zhao et al., 2023). Conventional Q-based actor–critic methods, such as soft actor–critic (SAC), improve a policy by evaluating a single policy-generated action at a given state (Haarnoja et al., 2018a). Their critics summarize long-term returns, while the actor objective directly optimizes the sampled action. Action chunking introduces temporal structure by predicting several future actions together, providing a natural representation for short sequences of coordinated behavior (Zhao et al., 2023; Chi et al., 2025). This raises a further question: how many policy-generated actions should be jointly optimized during learning?

Answering this question requires distinguishing three horizons. The *prediction horizon*, H_{pred}, specifies how many actions are proposed during interaction; the *execution horizon*, H_{exec}, specifies how many are applied before replanning; and the *policy-update horizon*, H_{upd}, specifies how many consecutive policy-generated actions are jointly evaluated during actor optimization. These horizons serve different purposes. Prediction and execution determine how the agent interacts with its environment, whereas the policy-update horizon determines the temporal extent of the generated sequence directly optimized by the actor. It is also distinct from the rollout horizon and the return horizon used to train a critic. In particular, extending critic supervision over longer replay sequences does not itself extend the sequence of policy-generated actions optimized in an actor update. For example, T-SAC trains a sequence-conditioned critic with multi-step targets while retaining a one-action actor update (Tian et al., 2026).

This distinction suggests that temporal coordination during learning can be developed beyond the action chunk used for interaction. Increasing the execution horizon makes the agent act for longer before incorporating a new observation, as in receding-horizon action-sequence policies (Chi et al., 2025). Extending the policy-update horizon instead allows the learning objective to evaluate how generated actions work together across a longer sequence, while preserving a short replanning interval. The resulting design question is whether longer joint policy updates can improve learning when the interaction-time prediction and execution horizons remain fixed.

We introduce HOPS (Horizon-Extended Optimization over Policy-Generated Sequences), an off-policy actor–critic method trained entirely online from scratch. Its central design is to control the policy-update horizon through autoregressive composition of action chunks of length C, while keeping the interaction-time prediction and execution horizons fixed for a given configuration. The chunk length C is a task-dependent hyperparameter that remains fixed within each experimental configuration. A compact causal Transformer (Vaswani et al., 2017) generates C-action chunks, conditioning each subsequent chunk on the initial state and previously generated actions. During actor optimization, HOPS composes these chunks into longer sequences and optimizes their prefix values using twin prefix-conditioned critics. Generated actions remain differentiable as critic inputs, while their copies used to condition subsequent chunks are detached. In the default Meta-World configuration, $C = 4$: actor updates use sequences of four, eight, twelve, or sixteen actions, while interaction predicts and executes four actions before replanning, with $H_{\text{pred}} = H_{\text{exec}} = 4$.

We evaluate HOPS on all 50 Meta-World ML1 tasks (Yu et al., 2020) and dense-reward FANCY GYM Box Pushing (Otto et al.), training a separate agent for each task without offline data or pre-training. The benchmark comparisons show improved aggregate terminal-success performance over T-SAC (Tian et al., 2026) under the evaluated configurations. To examine the policy-update horizon within HOPS, we compare $H_{\max} \in \{4, 8, 16\}$ on a 20-task Meta-World subset while holding the interaction horizons and critic training fixed. Longer maximum update horizons improve aggregate performance in this study, although individual tasks respond differently.

Our contributions are threefold:

- We formulate policy-update horizon control under fixed interaction-time prediction and execution horizons, separating the temporal scope of actor optimization from interaction and critic-training horizons.
- We develop HOPS, which implements this separation through a compact chunk-autoregressive Transformer actor and twin prefix-conditioned critics, enabling longer

policy-generated sequences to be optimized while retaining short interaction horizons, with a task-dependent chunk length C.

- We evaluate HOPS across all 50 Meta-World ML1 tasks and on dense-reward Box Pushing, and provide controlled horizon ablations on 20 Meta-World tasks together with task-subset studies of other design choices.

2 PRELIMINARIES

Reinforcement Learning and Soft Actor-Critic. We consider a discounted Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$ (Sutton & Barto, 1998; León & Kiencke, 2013). Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) learns a stochastic policy π_ϕ and twin critics $Q_{\theta_1}, Q_{\theta_2}$ from a replay buffer $\mathcal{D}$, with actor objective

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_{\substack{s_t \sim \mathcal{D}, \\ a_t \sim \pi_\phi(\cdot|s_t)}} \left[\alpha \log \pi_\phi(a_t \mid s_t) - \min_{i \in \{1,2\}} Q_{\theta_i}(s_t, a_t) \right], \tag{1}$$

where $\alpha > 0$ controls entropy regularization (Haarnoja et al., 2018b).

Causal Transformers and T-SAC. Causal Transformers model sequence dependencies through masked self-attention and positional embeddings (Vaswani et al., 2017).

Transformer-based SAC (T-SAC) (Tian et al., 2026) uses this structure to evaluate action prefixes from replay. Let $H_Q \in \mathbb{N}_{\geq 1}$ denote the critic-update horizon, i.e., the maximum replay action-prefix length used in a critic update. The critics predict $Q^{(h)}_{\theta_i}(s_t, \mathbf{a}^{(h)}_t)$ for each replay prefix $\mathbf{a}^{(h)}_t = (a_t, \ldots, a_{t+h-1})$, $h \leq H_Q$. Each prefix receives a non-soft target

$$y^{(h)}_t = \sum_{k=0}^{h-1} \gamma^k r_{t+k} + \gamma^h \bar{V}(s_{t+h}), \tag{2}$$

where $\bar{V}$ is the non-entropy bootstrap value under the current policy, set to zero at terminal states. Each critic minimizes

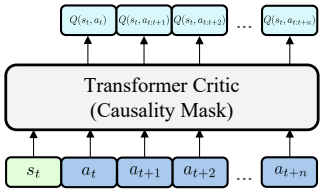


Figure 2: Prefix-conditioned Transformer critic. Causal attention produces a value for each action prefix.

$$\mathcal{L}_{Q_i}(\theta_i) = \mathbb{E}_{\mathcal{D}, H_Q} \left[\frac{1}{H_Q} \sum_{h=1}^{H_Q} \left(Q^{(h)}_{\theta_i}(s_t, \mathbf{a}^{(h)}_t) - y^{(h)}_t \right)^2 \right], \tag{3}$$

with targets held fixed during differentiation. Entropy regularization applies only to the actor, whose update evaluates a single policy-generated action through $Q^{(1)}(s_t, a_t)$, even when the critic is trained on longer replay prefixes.

Best-of-N Action Selection. Q-chunking (Li et al., 2026) selects high-value action chunks from a behavior-cloned flow policy $f_\xi(\cdot \mid s)$, which learns the distribution of action chunks in the training data. For $N \in \mathbb{N}_{\geq 1}$ independent length-h candidates,

$$\begin{aligned} \mathbf{a}^{(n)} &\overset{\text{i.i.d.}}{\sim} f_\xi(\cdot \mid s), \qquad n = 1, \ldots, N, \\ n^\star &\in \operatorname*{arg\,max}_{n \in \{1,\ldots,N\}} Q^{(h)}_\theta(s, \mathbf{a}^{(n)}), \qquad \mathbf{a}^\star = \mathbf{a}^{(n^\star)}. \end{aligned} \tag{4}$$

With uniform tie-breaking, the selected-chunk distribution π_{BoN} satisfies

$$D_{\text{KL}}(\pi_{\text{BoN}}(\cdot \mid s) \,\|\, f_\xi(\cdot \mid s)) \leq \log N - \frac{N-1}{N}. \tag{5}$$

Thus, $N = 1$ recovers direct sampling, while larger N permits stronger critic-guided selection.

3 RELATED WORK

We categorize prior work by where temporal extension is introduced: critic inputs, policy outputs and updates, model-based lookahead, or credit assignment.

Sequence-conditioned critics. TOP-ERL evaluates movement-primitive segments with a Transformer critic and prefix-wise n-step targets (Li et al., 2025). T-SAC adapts this critic to SAC, but its one-step MLP actor is updated only through $Q_\psi^{(1)}(s_t, a_t)$, so $H_{\text{upd}} = 1$ despite longer critic supervision (Tian et al., 2026). CQN-AS is instead a critic-only, coarse-to-fine method that selects discretized action sequences (Seo & Abbeel, 2026); without a reparameterized continuous actor, its sequence length is not an actor-update horizon under our definition. HOPS retains T-SAC's twin prefix critics but jointly optimizes longer policy-generated sequences with a chunk-autoregressive actor.

Action chunking and temporal abstraction. Options and learned variable-length routines provide temporally extended decisions, including in online off-policy RL (Sutton et al., 1999; Cetin & Celiktutan, 2021). In imitation learning, ACT predicts action chunks but temporally ensembles one executed action, while Diffusion Policy predicts and executes different-length prefixes under receding-horizon control (Zhao et al., 2023; Chi et al., 2025); these distinguish H_{pred} from H_{exec}, but have no RL actor-update horizon. Among chunked RL methods, Q-chunking evaluates fixed policy chunks offline-to-online (Li et al., 2026), DEAS learns fixed-H sequence values with detached critic learning offline (Kim et al., 2026), and AC3 uses demonstrations, fixed continuous chunks, and success-only actor updates (Yang et al., 2026). HOPS needs neither offline data nor demonstrations. Closest concurrent work, SEAR, also learns online from scratch with twin causal Transformer critics and prefix-wise multi-horizon targets, but fixes an MLP actor's jointly optimized chunk length $H_{\text{upd}} = N$ and uses random-prefix collection with receding-horizon execution (Nagy et al., 2026). HOPS instead autoregressively composes chunks of C actions, where C is a task-dependent hyperparameter, and varies the policy-update horizon while holding the interaction horizons fixed in the corresponding controlled ablations. In the default Meta-World configuration, $C = 4$, $H_{\text{upd}} \in \{4, 8, 12, 16\}$, and $H_{\text{pred}} = H_{\text{exec}} = 4$.

Multi-step improvement, models, and credit. Multi-step policy improvement is related but operationally distinct: h-greedy policy iteration uses h-step lookahead to select the improved policy's first action, whereas DoMo-AC (Tang et al., 2023)differentiates multi-step off-policy V-trace backups for a stationary one-action policy (Efroni et al., 2018). These lookahead, trace, and return depths do not equal H_{upd}. Model-based SVG variants and Dreamer propagate reparameterized value gradients through learned dynamics or latent-world-model rollouts (Heess et al., 2015; Hafner et al., 2019); PIPPS and Zhang et al. analyze unstable or exploding-variance gradients through long such unrolls (Parmas et al., 2018; Zhang et al., 2023). HOPS unrolls policy actions through a sequence critic, not dynamics. RUDDER instead redistributes delayed returns over observed trajectories, so its environment-time credit span is also not H_{upd} (Arjona-Medina et al., 2019).

Other Transformer roles. GTrXL uses Transformer-XL as memory for partially observable control (Parisotto et al., 2020); Decision Transformer performs return-conditioned offline action prediction (Chen et al., 2021), and Trajectory Transformer models trajectories for beam-search planning (Janner et al., 2021). HOPS instead uses causal Transformers on both policy and value sides for online action-sequence optimization.

4 HOPS: Chunk-Autoregressive Policy Optimization

HOPS replaces T-SAC's one-step MLP actor with a chunk-autoregressive Transformer (Fig. 1a), retaining its twin prefix-conditioned Transformer critics (Fig. 2) and non-soft TD targets (Tian et al., 2026). We define the policy-update horizon H_{upd} as the number of consecutive policy-generated actions jointly evaluated during an actor update. Conventional SAC and T-SAC use $H_{\text{upd}} = 1$; HOPS optimizes longer generated sequences. The prediction horizon H_{pred} is the number of actions generated at each interaction decision, and the execution horizon H_{exec} is the number executed before replanning. The critic-update horizon H_Q is the maximum replay action-prefix length used in a critic update. We separately use the bootstrap-action horizon H_{target} for the number of new policy actions generated at a bootstrap state; it need not equal H_Q.

The actor emits C actions per chunk, where C is a task-dependent hyperparameter fixed within each experimental configuration. All lengths satisfy

$$C, H_{\text{pred}}, H_{\text{exec}}, H_{\text{upd}}, H_Q, H_{\text{target}} \in \mathbb{N}_{\geq 1}, \qquad 1 \leq H_{\text{exec}} \leq H_{\text{pred}}. \tag{6}$$

The chunk length specifies the number of actions emitted at each generation stage; the other horizons specify how generated or recorded actions are used. The policy can truncate its final chunk, so a generated sequence length need not be a multiple of C. Replay-prefix lengths likewise need not be multiples of C. The multiples of C used for actor updates in the default configuration below are a sampling choice, not a requirement of the policy factorization. For the default Meta-World configuration, we set $C = 4$ and $H_{\text{pred}} = H_{\text{exec}} = 4$, while updates sample

$$H_{\text{upd}} \sim \text{Unif}\{4, 8, 12, 16\}, \qquad H_Q \sim \text{Unif}\{1, \ldots, 16\}. \tag{7}$$

In the Meta-World policy-update-horizon ablations, $H_{\max}$ denotes the largest sampled H_{upd}. These ablations hold C, H_{pred}, H_{exec}, the rollout horizon, and critic training fixed.

4.1 Chunk-Autoregressive Transformer Policy

Let $\mathbf{a}_t^{(H)} = (a_t, \ldots, a_{t+H-1})$, $B(H) = \lceil H/C \rceil$ the number of chunks, and $r_b(H) = \min\{C, H - bC\}$ the chunk size of chunk b. Denoting chunk b by $\mathbf{c}_{t,b}^{(H)} = (a_{t+bC}, \ldots, a_{t+bC+r_b(H)-1})$, the policy factorizes as

$$\pi_\phi^{(H)}(\mathbf{a}_t^{(H)} \mid s_t) = \prod_{b=0}^{B(H)-1} \pi_\phi^{[r_b(H)]}\left(\mathbf{c}_{t,b}^{(H)} \mid s_t, \mathbf{a}_t^{(bC)}\right), \tag{8}$$

where $\mathbf{a}_t^{(0)} = \varnothing$ and $\pi_\phi^{[r]}$ is the marginal over the first r actions of a C-action output. If H is not divisible by C, only the required $r_{B(H)-1}(H)$ actions from the final chunk are retained.

The Transformer encodes the state as a context token and preceding actions with positional embeddings and causal attention. At each chunk boundary, the final valid representation parameterizes a diagonal Gaussian over $C \dim(\mathcal{A})$ controls. Tanh-squashed reparameterized samples are fed back for subsequent decoding. Unless stated otherwise, the actor has two layers and hidden dimension 32.

4.2 Critic-Guided Best-of-N Selection

Following Q-chunking (Li et al., 2026), interaction and evaluation select among $N \in \mathbb{N}_{\geq 1}$ policy proposals using the mean endpoint value of the twin critics:

$$\begin{aligned} &\mathbf{a}^{(n)} \overset{\text{i.i.d.}}{\sim} \pi_\phi^{(H)}(\cdot \mid s), \quad n = 1, \ldots, N, \\ &n^\star = \arg\max_n \frac{1}{2} \sum_{q=1}^{2} Q_{\theta_q}^{(H)}(s, \mathbf{a}^{(n)}), \qquad \mathbf{a}^\star = \mathbf{a}^{(n^\star)}. \end{aligned} \tag{9}$$

Here $H = H_{\text{pred}}$. The default Meta-World configuration uses $H_{\text{pred}} = 4$ and best-of-N selection with $N = 32$. Selection runs without gradient tracking and is used neither in actor optimization nor in TD bootstrapping. Increasing N changes the number of interaction proposals; it does not extend H_{upd}.

4.3 Variable-Horizon Actor and Critic Updates

Each actor update samples $H = H_{\text{upd}}$ and then reparameterizes $\mathbf{a}_t^{(H)} \sim \pi_\phi^{(H)}(\cdot \mid s_t)$ at replay states s_t. With critic parameters fixed, it minimizes

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{s_t, H, \boldsymbol{\epsilon}}\left[\frac{1}{H} \sum_{h=1}^{H} \min_{q \in \{1,2\}} Q_{\theta_q}^{(h)}(s_t, \mathbf{a}_t^{(h)})\right], \tag{10}$$

where $\boldsymbol{\epsilon}$ is the sampling noise. This averages the minimum critic value over all retained prefixes, with uniform positional weights, independently of the return discount $\gamma = 0.99$. We fix $\alpha = 0$ and disable automatic temperature tuning. Sampled actions remain differentiable as critic inputs, while their copies used to condition subsequent chunks are detached.

Each critic trains on recorded replay prefixes $\mathbf{a}_t^{(h)}$ for valid $h \leq H_Q$, averaging losses against prefix-specific non-soft TD targets. The bootstrap target value is computed as the arithmetic mean

of all prefix-value outputs from the twin target critics. Specifically, H_{target} is the bootstrap-action horizon, which sets the length of a newly generated action proposal rather than the replay-prefix length H_Q used to train the critic. At each bootstrap state s', we draw a single policy proposal $\mathbf{a}'^{(H_{\text{target}})} \sim \pi_\phi^{(H_{\text{target}})}(\cdot \mid s')$ and compute

$$\widehat{V}_{\text{target}}(s') = \frac{1}{2H_{\text{target}}} \sum_{q=1}^{2} \sum_{j=1}^{H_{\text{target}}} Q_{\bar{\theta}_q}^{(j)}\left(s', \mathbf{a}'^{(j)}\right), \tag{11}$$

where $\bar{\theta}_q$ denotes the parameters of target critic q, and $\mathbf{a}'^{(j)}$ is the first j actions of the sampled proposal. The default Meta-World configuration uses $H_{\text{target}} = 4$. Both target critics evaluate the same proposal, and the mean is taken uniformly over the critic and prefix dimensions. This bootstrap value is used in the prefix-specific TD targets without an entropy correction.

5 Experimental Evaluation

We evaluate HOPS on all 50 Meta-World ML1 tasks and dense-reward FANCY GYM Box Pushing, training a separate agent for each task online from scratch, without offline data or pretraining. We first compare complete learning systems and then use controlled ablations to examine the policy-update horizon and other design choices.

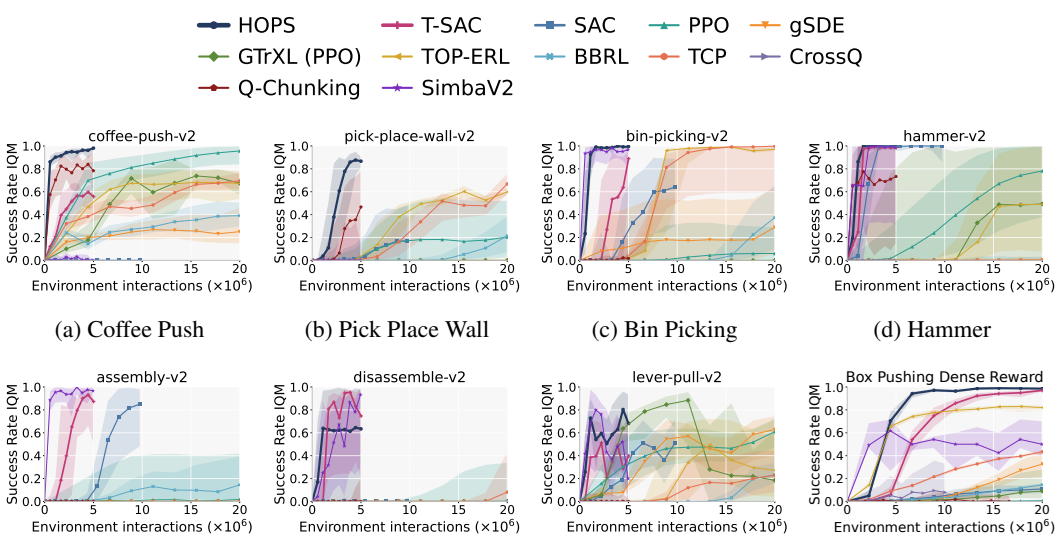
Table 1: **Meta-World ML1 performance near 5×10^6 interactions.** Unweighted mean of the 50 task-wise terminal-success IQMs. The table reports aggregate scores and the corresponding 95% intervals (Figure 1b).

Method	Mean task-wise IQM	95% interval
HOPS	**0.938**	[0.887, 0.976]
T-SAC	0.848	[0.788, 0.902]
SimbaV2	0.707	[0.590, 0.814]
SAC	0.672	[0.561, 0.776]
TOP-ERL	0.643	[0.529, 0.750]
PPO	0.613	[0.510, 0.709]
PPO+gSDE	0.585	[0.472, 0.693]
Q-Chunking	0.582	[0.479, 0.682]
TCP	0.571	[0.452, 0.690]
PPO+GTrXL	0.501	[0.392, 0.610]
CrossQ	0.500	[0.373, 0.631]
BBRL	0.479	[0.360, 0.598]

Protocol and metric. Episodes have fixed horizons of 500 primitive environment steps on Meta-World and 100 on Box Pushing. An evaluation episode is successful only if the task's success condition holds at the final timestep. For each task, we compute an interquartile mean (IQM) across seed-level terminal-success rates. The Meta-World score is the unweighted mean of its 50 task-wise IQMs, rather than an IQM over pooled task–seed results. Box Pushing uses its single-task IQM; terminal success requires position error at most $0.05\,\text{m}$ and orientation error at most $0.5\,\text{rad}$. Environment interactions count primitive transitions summed across parallel environments. On Meta-World, HOPS and T-SAC use five and eight training seeds per task, respectively. Additional protocol details appear in Appendix I.

Configuration and comparisons. HOPS uses a two-layer Transformer actor with hidden dimension 32 on Meta-World and four layers of width 128 on Box Pushing. The chunk length C is a task-dependent hyperparameter. For the default Meta-World configuration, $C = 4$, with $H_{\text{pred}} = H_{\text{exec}} = 4$ and $H_{\text{upd}} \sim \text{Unif}\{4, 8, 12, 16\}$. On Meta-World, interaction and evaluation use critic-guided best-of-N selection, with $N = 32$. T-SAC is the primary systems-level comparator: it uses prefix-conditioned Transformer critics and multi-horizon critic supervision, but retains a one-step MLP actor. The Meta-World comparison also includes PPO, PPO+gSDE, PPO+GTrXL, SAC, CrossQ, TCP, BBRL, TOP-ERL, Q-Chunking, and SimbaV2. The Box Pushing comparison includes the same baseline methods, including Q-Chunking and SimbaV2.

Reporting conventions and comparison scope. Task-level win, tie, loss, and failure counts require the underlying seed-level evaluation records. Comparisons are restricted to each method's available interaction range, and rankings refer to the point estimates at the stated budgets. Cross-method comparisons evaluate complete learning systems under their respective protocols and budgets; they neither isolate the effect of the policy-update horizon nor match computational budgets. The horizon ablations instead compare HOPS configurations with fixed interaction horizons and critic training. Computational costs are reported separately in Appendix G.



(a) Coffee Push (b) Pick Place Wall (c) Bin Picking (d) Hammer

(e) Assembly (f) Disassemble (g) Lever Pull (h) Box Pushing

Figure 3: **Representative Meta-World ML1 tasks and Box Pushing.** Panels (a)–(g) show seven Meta-World tasks spanning pushing, grasping and placement, tool use, assembly and disassembly, and articulated-object manipulation; panel (h) shows dense-reward FANCY GYM Box Pushing. Curves report task-wise terminal-success IQMs versus primitive environment interactions. Shading shows the reported 95% intervals for Meta-World and the supplied uncertainty bands for Box Pushing. The shared legend identifies the compared methods. Box Pushing scores are also reported in Table 2. The selected tasks illustrate improvements, competitive performance, and remaining limitations; the Meta-World benchmark-level comparison uses all 50 tasks. Methods retain their respective plotted interaction budgets.

5.1 Main Results

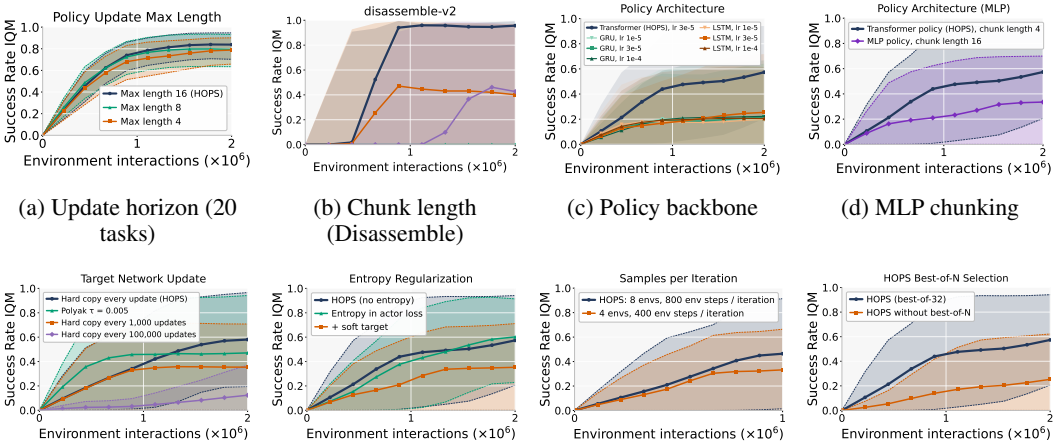
Meta-World aggregate performance. At 5×10^6 environment interactions, HOPS achieves a mean task-wise terminal-success IQM of 0.938, compared with 0.848 for T-SAC, a difference of 0.090 (Table 1). HOPS has the highest aggregate score among the evaluated configurations at the comparison budgets. The Q-Chunking reference reaches 0.582 at 5×10^6 interactions, while SimbaV2 reaches 0.707 at its 4.95×10^6-interaction endpoint.

Meta-World task-level variation. The aggregate improvement is accompanied by substantial variation across tasks (Figure 3). Around 5×10^6 interactions, HOPS and T-SAC score 0.98 and 0.56 on `coffee-push-v2`, and 0.87 and 0.00 on `pick-place-wall-v2`, respectively. HOPS also reaches 0.99 on `bin-picking-v2`. However, its task-wise IQM remains zero on `assembly-v2`, where T-SAC and SimbaV2 reach 0.87 and 0.97 at their respective final checkpoints. On `disassemble-v2`, HOPS finishes at 0.63, compared with 0.75 for T-SAC and 0.93 for SimbaV2. HOPS reaches 0.68 on `lever-pull-v2` at 5×10^6 interactions, while `hammer-v2` illustrates a task on which several methods rapidly reach high success. Full results for all 50 tasks appear in Figures 5 and 6.

Box Pushing. On dense-reward FANCY GYM Box Pushing, a 7-DoF arm pushes a box to a target position and orientation. At 8×10^6 interactions, HOPS reaches a terminal-success IQM of 0.941, compared with 0.637 for T-SAC and 0.757 for TOP-ERL (Figure 3(h); Table 2). The HOPS–T-SAC difference at this budget is 0.304. At 20×10^6 interactions, HOPS reaches 0.986; T-SAC finishes at 0.966 after 19.986×10^6 interactions, and TOP-ERL finishes at 0.822 after 20×10^6. The HOPS–T-SAC gap is therefore larger at the earlier comparison budget than at the endpoints. CrossQ is evaluated only within its observed range, which ends at about 10.05×10^6 interactions.

5.2 Ablation Studies

We examine the policy-update horizon, chunk length, policy architecture, and training configuration (Figure 4). Each configuration uses four seeds per task. The update-horizon study covers 20 Meta-



(a) Update horizon (20 tasks) (b) Chunk length (Disassemble) (c) Policy backbone (d) MLP chunking

(e) Target-critic updates (f) Entropy (g) Samples per iteration (h) Best-of-N selection

Figure 4: **HOPS ablations.** Panel (a) shows the 20-task aggregate; panel (b) shows the task-wise IQM on Disassemble, with all six chunk-length tasks reported in Appendix B.2. In (b), blue circles, green triangles, orange squares, and purple diamonds denote $C = 4, 3, 2, 1$, respectively. Panels (c)–(h) show the four-task aggregates. Shading indicates the reported 95% intervals, with four seeds per task. Panel (g) ends at 10^6 interactions; the others end at 2×10^6.

World tasks, and the chunk-length study reports six task-wise comparisons. The remaining studies use `assembly-v2`, `coffee-push-v2`, `disassemble-v2`, and `pick-place-wall-v2`. The interaction budget is 2×10^6, except for the samples-per-iteration study, which uses 10^6. Comparisons are made within each study because task sets, seeds, and budgets differ across studies. Full task-level curves appear in Appendix B.

Policy-update horizon. We vary the maximum actor-update horizon $H_{\max} \in \{4, 8, 16\}$ while keeping $C = 4$, $H_{\text{pred}} = H_{\text{exec}} = 4$, and critic training fixed. Here $H_{\max}$ bounds the sampled H_{upd}; it does not set the critic-training horizon H_Q. The candidate sampling specification and the remaining configuration checks are given in Section 4. At 2×10^6 interactions, the aggregate scores are 0.788, 0.795, and 0.838 for maximum update horizons of four, eight, and sixteen, respectively. Thus, extending the maximum update horizon from four to sixteen improves the aggregate score by 0.050 on the evaluated 20-task subset. The effects are task dependent and non-monotonic: longer updates help on `shelf-place-v2` and `pick-place-wall-v2`, whereas `basketball-v2` favors a maximum horizon of eight.

Table 2: **Box Pushing performance.** Terminal-success IQMs at 8×10^6 interactions and at each method's final evaluation budget.

Method	IQM at 8×10^6	Endpoint IQM
HOPS	**0.941**	**0.986**
T-SAC	0.637	0.966
TOP-ERL	0.757	0.822
TCP	0.186	0.426
PPO+gSDE	0.015	0.315
BBRL	0.023	0.140
SAC	0.032	0.107
PPO+GTrXL	0.012	0.088
CrossQ	0.079	0.082
PPO	< 0.001	0.005
SimbaV2	0.525	0.500
Q-Chunking	< 0.001	0.005

Policy chunk length. The six task-wise comparisons of $C \in \{1, 2, 3, 4\}$ show substantial sensitivity to chunk length. On `disassemble-v2`, $C = 4$ reaches 0.96, compared with roughly 0.4 for $C = 1, 2$ and zero IQM for $C = 3$. Shorter chunks finish higher on `basketball-v2` and `pick-place-v2`, while $C = 2$ gives the highest final score on `pick-place-wall-v2`. All configurations finish above 0.9 on `coffee-push-v2`, and all have zero final IQM on `assembly-v2`. These results support treating C as a task-dependent hyperparameter that should be selected for the target task. We use $C = 4$ as the default for Meta-World; the comparisons do not identify a universally preferred chunk length.

Policy architecture. The Transformer at learning rate 3×10^{-5} reaches an aggregate score of 0.57, compared with 0.20–0.25 for the GRU and LSTM configurations tested at 10^{-5}, 3×10^{-5}, and 10^{-4}.

The largest task-level differences occur on `disassemble-v2` and `pick-place-wall-v2`. The Transformer with $C = 4$ also finishes above the evaluated MLP with $C = 16$ (0.57 versus 0.34). Because this comparison changes both architecture and chunk length, it measures the combined configuration effect.

Target-critic updates. Hard copying after every critic update gives a final aggregate score of 0.58, followed by Polyak updates with $\tau = 0.005$ at 0.47, hard copies every 10^3 critic updates at 0.35, and hard copies every 10^5 updates at 0.12. Polyak updates improve faster initially, while immediate hard copies finish higher within the evaluated budget.

Entropy regularization. The entropy-free default ($\alpha = 0$) learns faster initially than the two entropy-regularized variants. Automatically tuned entropy in the policy loss alone gives a slightly higher final score (0.60 versus 0.57), while adding entropy to both the policy loss and critic target reduces the final score to 0.35. The entropy-free default therefore favors early learning performance, with policy-only entropy remaining competitive at the endpoint.

Samples per iteration. Eight parallel environments collecting 800 steps per iteration reach an aggregate score of 0.46 at 10^6 interactions, compared with 0.33 for four environments collecting 400 steps. This comparison changes both parallelism and collection volume, so it does not isolate an update-to-data-ratio effect.

Critic-guided selection. Best-of-N selection with $N = 32$ reaches an aggregate score of 0.57 at 2×10^6 interactions, compared with 0.25 when a single chunk is sampled without critic-based ranking. The largest task-level difference occurs on `disassemble-v2` (0.96 versus zero IQM). Selection changes during both data collection and evaluation, so the comparison measures its combined effect on the interaction procedure.

Together, these studies support the evaluated HOPS configuration. The horizon study specifically shows that longer actor updates can improve aggregate performance while the interaction horizons and critic training remain fixed. The remaining studies characterize complementary design choices; their effects cannot be added across different task sets and budgets.

6 Conclusion and Future Work

We introduce HOPS, an off-policy actor–critic method that independently controls the policy-update horizon through autoregressive composition of action chunks of length C. The chunk length is a task-dependent hyperparameter, with $C = 4$ used as the default for Meta-World. This design allows actor optimization over longer policy-generated sequences while retaining short prediction and execution horizons that are fixed for a given configuration. Comparisons with the evaluated step-based, temporally extended, and action-chunking baselines show strong terminal-success performance on Meta-World and dense-reward Box Pushing. On the 50-task Meta-World benchmark, HOPS achieves the highest reported mean task-wise terminal-success IQM among the evaluated configurations at the stated comparison budgets, including the closely related T-SAC baseline with prefix-conditioned Transformer critics. This aggregate advantage does not imply uniform task-level dominance: failures and task-dependent limitations remain. Controlled ablations on a 20-task Meta-World subset further show that extending the maximum update horizon can improve aggregate performance without changing the interaction horizons or critic training, although the gains are task dependent. These findings support the policy-update horizon as a practical design choice for action-chunked reinforcement learning. Future work will examine comparisons under matched computational budgets and extend the evaluation to visual observations and physical robots.

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 HOPS architecture, action-generation procedure, and actor–critic updates. Section 5 and Appendix I 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 HOPS 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

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

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.

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

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.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. Beyond the one-step greedy approach in reinforcement learning. In *International Conference on Machine Learning*, pp. 1387–1396. 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.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

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/.

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.

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.

Fernando Puente León and Uwe Kiencke. *Ereignisdiskrete Systeme*. Oldenbourg Wissenschaftsv, 2013.

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

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.

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.

Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. Pipps: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pp. 4065–4074. PMLR, 2018.

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.

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.

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.

R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998. doi: 10.1109/TNN.1998.712192.

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.

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.

Shenao Zhang, Boyi Liu, Zhaoran Wang, and Tuo Zhao. Model-based reparameterization policy gradient methods: Theory and practical algorithms. *Advances in Neural Information Processing Systems*, 36:68391–68419, 2023.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A FULL META-WORLD ML1 RESULTS

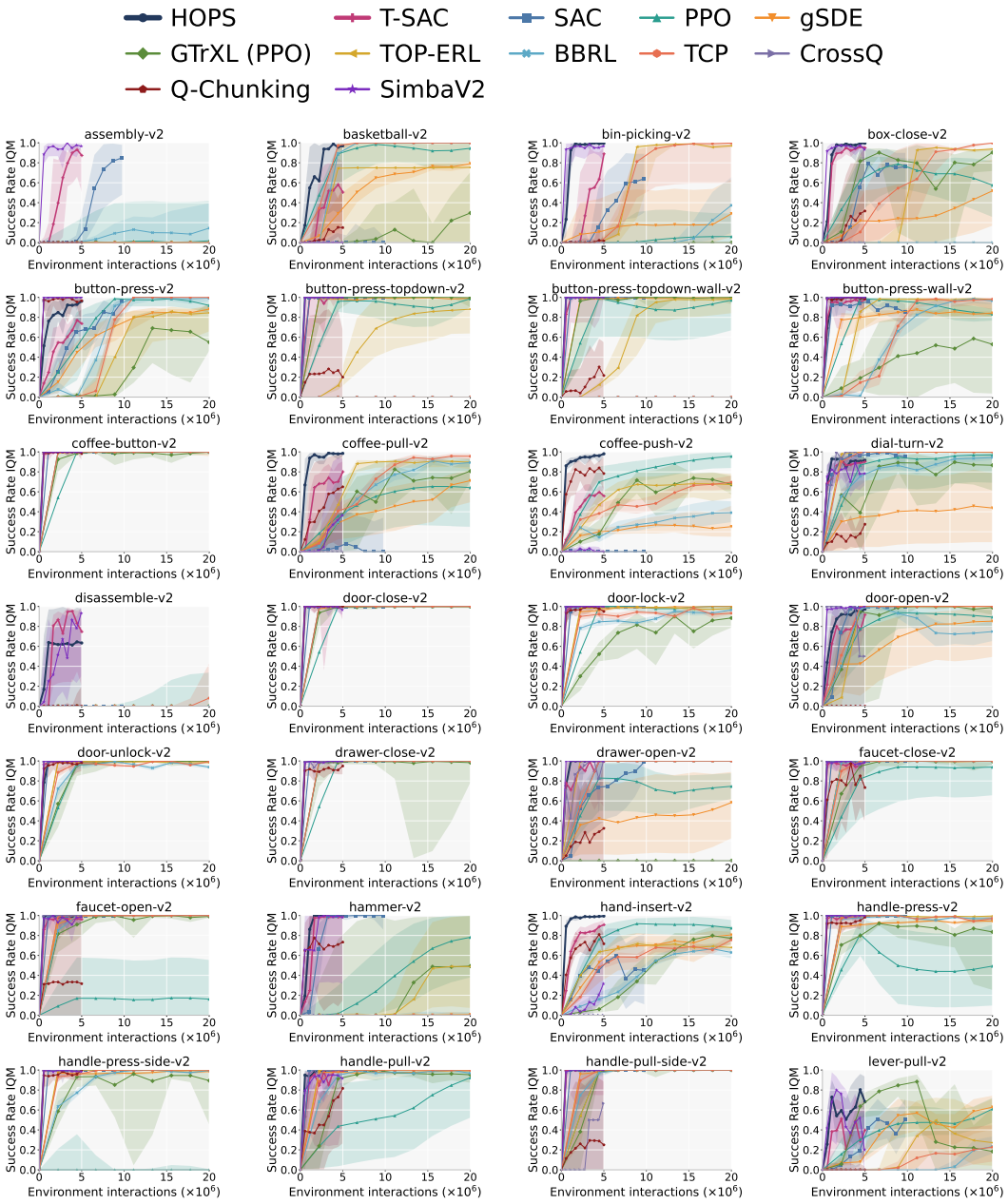


Figure 5: Task-wise terminal-success IQMs on Meta-World ML1 (tasks 1–28; Part 1 of 2).

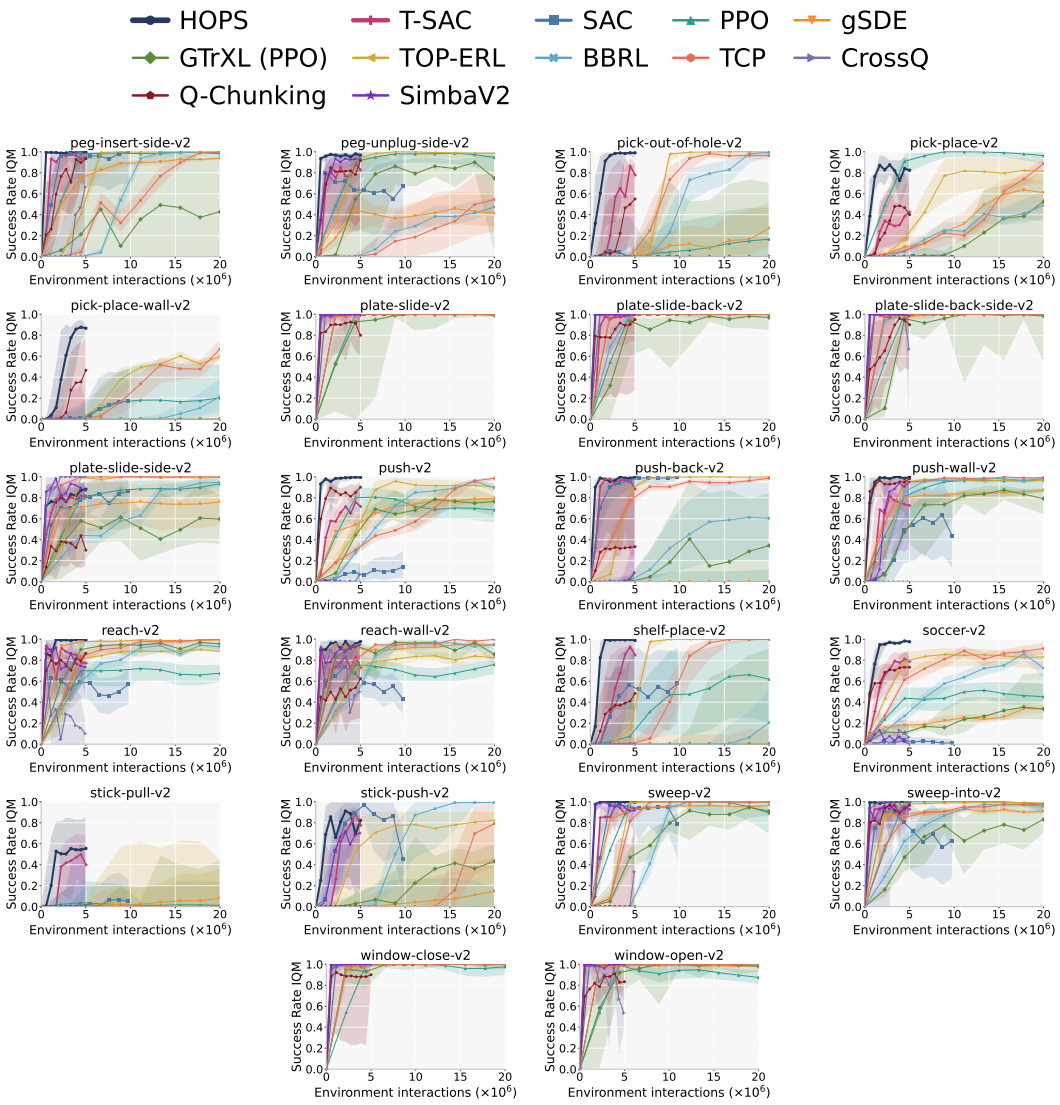


Figure 6: Task-wise terminal-success IQMs on Meta-World ML1 (tasks 29–50; Part 2 of 2).

B Full Ablation Results

This appendix provides the task-level curves for Section 5.2. Each configuration uses four random seeds per task. The central curves report success-rate IQM, and shading denotes 95% confidence intervals. The numerical comparisons below report task-wise IQMs and, where applicable, aggregate scores. The reported differences are descriptive; no statistical-significance claim is inferred from the ordering of the central estimates. The horizon study covers 20 tasks. The chunk-length study reports six task-wise comparisons, without an aggregate ranking. The other studies use the four-task subset specified in Section 5.2. The interaction budget is 2×10^6, except for the samples-per-iteration study (10^6).

B.1 Policy-Update Horizon

We compare maximum policy-update horizons $H_{\max} \in \{4, 8, 16\}$, with the interaction horizons and critic training held fixed (Section 4). The final aggregate central estimates are approximately 0.788, 0.795, and 0.838, respectively. Figure 7 shows all 20 tasks. The benefit of extending the update horizon is task dependent: on `shelf-place-v2`, 16 learns fastest and 16 and 8 finish near 1.0, compared with 0.59 for 4; on `pick-place-wall-v2`, 16 reaches approximately 0.45, compared with 0.11 and 0.17 for 8 and 4. Conversely, `basketball-v2` favors 8 (0.93 versus 0.49 for 16), while `disassemble-v2` favors 16 and 4 (both about 0.96, versus 0.42 for 8). On `stick-push-v2`, the 16-step curve improves earlier but declines late in training, finishing below 4. All three central IQM curves remain at zero on `assembly-v2`. These outcomes preclude a monotonic, task-independent interpretation of the aggregate improvement.

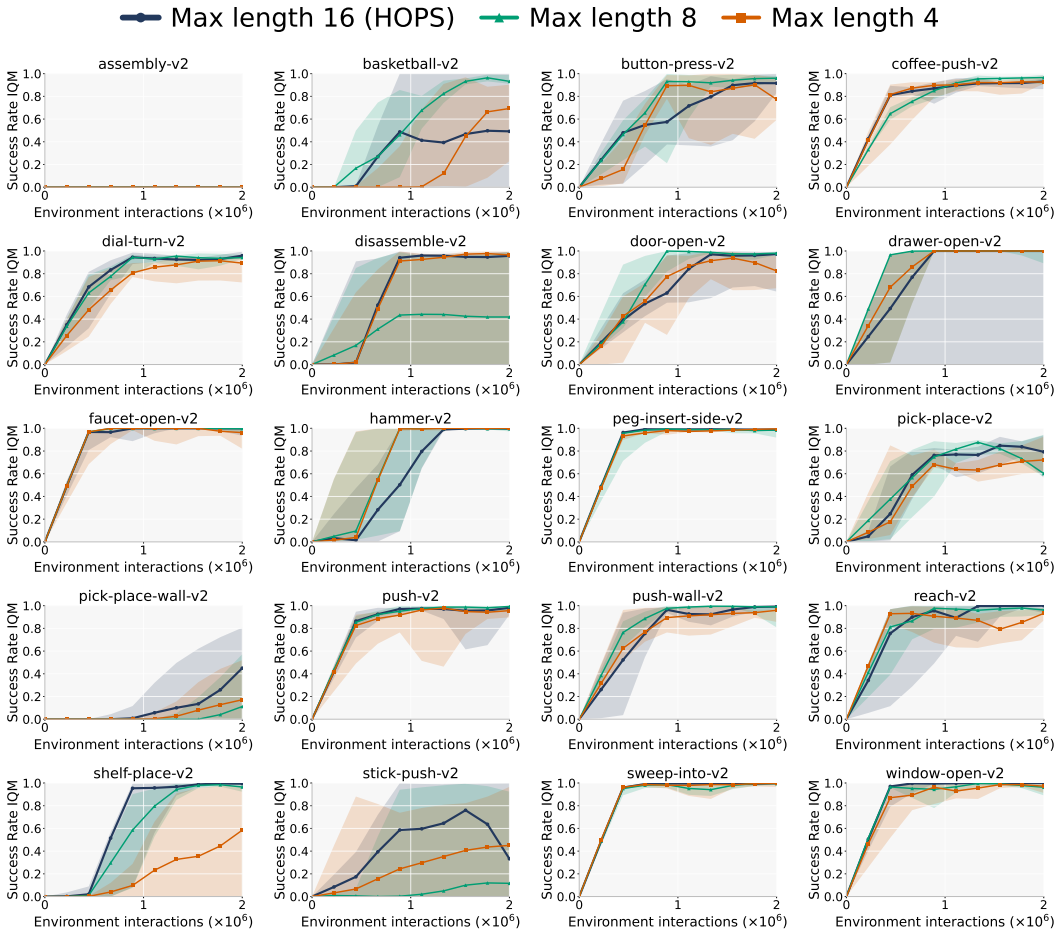


Figure 7: Policy-update horizons. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

B.2 Policy Chunk Length

Figure 8 compares $C \in \{1, 2, 3, 4\}$ on six tasks. We discuss these task-wise curves without assigning an aggregate ranking to the chunk lengths. The task-level results show pronounced sensitivity to chunk length. On `disassemble-v2`, $C = 4$ reaches approximately 0.96, whereas $C = 3$ remains at zero and $C = 1, 2$ finish near 0.4. Shorter chunks finish above $C = 4$ on `basketball-v2` and `pick-place-v2`; $C = 2$ also has the highest final central estimate on `pick-place-wall-v2`. All configurations finish above 0.9 on `coffee-push-v2`, while their central IQM curves remain at zero on `assembly-v2`. In particular, $C = 3$ performs well on basketball, coffee-push, and pick-place but retains zero central IQM on disassemble and pick-place-wall. These results indicate that the chunk length C should be selected for the target task. The value $C = 4$ is the default for Meta-World, rather than a universally optimal chunk length.

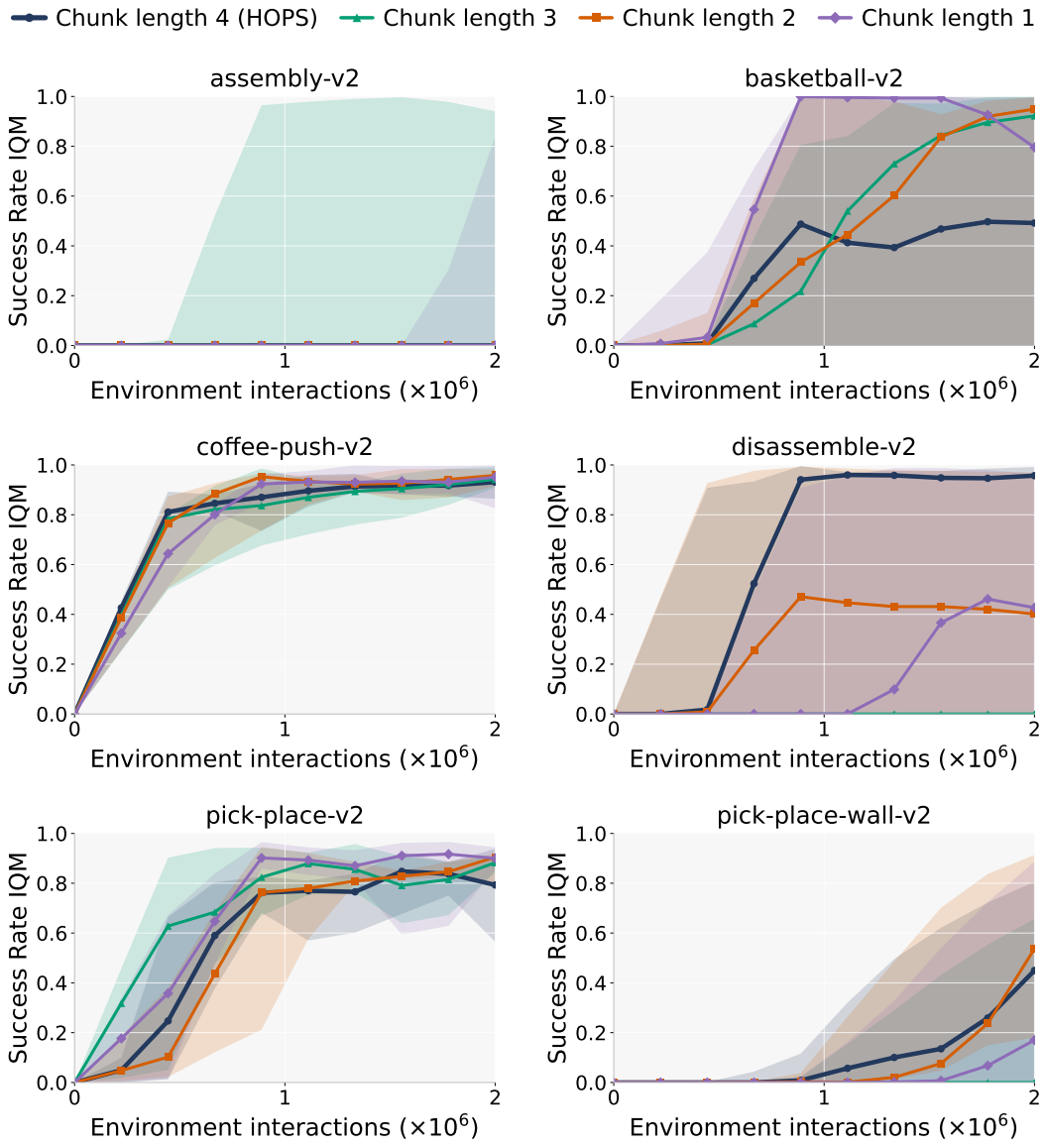


Figure 8: Policy chunk lengths $C \in \{1, 2, 3, 4\}$. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

B.3 Transformer and Recurrent Policies

Figure 9 compares the Transformer at learning rate 3×10^{-5} with GRU and LSTM policies, each tested at 10^{-5}, 3×10^{-5}, and 10^{-4}. The Transformer reaches approximately 0.57 final aggregate IQM. The best final aggregate central estimate among the tested GRUs is approximately 0.22 at 10^{-4}; among the tested LSTMs, it is approximately 0.25 at 3×10^{-5}. At the shared learning rate of 3×10^{-5}, the Transformer also finishes higher in the aggregate.

The largest task-level differences occur on `disassemble-v2`, where the Transformer reaches about 0.96 while the recurrent policies remain at zero central IQM, and on `pick-place-wall-v2`, where it reaches about 0.45 compared with at most 0.06 for the recurrent alternatives. On `coffee-push-v2`, the Transformer learns faster early in training, but LSTM at 3×10^{-5} finishes slightly higher (about 0.96 versus 0.93). All central curves remain at zero on `assembly-v2`. The comparison favors the evaluated Transformer configuration; it does not establish equivalence in parameter counts or computation.

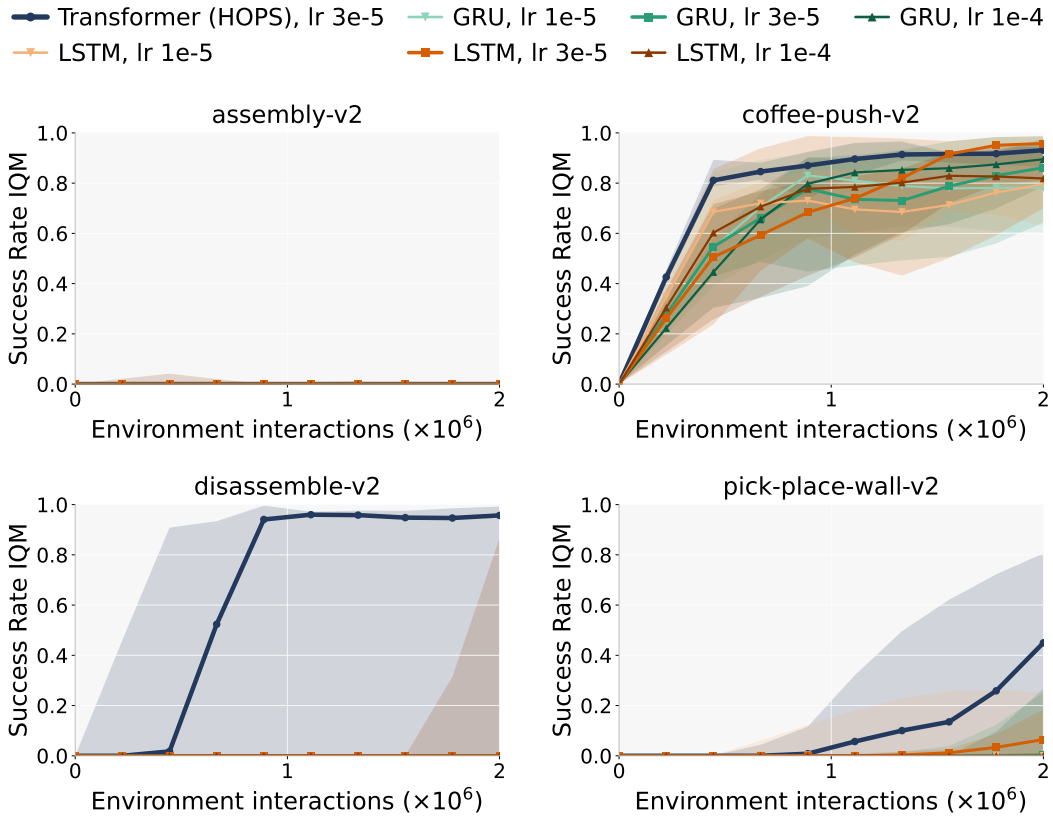


Figure 9: Transformer at 3×10^{-5} versus GRU and LSTM learning-rate sweeps. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

B.4 MLP CHUNKING

Figure 10 compares the Transformer with chunk length $C = 4$ against an MLP policy with $C = 16$. Their final aggregate IQMs are approximately 0.57 and 0.34. Both configurations finish near 0.93 on `coffee-push-v2`, although the Transformer improves faster early in training. The larger differences occur on `disassemble-v2` (0.96 versus 0.41) and `pick-place-wall-v2` (0.45 versus zero). Both central IQM curves remain at zero on `assembly-v2`. Because the policy architecture and chunk length change together, these results compare the two complete policy configurations and cannot attribute the difference solely to the use of a Transformer.

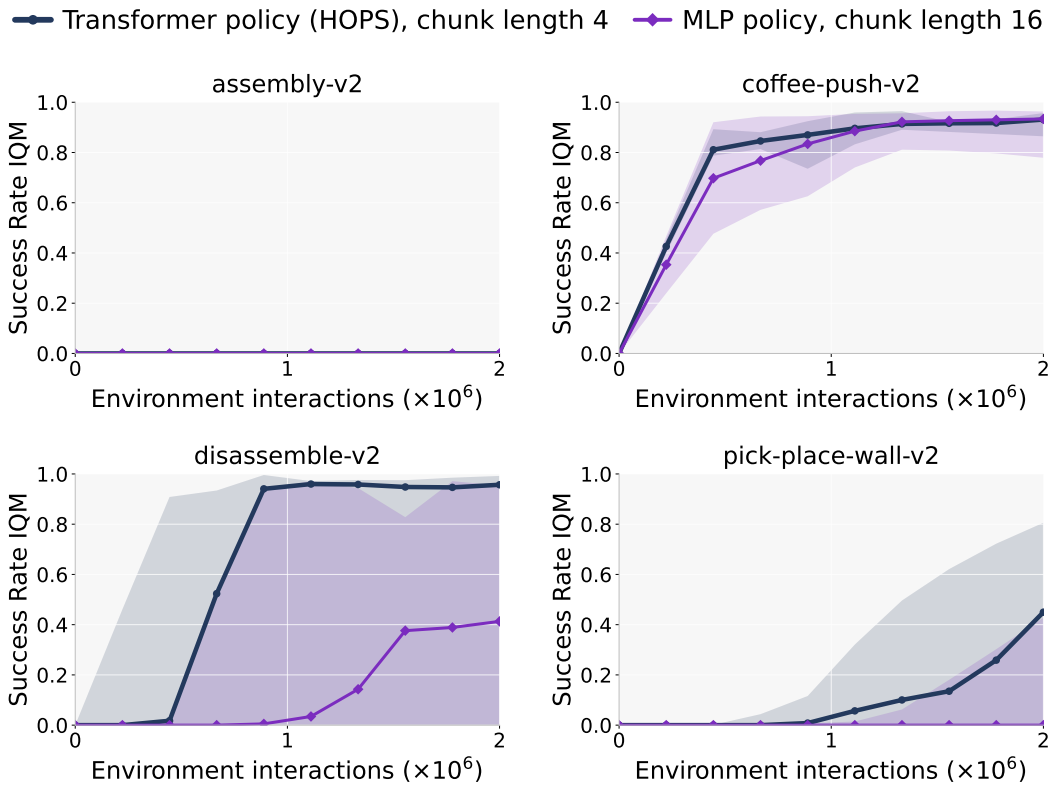


Figure 10: Transformer with $C = 4$ versus MLP with $C = 16$. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

B.5 TARGET-CRITIC UPDATE RULE

Figure 11 compares hard copying after every critic update ($\tau = 1$), Polyak updates with $\tau = 0.005$ after every critic update, and hard copies every 10^3 or 10^5 critic updates. Their final aggregate central estimates are approximately 0.58, 0.47, 0.35, and 0.12, respectively. This experiment varies target-network synchronization; it does not vary the bootstrap-action horizon H_{target}.

Polyak updates learn faster initially, especially on `disassemble-v2`, but immediate hard copies catch up and finish slightly higher there (about 0.99 versus 0.96). Immediate hard copies are also the only configuration to attain substantial nonzero central IQM on `pick-place-wall-v2` (about 0.39). Hard copies every 10^3 updates reach about 0.48 on disassemble, while the 10^5-update interval remains at zero there and reaches only about 0.51 on `coffee-push-v2`. The other three configurations finish near 0.93–0.95 on coffee-push. All central curves remain at zero on `assembly-v2`.

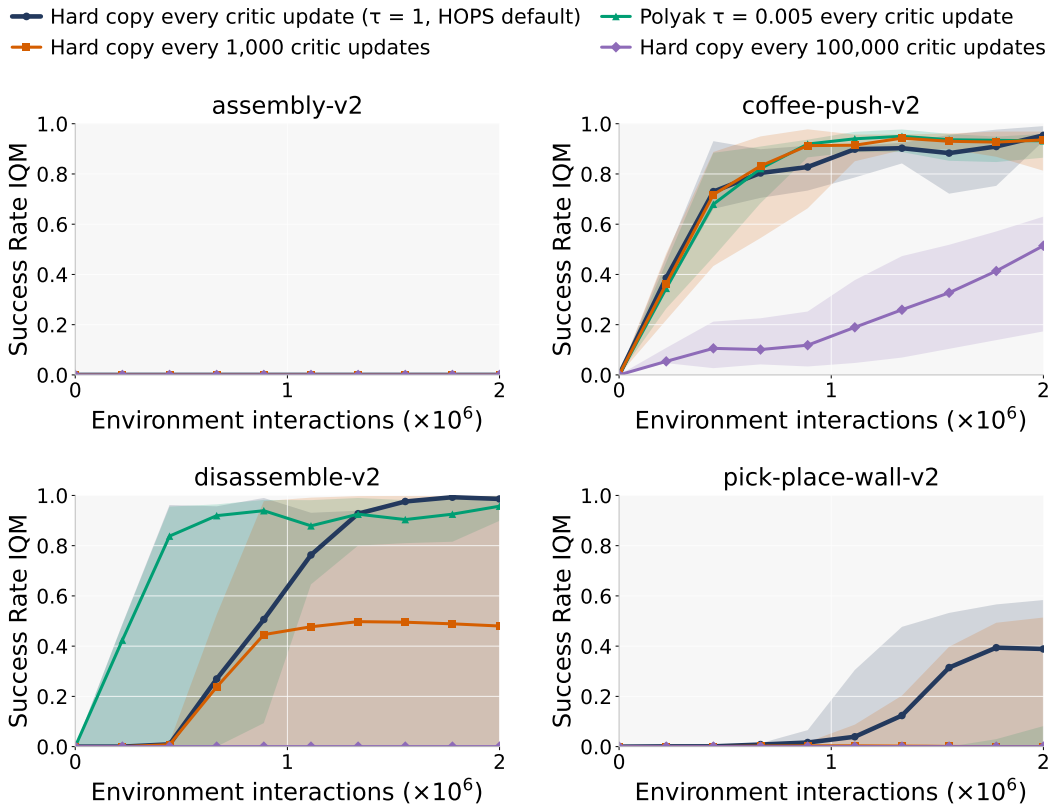


Figure 11: Target-critic synchronization: immediate hard copies, Polyak updates, and delayed hard copies. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

B.6 Entropy Regularization

Figure 12 compares the entropy-free default ($\alpha = 0$), automatically tuned entropy in the policy loss, and automatically tuned entropy in both the policy loss and a soft critic target. The final aggregate IQMs are approximately 0.57, 0.60, and 0.35, respectively. The entropy-free default learns faster early in training on `coffee-push-v2` and `disassemble-v2`. Policy-only entropy improves the final central estimate on `pick-place-wall-v2` (about 0.60 versus 0.45), while finishing slightly lower on disassemble (about 0.91 versus 0.96) and similarly on coffee-push. Adding the soft target yields about 0.46 on disassemble and zero on pick-place-wall, despite reaching about 0.95 on coffee-push. All three central IQM curves remain at zero on `assembly-v2`. These results distinguish the effect of policy-loss entropy from the additional change to the critic target; they do not support claiming that entropy is uniformly beneficial or unnecessary.

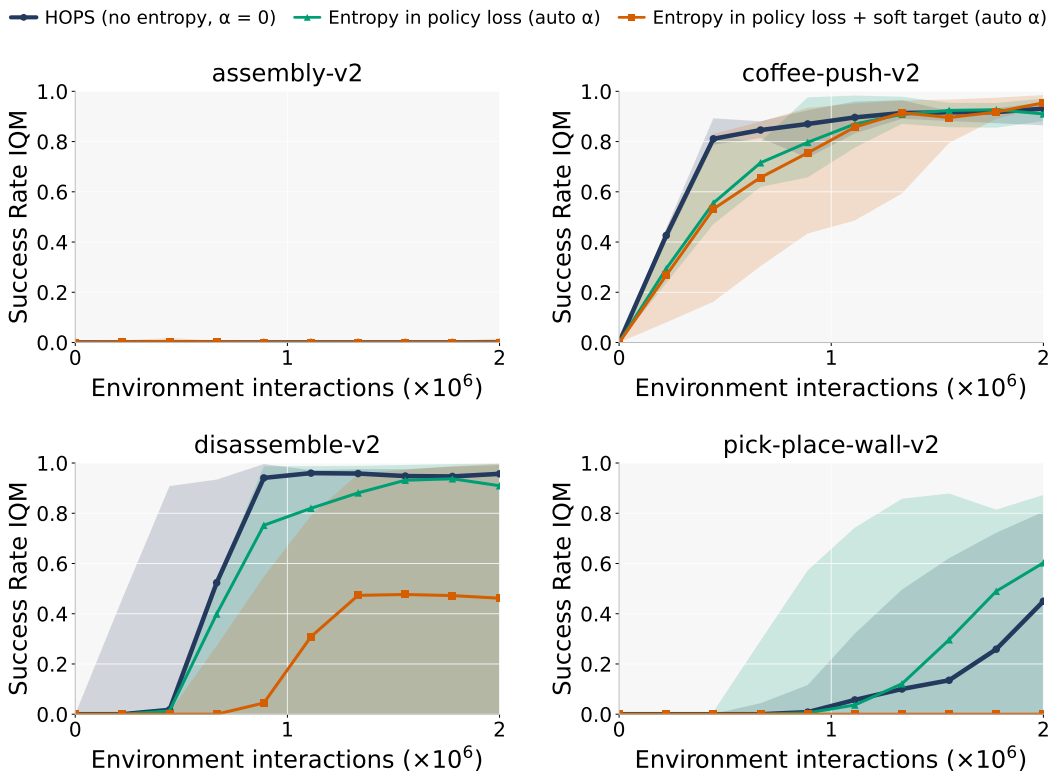


Figure 12: No entropy, policy-loss entropy, and policy-loss entropy with a soft critic target. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

B.7 Samples per Iteration

Figure 13 compares eight parallel environments collecting 800 environment steps per iteration with four environments collecting 400 steps per iteration. The plotted budget is 10^6 interactions. At that point, the aggregate central estimates are approximately 0.46 and 0.33, respectively. The largest difference is on `disassemble-v2`, where the 800-step configuration reaches about 0.95, compared with 0.48 for the 400-step configuration. It also learns faster initially on `coffee-push-v2`, although the final central estimates are close (0.88 versus 0.86). Both retain zero central IQM on `assembly-v2`, and `pick-place-wall-v2` shows only limited progress within this budget (about 0.03 versus zero). The comparison jointly changes the number of parallel environments and the collection volume per iteration, without independently identifying the effect of either factor or the update-to-data ratio.

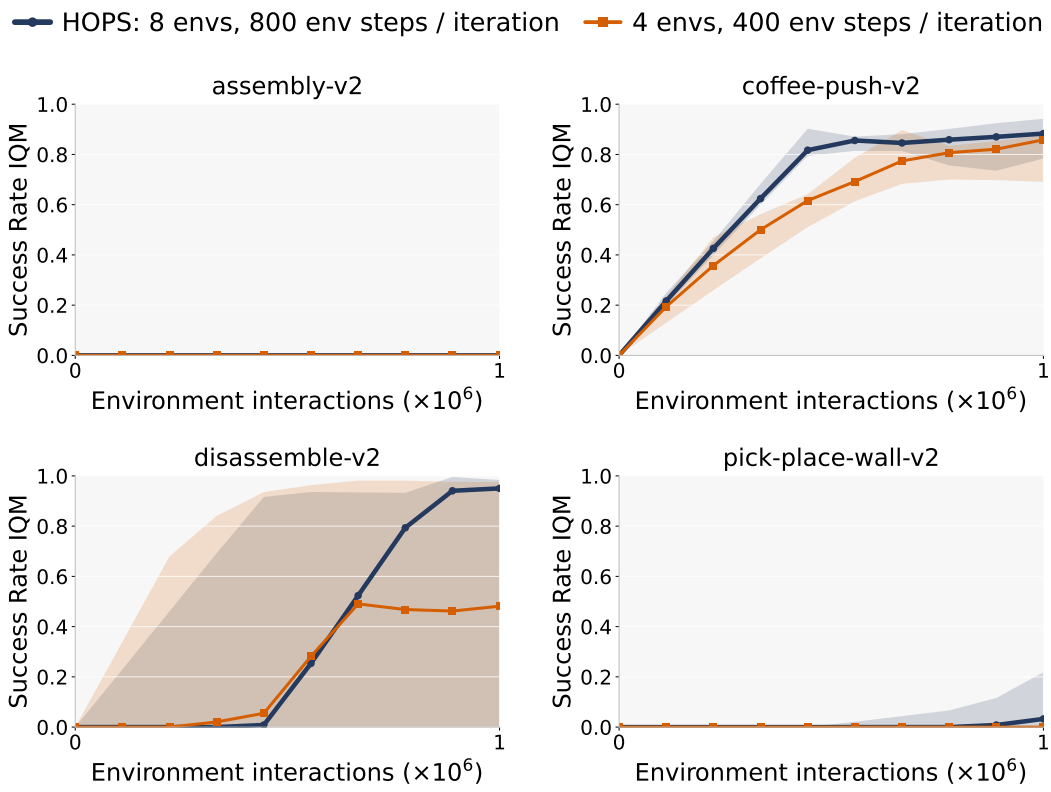


Figure 13: Eight environments with 800 steps per iteration versus four environments with 400 steps per iteration. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

B.8 BEST-OF-N SELECTION

Figure 14 compares HOPS with best-of-32 chunk selection against a variant that directly samples a single chunk without critic-based ranking during interaction and evaluation. The chunk length remains $C = 4$. As described in Section 4, candidate selection is used for interaction and evaluation, not in the actor loss or TD bootstrapping. At 2×10^6 environment interactions, the final aggregate central estimates are approximately 0.57 and 0.25, respectively, a difference of about 0.32.

The largest task-level separation occurs on `disassemble-v2`: best-of-32 reaches approximately 0.96, whereas the single-sample variant retains zero central IQM throughout the plotted budget. This does not imply failure in every seed, as its uncertainty band widens late in training. On `coffee-push-v2`, selection produces substantially faster early learning and a higher final central estimate (approximately 0.93 versus 0.83). On `pick-place-wall-v2`, it reaches approximately 0.45, compared with 0.21 without selection, while both configurations retain zero central IQM on `assembly-v2`.

These results support critic-guided selection within the evaluated HOPS configuration, with the benefit concentrated in a subset of tasks. Because selection is removed during both data collection and evaluation, the comparison does not isolate its contribution to either stage. The overlapping confidence intervals alone do not establish the statistical significance of the observed differences.

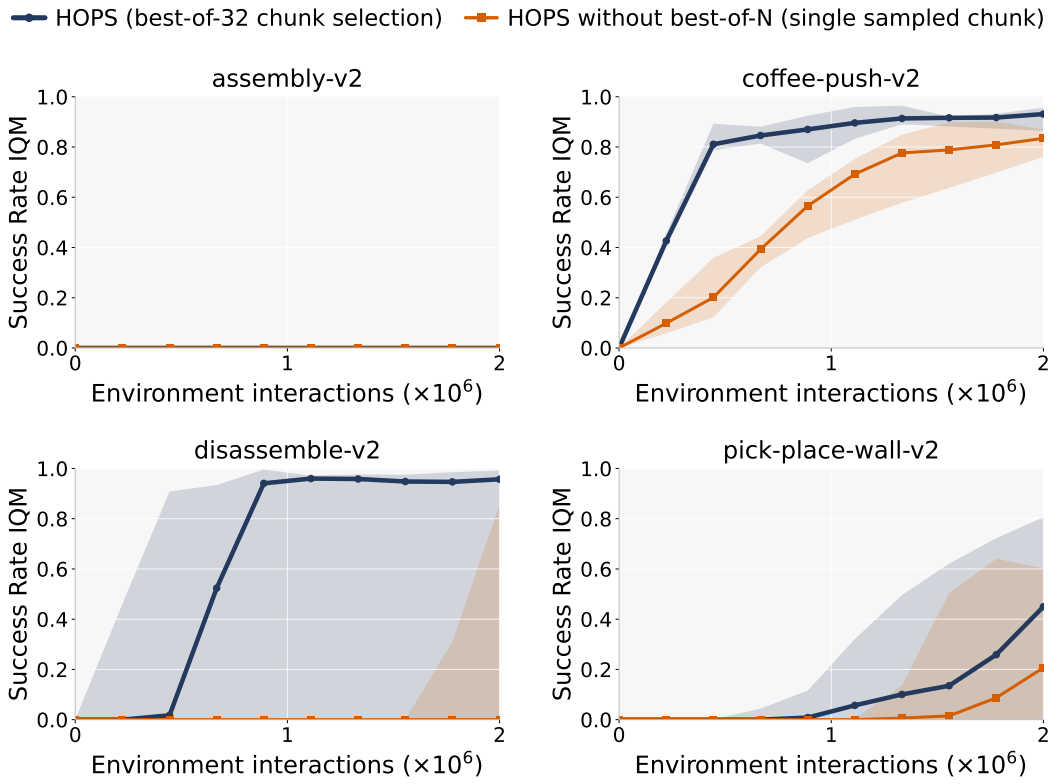
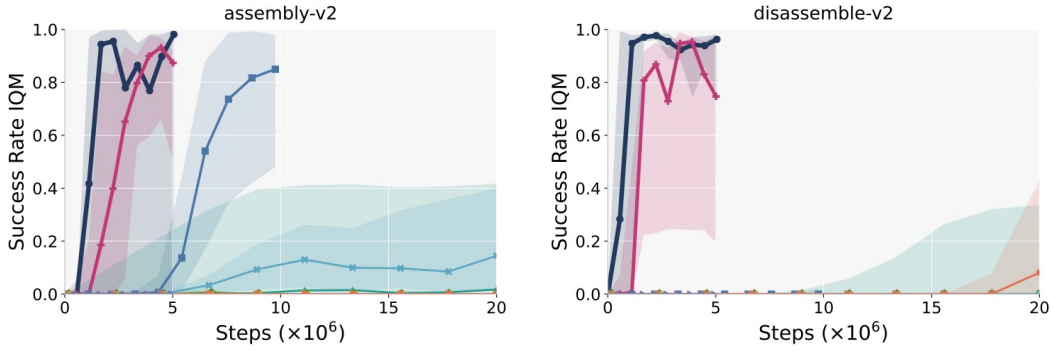
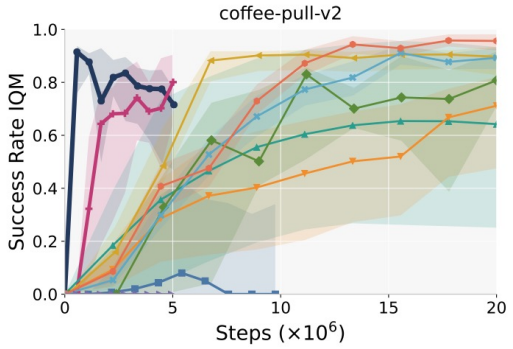


Figure 14: Best-of-32 chunk selection versus a single sampled chunk without critic-based ranking. Success-rate IQM versus environment interactions; shading denotes 95% confidence intervals from four seeds per task.

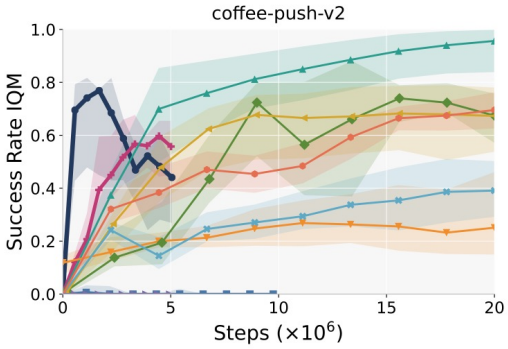


(a) Assembly

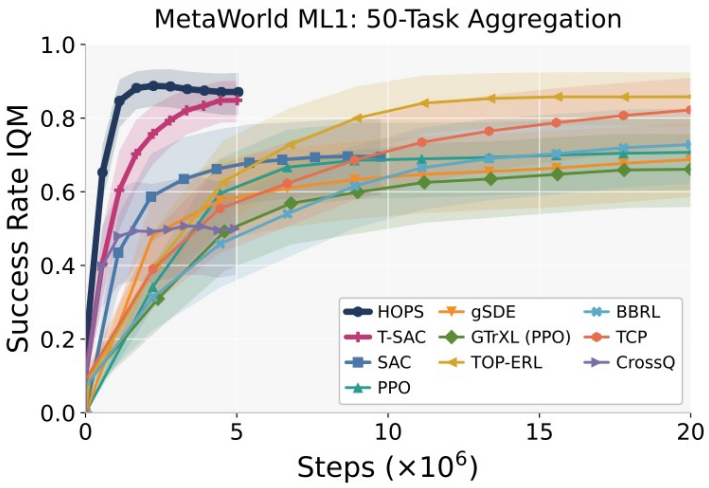
(b) Disassemble



(c) Coffee Pull



(d) Coffee Push



(e) 50-task aggregate

Figure 15: **Task recovery and performance trade-offs under an alternative HOPS configuration.** Dark-blue HOPS curves use eight samples per collection step and a replay capacity of 5,000 entries, versus four samples and 100,000 entries in the standard configuration. Task-wise IQMs use four independent random seeds per task. Panels (a)–(b) show high success-rate IQMs on Assembly and Disassemble; (c)–(d) show late-training degradation on Coffee Pull and Coffee Push. Panel (e) shows the unweighted mean of task-wise IQMs across all 50 tasks and the shared legend. Other methods serve as references; standard HOPS curves are reported in the main results and omitted here.

C Failure Analysis and Sensitivity to Experience Replay

Under the standard configuration, HOPS fails to learn successful behavior on `assembly-v2`, while `disassemble-v2` shows seed-dependent failure: some runs succeed, but two seeds do not converge within the evaluated budget. This seed-dependent failure is evident in individual training runs rather than the aggregate IQM alone, highlighting the need to assess reliability across seeds alongside central performance estimates. An exploratory configuration improves both tasks but reduces aggregate performance and causes late-training degradation on other tasks, motivating further study of experience collection and replay.

An alternative collection and replay configuration. We jointly increase the number of samples per collection step from four to eight and reduce replay-buffer capacity from $100{,}000$ to $5{,}000$ entries. The standard buffer retains the full collected history in these runs; the smaller buffer limits it. These joint changes are exploratory, not a controlled ablation of either parameter. All exploratory HOPS experiments use four independent random seeds per task across all 50 tasks, with each task-wise IQM computed across the four seed-level terminal-success rates.

At 5×10^6 interactions, terminal-success IQM reaches approximately 0.98 on `assembly-v2` and 0.96 on `disassemble-v2` (Figure 15(a)–(b)). Assembly improves substantially but fluctuates during training, whereas Disassemble maintains a high central estimate after its initial improvement. These results show that the HOPS architecture can learn successful behavior on both tasks, but do not establish convergence of every seed.

Trade-offs and selection of the standard configuration. The gains on these tasks come with losses elsewhere. On `coffee-pull-v2`, the central IQM falls from an early peak of approximately 0.91 to 0.72; on `coffee-push-v2`, it decreases from approximately 0.77 to 0.44 (Figure 15(c)–(d)). The 50-task aggregate finishes near 0.87 at 5×10^6 interactions, below approximately 0.938 for the standard configuration (Table 1). We therefore retain the standard configuration for the main benchmark evaluation and report these exploratory results separately, without substituting individual tasks in the main aggregate.

Working hypothesis. Assembly and Disassemble may be particularly sensitive to the collection and replay of informative experience. A history dominated by early behavior that provides limited support for successful actions may dilute the influence of newly acquired useful experience. A smaller buffer could reduce the persistence of such data, allowing recent informative samples to influence learning more quickly; increased collection volume may also shift this balance.

However, retaining less history may reduce replay diversity and discard experience supporting previously successful behavior. Learning from a narrower set of recent experiences could encourage locally effective but suboptimal behavior or contribute to late-training degradation. These mechanisms remain hypotheses rather than conclusions established by the curves. Because collection volume and buffer capacity change jointly, the observed effects cannot be attributed to replay capacity alone.

Directions for further study. Controlled experiments should vary collection volume and replay capacity independently, track the age and success composition of replayed data, and relate these measurements to individual seed trajectories. Such analyses could clarify how retaining useful experience while limiting the influence of outdated data affects both learning reliability and long-term performance.

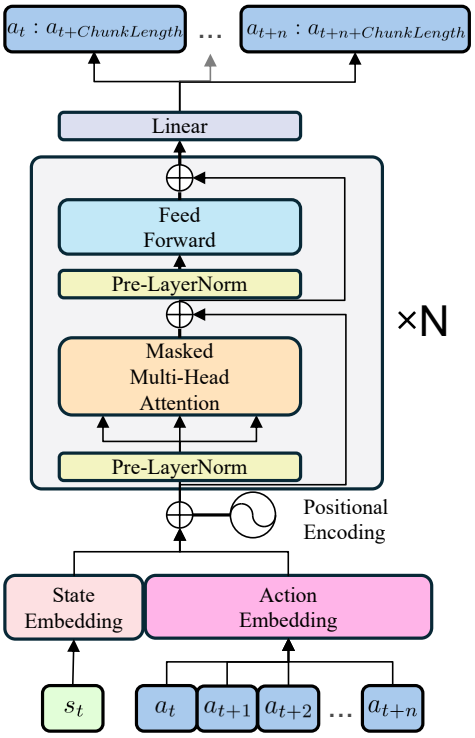


Figure 16: **HOPS Transformer policy.** State and preceding-action embeddings, augmented with positional encodings, pass through causal pre-layer-normalized Transformer blocks with residual connections. At each chunk boundary, distinct linear maps convert the final valid representation into C action means. The state token predicts the first chunk; subsequent chunks condition on previously sampled actions. The variance network, Gaussian sampling, and tanh squashing are omitted. The schematic's N denotes the stack depth L_π, independently of the best-of-N candidate count.

D Transformer Policy Architecture

HOPS generates action sequences with a compact causal Transformer. At each chunk boundary, one hidden representation predicts C actions jointly; autoregressive conditioning operates between chunks, following Eq. equation 8.

Tokenization and causal processing. Before generating chunk b, the input is $(s_t, a_t, \ldots, a_{t+bC-1})$: one state token followed by bC individual action tokens. For $b = 0$, only the state token is present. Separate embeddings map states and actions to width d:

$$\begin{aligned} h_0^{(0)} &= E_s(s_t) + p_0, \\ h_j^{(0)} &= E_a(a_{t+j-1}) + p_j, \qquad j = 1, \ldots, bC, \end{aligned} \tag{12}$$

where p_j denotes the positional encoding. Each of the L_π Transformer blocks applies layer normalization before masked multi-head self-attention and before a position-wise feed-forward network, with residual connections around both sublayers. Position j attends only to positions $0, \ldots, j$. The final valid representation, $z_{t,b} = h_{bC}^{(L_\pi)}$, predicts $(a_{t+bC}, \ldots, a_{t+(b+1)C-1})$. Thus, no predicted action appears in its own conditioning prefix, and no intermediate environment states are required.

Chunk readout and sampling. The output layer contains C distinct linear maps:

$$\boldsymbol{\mu}_{t,b,k} = W_k z_{t,b}, \qquad W_k \in \mathbb{R}^{d_a \times d}, \qquad k = 1, \ldots, C, \tag{13}$$

where $d_a = \dim(\mathcal{A})$. These bias-free maps contain no output activation and are reused across chunk boundaries. They are equivalent to one stacked linear projection into $\mathbb{R}^{Cd_a}$. All actions in a chunk therefore share a representation but have separate mean-output matrices, as analyzed in Appendix E.

A separate variance network supplies state-dependent standard deviations. Actions are sampled using tanh-squashed reparameterization:

$$a_{t+bC+k-1} = \tanh(W_k z_{t,b} + \boldsymbol{\sigma}_{t,b,k} \odot \boldsymbol{\epsilon}_{t,b,k}), \qquad \boldsymbol{\epsilon}_{t,b,k} \sim \mathcal{N}(0, I_{d_a}). \tag{14}$$

The Gaussian is diagonal across the Cd_a controls. Within a chunk, sampling one action is not required to predict another.

Chunk-autoregressive generation. After sampling a chunk, its actions are appended as individual tokens to the conditioning prefix. The policy then generates the next chunk from the extended prefix, using the same initial state s_t. A horizon of H actions requires $\lceil H/C \rceil$ generation stages, retaining only the required actions from the final chunk if necessary. Available prefix tokens can be processed in parallel, but successive chunks remain autoregressive. For the Meta-World default $C = 4$, horizons of 4 and 16 require one and four generation stages, respectively.

Differentiation during actor updates. Sampled chunks are concatenated into the sequence evaluated by the prefix-conditioned critics. Copies used to condition later chunks are detached, $\widetilde{\mathbf{a}}_t^{(bC)} = \mathrm{sg}(\mathbf{a}_t^{(bC)})$, while the original sampled actions remain differentiable as critic inputs. Detachment removes gradient paths through inter-chunk action feedback; earlier actions still receive gradients from critic prefixes that contain them. With critic parameters fixed, the actor averages the minimum twin-critic value over all retained prefixes, as in Eq. equation 10. Best-of-N selection is used only during interaction and evaluation, not during actor optimization or TD bootstrapping.

Architecture settings. Chunk length C is a task-dependent hyperparameter. On Meta-World, the actor has two Transformer layers, one attention head, and embedding width 32; the variance MLP has two hidden layers of width 128 (Table 4). The Meta-World default is $C = 4$, with $H_{\mathrm{pred}} = H_{\mathrm{exec}} = H_{\mathrm{target}} = 4$ and $H_{\mathrm{upd}} \sim \mathrm{Unif}\{4, 8, 12, 16\}$. The Box Pushing actor uses four Transformer layers of width 128. The policy remains stochastic with entropy coefficient $\alpha = 0$.

E Policy Differentiation and Gradient-Variance Analysis

We analyze two complementary aspects of chunked policy updates: the differentiation of the position-specific output embeddings and the variance of gradients applied to shared policy parameters.

E.1 Output-Embedding Differentiation and Prefix-Value Consistency

We investigate whether predicting several actions from a shared Transformer representation causes HOPS to collapse to action repetition. On Meta-World Coffee Push, using the default chunk length $C = 4$, we examine the final output embedding, executed actions, and prefix values used during actor updates. The diagnostics show that the four output mappings separate during training and that aggregate action distances remain nonzero, while differences between prefix values decrease in the later stages of training.

Shared representation and the repetition question. The analysis concerns the *final mean-output embedding layer* of the Transformer policy in Section 4, rather than its input embeddings or attention heads. For a chunk size of $C = 4$, this layer consists of four linear maps applied to the same chunk-boundary hidden representation:

$$\boldsymbol{\mu}_{t,b,k} = W_k z_{t,b}, \qquad k \in \{1, 2, 3, 4\}, \qquad W_k \in \mathbb{R}^{d_a \times d}, \tag{15}$$

where $z_{t,b} \in \mathbb{R}^d$ is the representation for chunk b, $d_a = \dim(\mathcal{A})$, and $\boldsymbol{\mu}_{t,b,k}$ is the Gaussian mean for its kth action, before squashing. In this implementation, these output embeddings contain neither bias terms nor activation functions: the four means are obtained solely through matrix multiplication. The Gaussian sampling and tanh squashing described in Section 4 occur after this linear layer. Thus, within a chunk, all four mean predictions share the same representation but have separate output matrices; autoregressive conditioning operates between chunks as in Eq. equation 8.

The prefix-value formulation provides a reason to examine possible repetition. Under the interpretation underlying the non-soft targets in Eq. equation 2, $Q^{(1)}(s_t, a_t)$ leaves subsequent actions to the continuation policy, whereas $Q^{(2)}(s_t, (a_t, a_{t+1}))$ additionally specifies the second action. Both describe a return measured from the same starting state. For exact values under a fixed policy with consistent continuation semantics, marginalizing an additional policy-sampled action recovers the shorter-prefix value in expectation. This motivates approximate agreement among policy-consistent prefix values, rather than equality for every sampled continuation or learned critic output. Since the actor objective in Eq. 10 jointly optimizes these prefix values, a natural question is whether similar value signals, together with a shared hidden representation, lead the four output mappings to produce repeated actions.

Diagnostics and aggregation. We use the four seeds 0–3 with complete records for all four diagnostic families in the W&B group.[1] At each plotted environment-interaction step, the central curve is the interquartile mean (IQM) across these four seeds, and shading denotes the 95% bootstrap confidence interval obtained by resampling seeds. Each individual quantity is shown in a separate subplot. Vertical axes are scaled separately, and initial values are retained without subtraction or a forced zero baseline.

For the output matrices, we measure the six pairwise Frobenius distances and four individual Frobenius norms:

$$D^W_{ij} = \|W_i - W_j\|_F \quad (i < j), \qquad N^W_k = \|W_k\|_F \quad (k = 1, \ldots, 4). \tag{16}$$

For behavior, we measure all six Euclidean distances $D^a_{ij} = \|a_i - a_j\|_2$ between actions within an executed evaluation chunk, where a_k denotes its kth action. For the actor-update values, we report the logged L_2 norm of the prefix-value vector per chunk and the within-chunk L_2 Q-difference statistic. In addition, for each allowed policy-update length $L \in \{4, 8, 12, 16\}$, we report the L_2 norm of pairwise differences among all prefix values within that update sequence:

$$\mathbf{q}^{(L)} = (q_1, \ldots, q_L), \qquad q_h = \min_{c \in \{1,2\}} Q^{(h)}_{\theta_c}(s_t, \mathbf{a}^{(h)}_t). \tag{17}$$

[1] The retained run IDs are `53gllkkk`, `8scvnqlz`, `yuac080f`, and `kgmopbz2`. The four earlier runs recorded only output-weight differences and norms and are excluded from all panels.

Here q_h is the value of the length-h prefix in the actor objective of Eq. 10. Each all-pair curve therefore compares prefix values within a fixed update length L. Frobenius norms describe matrix parameters, whereas L_2 norms describe action vectors and vectors of values or value differences. Q diagnostics are available only from iteration 200 onward; no earlier Q values are imputed.

Output mappings separate without action-repetition collapse. Figures 17 and 18 show that the output matrices remain unchanged during policy warmup, covering approximately the first 0.2×10^6 environment interactions. Once actor updates begin, all six pairwise weight distances increase, reaching approximately 2.7–3.7 near 10^6 interactions. The individual matrix norms also increase, indicating that training changes both the scale of the readouts and their absolute separation. These results support differentiation of the four output mappings, although absolute distances alone do not establish increasing separation relative to their norms.

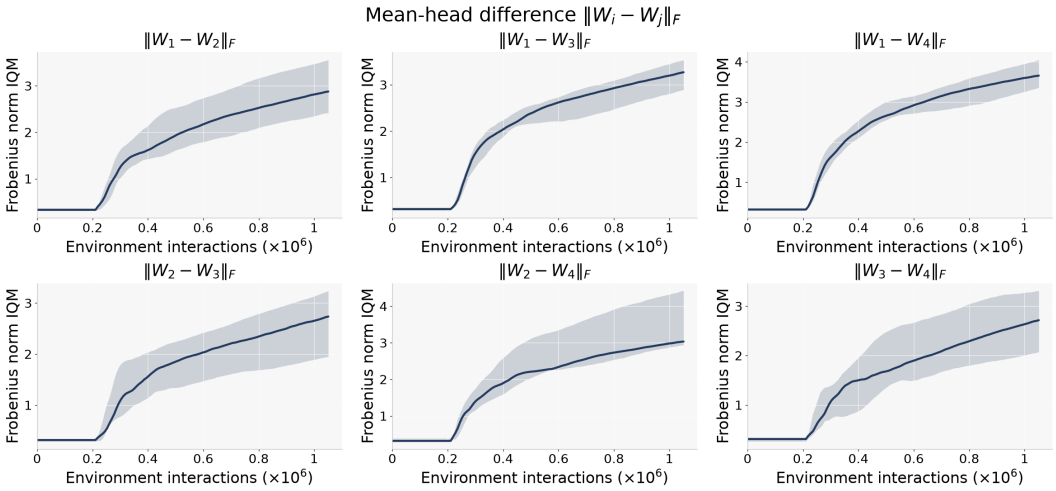


Figure 17: **Pairwise separation of the mean-output embeddings.** Each panel shows one of the six Frobenius distances $\|W_i - W_j\|_F$ on Meta-World Coffee Push. All distances increase after policy warmup, showing growing absolute separation between the four linear output mappings. Curves are IQMs over four seeds; shading denotes 95% bootstrap confidence intervals.

Parameter separation does not by itself establish action diversity: different matrices can yield similar outputs on the representations encountered by the policy. We therefore also inspect the executed actions in Fig. 19. Their pairwise distances drop sharply when actor updates begin, then fluctuate and remain appreciably above zero through the end of the recorded training. The confidence bands are broad for several pairs, reflecting variation across seeds. Together with the separating output matrices, these observations support the conclusion that the evaluated chunks do not degenerate into simple repetition of a single action. Because the policy is stochastic, the executed-action distances also reflect sampling; they should not be attributed exclusively to differences between the predicted means.

Prefix values become more consistent in later training. Figure 20 reveals a different trend in value space. The L_2 norm of the prefix-value vector per chunk continues to grow, reaching approximately 1500 near the end of the recorded training. In contrast, the within-chunk and all-pair Q-difference statistics peak around 0.4–0.5×10^6 interactions and then decline. Thus, after the initial learning phase, prefix-value differences shrink even as the overall value scale increases. This is consistent with the expectation that policy-generated prefixes can have similar predicted returns without containing identical actions.

Longer sequences have larger aggregate all-pair difference norms. However, increasing L also increases the number of compared pairs, so the ordering across lengths alone does not establish greater disagreement per pair. The relevant common trend is the decline within each fixed length during later training. The differences remain nonzero, and their early trajectories rise; the evidence therefore supports increasing consistency in later training, rather than monotonic or exact convergence

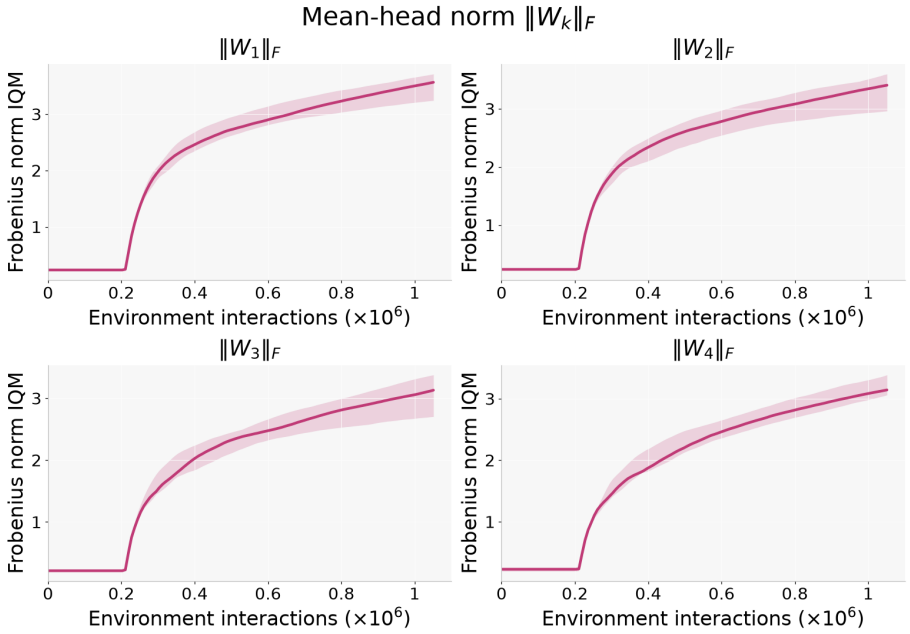


Figure 18: **Scale of the mean-output embeddings.** Each panel shows the Frobenius norm $\|W_k\|_F$ of one output matrix on Meta-World Coffee Push. Norms remain constant before actor updates and increase afterward, providing scale context for Fig. 17. Curves are four-seed IQMs with 95% bootstrap confidence intervals.

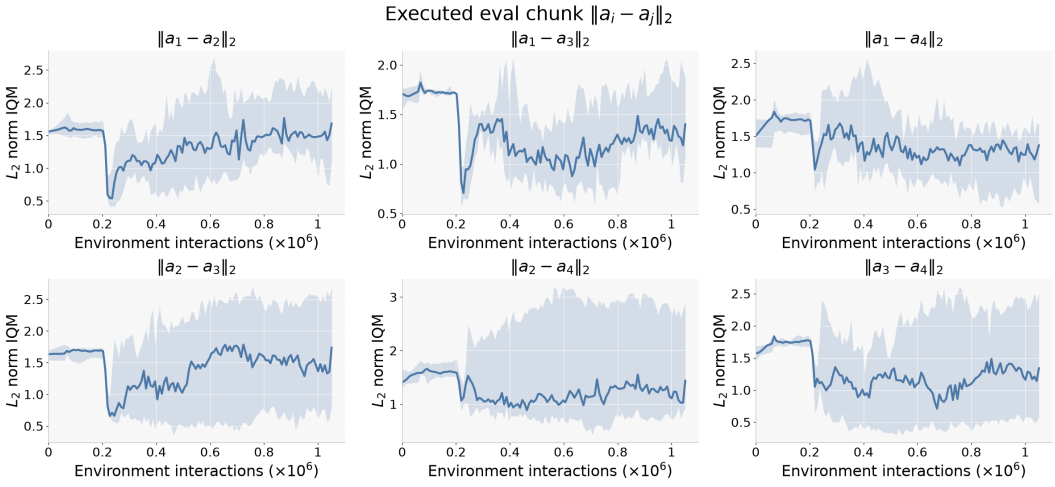


Figure 19: **Action differences within executed evaluation chunks.** The six panels report $\|a_i - a_j\|_2$ for all pairs in a four-action chunk on Meta-World Coffee Push. Despite an initial reduction after warmup, aggregate action distances remain nonzero during subsequent training. Curves are four-seed IQMs with 95% bootstrap confidence intervals.

throughout training. Taken together, the diagnostics support distinct within-chunk controls whose prefix values become more closely aligned as learning progresses beyond its initial phase.

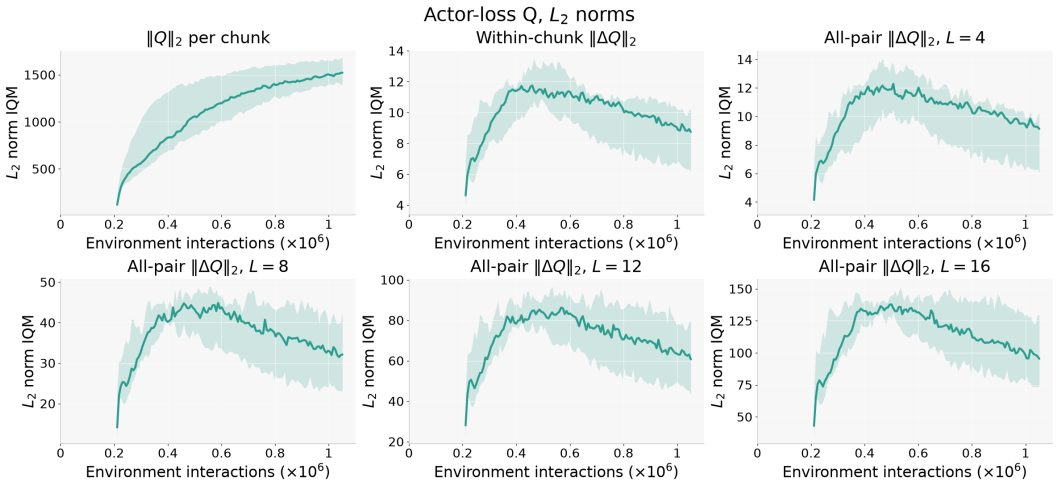


Figure 20: **Magnitude and differences of actor-update prefix values.** Diagnostics on Meta-World Coffee Push: the per-chunk value-vector L_2 norm, the within-chunk L_2 Q-difference statistic, and all-pair L_2 Q-difference statistics at $L = 4, 8, 12, 16$. Each all-pair panel compares all prefix Q values within its indicated update length. Value magnitudes increase, whereas differences decline after their mid-training peaks, consistent with later-stage agreement among prefix values. Each panel uses its own vertical scale. Curves are four-seed IQMs with 95% bootstrap confidence intervals; Q logging starts at iteration 200.

E.2 Gradient Variance under Shared Policy Parameters

Appendix D of Tian et al. (2026) analyzes averaging per-horizon critic gradients under a homoscedastic, equicorrelated covariance model. We adapt its gradient-averaging argument to the HOPS actor. The relevant architectural property is that the C mean predictions within a chunk use one hidden representation: their paths differ at $W_1, \ldots, W_C$, but meet at the same Transformer computation. Consequently, their backward signals contribute to updates of the same upstream parameters.

Actor-gradient decomposition. Fix the task-specific chunk length C and the current policy and critic parameters, and condition on an integer actor-update length $L \geq 1$. Let q_h be the actor's prefix value from Eq. equation 17, and let ξ collect the sampled replay state and reparameterization noise. The sample actor loss and its gradient for a shared scalar parameter w are

$$\begin{aligned} \widehat{\mathcal{L}}_L(\xi) &= \frac{1}{L}\sum_{h=1}^{L} \ell_h(\xi), & \ell_h(\xi) &= -q_h(\xi), \\ \bar{g}_L(w) &= \frac{\partial \widehat{\mathcal{L}}_L}{\partial w} = \frac{1}{L}\sum_{h=1}^{L} g_h(w), & g_h(w) &= \frac{\partial \ell_h}{\partial w}. \end{aligned} \tag{18}$$

These derivatives follow the implemented computation graph, including tanh-squashed sampling and the detached action copies used to condition later chunks (Section 4). The critic is held fixed during this update.

To expose the shared path, let ψ denote the Transformer parameters, $J_b = \partial z_{t,b}/\partial \psi$, and $u_{h,b,k} = \nabla_{\boldsymbol{\mu}_{t,b,k}} \ell_h$. The contribution through the mean-output branch is

$$\nabla_\psi \ell_h|_{\text{mean}} = \sum_{b=0}^{B(L)-1} J_b^\top \left(\sum_{k=1}^{r_b(L)} W_k^\top u_{h,b,k} \right), \tag{19}$$

where $B(L) = \lceil L/C \rceil$ and $r_b(L) = \min\{C, L - bC\}$, as in Eq. equation 8, and $u_{h,b,k} = 0$ when action $bC + k$ lies outside prefix h. For each chunk, the same J_b receives signals from its retained output maps, with C such maps for a full chunk. Distinct W_k and action derivatives can nevertheless produce different gradient contributions. The covariance analysis below uses the complete $g_h(w)$, so any other differentiable paths to w are included as well.

Variance under the covariance model. For any joint distribution of the prefix gradients with finite second moments, their conditional variance satisfies

$$\mathrm{Var}(\bar{g}_L(w) \mid L) = \frac{1}{L^2}\left[\sum_{h=1}^{L} \mathrm{Var}(g_h(w) \mid L) + 2\sum_{i<j} \mathrm{Cov}(g_i(w), g_j(w) \mid L) \right]. \tag{20}$$

Following the covariance approximation in Tian et al. (2026, Appendix D), assume at this fixed L that

$$\begin{aligned} \mathrm{Var}(g_h(w) \mid L) &= \sigma^2_{w,L}, \\ \mathrm{Cov}(g_i(w), g_j(w) \mid L) &= \rho_{w,L}\sigma^2_{w,L} \quad (i \neq j), \qquad 0 \leq \rho_{w,L} < 1. \end{aligned} \tag{21}$$

Substitution gives the actor-side prediction

$$\boxed{\mathrm{Var}(\bar{g}_L(w) \mid L) = \sigma^2_{w,L}\frac{1 + (L-1)\rho_{w,L}}{L}.} \tag{22}$$

For nonzero constituent variance and $L > 1$, this is smaller than the variance of one prefix gradient when $\rho_{w,L} < 1$. For a single full chunk, $L = C$, the factor is $(1 + (C-1)\rho_{w,C})/C$; when $C = 1$, the factor equals 1. Under the Meta-World default $C = 4$, the factor becomes $(1 + 3\rho_{w,4})/4$: it equals $1/4$ for uncorrelated contributions, $5/8$ at correlation $1/2$, and approaches 1 as correlation approaches 1. For the Meta-World update lengths, the factors are

$$\frac{\mathrm{Var}(\bar{g}_L(w) \mid L)}{\sigma^2_{w,L}} = \begin{cases} (1 + 3\rho_{w,4})/4, & L = 4, \\ (1 + 7\rho_{w,8})/8, & L = 8, \\ (1 + 11\rho_{w,12})/12, & L = 12, \\ (1 + 15\rho_{w,16})/16, & L = 16. \end{cases} \tag{23}$$

Their uncorrelated limits are respectively 0.25, 0.125, 0.0833, and 0.0625 times the variance of a constituent gradient. These are theoretical model values, not measured variance reductions. For $L > C$, the covariance assumption also covers prefixes ending in different chunks; sharing weights alone does not establish this assumption or a monotonic improvement across L.

Random update lengths and policy-level variance. Let $\mathcal{H}_{\mathrm{upd}}$ denote the support of the configured actor-update length distribution, and let $p_\ell = \Pr(L = \ell)$. Writing $m_{w,L} = \mathbb{E}[\bar{g}_L(w) \mid L]$, the variance of the resulting update obeys

$$\mathrm{Var}(\bar{g}_L(w)) = \sum_{\ell \in \mathcal{H}_{\mathrm{upd}}} p_\ell \sigma^2_{w,\ell} \frac{1 + (\ell - 1)\rho_{w,\ell}}{\ell} + \mathrm{Var}_L(m_{w,L}). \tag{24}$$

If $\sigma^2_{w,L} = \sigma^2_w$ and $\rho_{w,L} = \rho_w$ are independent of L, its first term simplifies to $\sigma^2_w[\rho_w + (1 - \rho_w)\mathbb{E}_L(1/L)]$. For the default Meta-World sampling distribution $L \sim \mathrm{Unif}\{4, 8, 12, 16\}$, $\mathbb{E}_L(1/L) = 25/192$. The additional term disappears only if the conditional mean gradients agree across lengths.

For a parameter block ϑ, define its total gradient variance as $\mathcal{V}_L(\vartheta) = \mathrm{tr}\,\mathrm{Cov}(\nabla_\vartheta \widehat{\mathcal{L}}_L \mid L)$. Let ψ_{sh} collect all shared parameters for which Eq. equation 21 is assumed, and partition the full actor as $\phi = (\psi_{\mathrm{sh}}, W_1, \ldots, W_C, \eta)$, where η contains any remaining parameters, such as separately parameterized scale outputs. Then

$$\mathcal{V}_L(\phi) = \sum_{w \in \psi_{\mathrm{sh}}} \sigma^2_{w,L} \frac{1 + (L - 1)\rho_{w,L}}{L} + \sum_{k=1}^{C} \mathcal{V}_L(W_k) + \mathcal{V}_L(\eta). \tag{25}$$

This trace decomposition does not assume independence between parameter blocks. Additional shared scale-network parameters can be included in ψ_{sh} under the same covariance hypothesis; η is empty if no parameters remain.

The output embeddings have separate parameters for different within-chunk positions, so their gradients contribute to distinct output matrices rather than to one shared output matrix. For example, $\nabla_{W_k} \ell_h = \sum_b u_{h,b,k} z_{t,b}^\top$. Section E.1 addresses these position-specific mappings through their norms, pairwise distances, and resulting action differences. Those diagnostics characterize their differentiation; they do not estimate the gradient covariance terms in Eq. equation 25. Thus, the two analyses are complementary: under the stated covariance model, averaging prefix gradients can reduce variance on shared parameters, while the output-embedding diagnostics show how the unshared mean readouts develop distinct mappings.

F PSEUDO-CODE

Algorithm 1: HOPS

Initialization: Policy ϕ; twin critics $\theta = (\theta_1, \theta_2)$; target critics $\bar{\theta} \leftarrow \theta$; replay buffer $\mathcal{B}$; reset environment $\rightarrow s$.

Input: Segment length M; task-specific chunk length C; rollout / target horizons (H_s, H_t); actor horizon bounds $(h_{\min}, h_{\max})$; critic window bounds $(\ell_{\min}, \ell_{\max})$; candidates N; updates (U, N_c, N_p); target rate τ.

Function `Generate(`s, H`)`**:**

- $A \leftarrow [\,]$; $c \leftarrow \varnothing$;
- **while** $|A| < H$ **do**
  - $d \leftarrow \min(C, H - |A|)$;
  - Sample a C-action block from $\pi_\phi(\cdot \mid s, c)$ using tanh-squashed reparameterization;
  - Retain its first d actions as A_{new};
  - Append A_{new} to A;
  - $c \leftarrow \text{sg}(A)$;
- **end**
- **return** A;

repeat

- **Data collection:** start an empty segment;
- **while** *fewer than M transitions have been collected* **do**
  - Draw N candidates $A^{(n)}$ using $\text{Generate}(s, H_s)$ without gradient tracking;
  - $n^\star \leftarrow \arg\max_n \frac{1}{2} \sum_{q=1}^{2} Q_{\theta_q}(s, A^{(n)})$;
  - Execute $A^{(n^\star)}$ in order until chunk, segment, or episode end; record transitions, update s, and reset on episode termination;
- **end**
- Store the segment in $\mathcal{B}$; retain the current environment state;
- **if** $\mathcal{B}$ *is not ready* **then** continue;
- **for** $u \leftarrow 1$ **to** U **do**
  - Sample a segment batch;
  - Cache bootstrap values without gradient tracking, using one $\text{Generate}(s', H_t)$ sample per bootstrap state and target critics $Q_{\bar{\theta}}$;
  - **for** $k \leftarrow 1$ **to** N_c **do**
    - Sample start i and length $\ell \in \{\ell_{\min}, \ldots, \ell_{\max}\}$; truncate at the segment end;
    - Update both Transformer critics with masked non-soft multi-step return targets;
    - $\bar{\theta} \leftarrow \tau\theta + (1 - \tau)\bar{\theta}$ when scheduled;
  - **end**
  - **if** *policy warmup is complete* **then**
    - Sample a fresh segment batch;
    - **for** $k \leftarrow 1$ **to** N_p **do**
      - Select start states s; draw $h \sim \mathcal{U}\{h_{\min}, \ldots, h_{\max}\}$;
      - $H_\pi \leftarrow C\lceil h/C \rceil$;
      - $a_{0:H_\pi - 1} \leftarrow \text{Generate}(s, H_\pi)$;
      - Hold critic parameters fixed and update ϕ by maximizing;
      - $$\mathbb{E}\left[\frac{1}{H_\pi} \sum_{j=0}^{H_\pi - 1} \min_{q \in \{1,2\}} Q_{\theta_q}(s, a_{0:j})\right];$$
    - **end**
  - **end**
- **end**

until *convergence*;

sg detaches the generated action prefix used as decoding context; sampled actions remain differentiable as critic inputs. $Q(s, A)$ scores the complete chunk, while $Q(s, a_{0:j})$ scores a prefix. Actor updates average all prefix values with equal weights.

G COMPUTATIONAL COSTS

We compare the training time and model size of HOPS with CrossQ, SimbaV2, and T-SAC on Meta-World. We report wall-clock training time per 10^6 environment interactions, together with the actor and critic parameter counts.

Hardware and software. The timing comparison uses a Lenovo IdeaCentre Gaming 5 14ACN6 desktop equipped with one 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 system RAM (14.98 GiB visible to the operating system). 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.

Reporting convention. Each environment interaction corresponds to one primitive environment transition, irrespective of the action-chunk length. Transitions are summed across parallel environments. Table 3 reports training time in hours per 10^6 interactions and exact actor and critic parameter counts. The parameter counts are taken directly from the corresponding runs, without an additional multiplier for critic networks. The SimbaV2 timing run uses a UTD ratio of 2, matching its cluster configuration.

Table 3: Computational costs and model sizes on Meta-World. Training time is 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	5.8	45,320	1,061,120
HOPS (ours)	4.77	83,600	1,061,120

Model size. As described in Section 4, HOPS replaces T-SAC's one-step MLP actor with a compact chunk-autoregressive Transformer while retaining its prefix-conditioned Transformer critics. On Meta-World, the HOPS actor uses two Transformer layers with hidden dimension 32. HOPS has 83,600 actor parameters and 1,061,120 critic parameters, totaling 1,144,720. Compared with T-SAC, HOPS adds 38,280 actor parameters while keeping the same reported critic parameter count. The total parameter count therefore increases by approximately 3.5%, from 1,106,440 to 1,144,720. For comparison, CrossQ and SimbaV2 have 8,672,772 and 4,707,949 actor and critic parameters in total, respectively.

Training time and interpretation. HOPS requires 4.77 h per 10^6 environment interactions, approximately 17.8% less time than T-SAC's reported approximately 5.8 h. CrossQ and SimbaV2 require 2.10 h and 3.25 h, respectively; HOPS therefore takes approximately $2.27\times$ and $1.47\times$ as long. Time comparisons use the unrounded second counts where available. Thus, HOPS has substantially fewer parameters than CrossQ and SimbaV2 but a higher training time per interaction, illustrating that parameter count alone does not determine wall-clock cost.

These comparisons characterize complete learning configurations, including their respective architectures and update schedules. They do not isolate the computational effect of extending the policy-update horizon or of individual components such as critic-guided candidate selection. The reported times also do not establish the time required to reach a common success threshold.

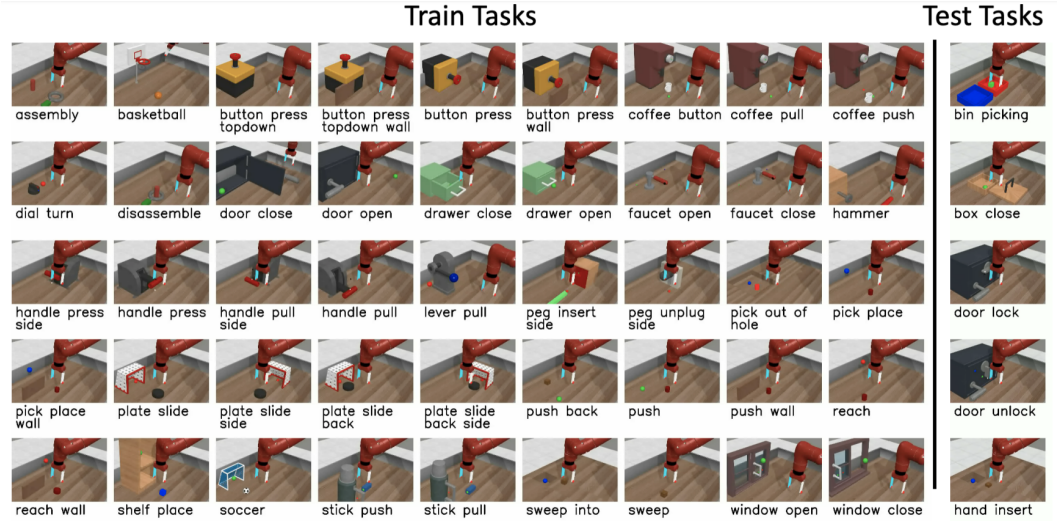


Figure 21: Meta-World tasks (Yu et al., 2020).

H Experiment Description

H.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 21 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.

H.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 ≤ 0.05 m and orientation error ≤ 0.5 rad.

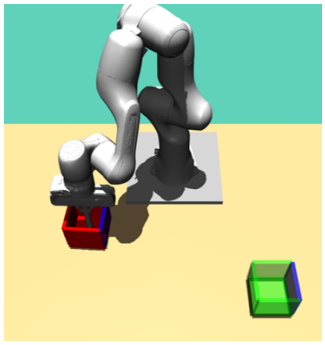


Figure 22: Box Pushing task (Otto et al.).

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

$$R_{\text{rod}} = \text{clip}(\|\boldsymbol{p} - \boldsymbol{h}_{\text{pos}}\|,\, 0.05,\, 10) + \text{clip}\big(\tfrac{2}{\pi}\arccos|\boldsymbol{h}_{\text{rot}} \cdot \boldsymbol{h}_0|\,,\, 0.25,\, 2\big)\,, \tag{26}$$

$$\tau_t = \sum_{i=1}^{7} (a_t^i)^2, \tag{27}$$

$$\text{err}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_{i:\,|q_i| > |q_i^b|} \big(|q_i| - |q_i^b|\big) + \sum_{j:\,|\dot{q}_j| > |\dot{q}_j^b|} \big(|\dot{q}_j| - |\dot{q}_j^b|\big)\,, \tag{28}$$

$$R_{\text{position}} = \|\boldsymbol{p} - \boldsymbol{p}_{\text{target}}\|, \tag{29}$$

$$R_{\text{rotation}} = \tfrac{1}{\pi}\arccos|\boldsymbol{r} \cdot \boldsymbol{r}_{\text{target}}|\,. \tag{30}$$

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

I BASELINES AND REPRODUCIBILITY

T-SAC is our primary systems-level comparator because it uses the same prefix-conditioned Transformer critics and online training regime as HOPS while retaining a conventional one-step actor. For the main systems-level comparison, HOPS uses five independent random seeds per task and T-SAC uses eight; ablation seed counts are reported separately in App. B. We additionally compare HOPS with PPO, PPO+gSDE, PPO+GTrXL, SAC, CrossQ, TCP, BBRL, TOP-ERL, Q-Chunking, and SimbaV2 where results under the final-timestep success metric are available. All Box Pushing results reported in this subsection use the dense-reward setting. Throughout all experiments and comparisons, an evaluation episode is counted as successful only if the task success indicator is positive at its final timestep. We do not use success-at-any-time or maximum-over-trajectory success.

Several Meta-World baseline results are taken from prior publications or official releases. All imported results included here use the same final-timestep success definition; they retain their original seed counts, interaction budgets, evaluation frequencies, and other evaluation settings. When these other protocol details differ from ours, we treat the results as contextual rather than strictly protocol-matched comparisons. We do not perform additional large-scale hyperparameter sweeps for imported methods. SAC and CrossQ on dense-reward Box Pushing are tuned within our experimental pipeline.

Comparison budgets. The main Meta-World comparison in Table 1 uses 5×10^6 environment interactions. We report aggregate scores and the corresponding uncertainty intervals at the specified interaction budgets. Comparisons are restricted to each method's available interaction range. Longer-budget baseline endpoints are discussed separately as descriptive comparisons. The reported method-wise intervals do not constitute a statistical test of a between-method difference.

Table 2 reports the comparison at 8×10^6 interactions. CrossQ's final evaluation budget is approximately 10.049×10^6 interactions, and comparisons involving CrossQ are restricted to that budget. The comparison between HOPS near 8.89×10^6 interactions and T-SAC near 20×10^6 is descriptive and uses comparable, rather than identical, central success estimates. We do not interpret it as an exact threshold-crossing ratio.

Metrics and uncertainty. Let $y_{m,t,s,e}(c) \in \{0, 1\}$ denote terminal success for method m, task t, training seed s, evaluation episode e, and checkpoint c. Each seed is evaluated for E_t episodes, and its terminal-success rate is

$$\widehat{p}_{m,t,s}(c) = \frac{1}{E_t} \sum_{e=1}^{E_t} y_{m,t,s,e}(c).$$

We first compute an IQM across training seeds separately for each task,

$$q_{m,t}(c) = \mathrm{IQM}_{s \in \mathcal{S}_{m,t}^{\mathrm{cmp}}} \left(\widehat{p}_{m,t,s}(c) \right),$$

and then report the unweighted arithmetic mean of the task-wise IQMs,

$$M_m(c) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} q_{m,t}(c).$$

Thus, all benchmark-level results are means of task-wise IQMs; we do not pool task–seed observations before computing a single IQM. For the main Meta-World result, $|\mathcal{T}| = 50$. For Box Pushing, $|\mathcal{T}| = 1$, so the reported value is its task-wise IQM.

We define the IQM as the central-50% trimmed mean,

$$\mathrm{IQM}(x_1, \ldots, x_S) = 2 \int_{0.25}^{0.75} \widehat{F}_S^{-1}(u) \, \mathrm{d}u,$$

where $\widehat{F}_S^{-1}$ is the empirical quantile function.

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 common variants are *PPO–Penalty*, which uses KL regularization, and *PPO–Clip*, which uses a clipped surrogate objective. We evaluate PPO–Clip because of 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 that uses twin Q–networks to mitigate overestimation and an entropy term to encourage exploration. We use the SAC implementation released with CrossQ (Bhatt et al., 2024). **Seeds: 20 (Meta–World ML1).**

PPO+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 together with a state–independent log–standard–deviation head following Huang et al. (2022). **Seeds: 4.**

PPO+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. Specifically, disturbances are generated as $\epsilon_t = \Theta s$, where s is the activation of the final hidden layer and Θ is resampled from a Gaussian distribution 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 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 by parameterizing policies with ProMPs (Paraschos et al., 2013). This formulation supports trajectory-level and non–Markovian objectives, although trajectory-level search may 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., 2024) augments episodic policy updates with step–level learning signals, narrowing the gap between episodic and step–based reinforcement learning while retaining smooth parameter–space exploration. **Seeds: 20.**

TOP–ERL. Transformer–based Off-Policy Episodic Reinforcement Learning (TOP–ERL) optimizes a distribution over motion–primitive parameters using (i) a KL–constrained trust region and (ii) a temporally structured covariance that induces smooth, correlated exploration throughout an episode. Our instantiation uses ProDMP (Li et al., 2023) as the trajectory generator. Unless otherwise stated, we use an adaptive scale (entropy) schedule and per–dimension normalization of the primitive parameters. **Seeds: 8.**

CrossQ. CrossQ (Bhatt et al., 2024) is an off–policy SAC variant that removes target networks and applies batch normalization in the critic, enabling strong sample efficiency at an update–to–data ratio of $\mathrm{UTD} = 1$. We follow the authors' reference implementation, using a single batch–normalized critic without target networks, default temperature tuning, and the recommended hyperparameters unless otherwise stated. **Seeds: 4 (Meta–World ML1), 5 (dense-reward Box Pushing).** Training on dense-reward Box Pushing is capped at 10×10^6 environment interactions because of the experiment budget; by that point, wall-clock training time exceeds 24 h.

J HYPERPARAMETERS

Table 4: Hyperparameters for the Meta-World experiments. Episode length $T = 500$. The HOPS chunk length C is a task-dependent hyperparameter; its Meta-World default is $C = 4$, unless otherwise specified.

	PPO	gSDE	GTrXL	SAC	CrossQ	TCP	BBRL	TOP-ERL	T-SAC	HOPS
Data collection										
number samples	16000	16000	19000	1000	1	16	16	2	4×125	4×100
GAE λ	0.95	0.95	0.95	n.a.	n.a.	0.95	n.a.	n.a.	n.a.	n.a.
discount factor	0.99	0.99	0.99	0.99	0.99	1	1	1.0	0.99	0.99
Trust region										
ϵ_μ	n.a.	n.a.	n.a.	n.a.	n.a.	0.005	0.005	0.005	n.a.	n.a.
ϵ_Σ	n.a.	n.a.	n.a.	n.a.	n.a.	0.0005	0.0005	0.0005	n.a.	n.a.
trust region loss coef.	n.a.	n.a.	n.a.	n.a.	n.a.	1	10	1.0	n.a.	n.a.
Optimization										
optimizer	adam	adam	adam	adam	adam	adam	adam	adam	adamw	adamw
epochs	10	10	5	1000	1	50	100	15	20	40
learning rate	3e-4	1e-3	2e-4	3e-4	3e-4	3e-4	3e-4	1e-3	2.5e-4	3e-5
use critic	True	True	True	True	True	True	True	True	True	True
epochs critic	10	10	5	1000	1	50	100	50	100	100
learning rate critic	3e-4	1e-3	2e-4	3e-4	3e-4	3e-4	3e-4	5e-5	2.5e-5	2.5e-5
number minibatches	32	n.a.	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	512
buffer size	n.a.	n.a.	n.a.	1e6	1e6	n.a.	n.a.	3000	5000×125	$100,000 \times 100$
learning starts	0	0	n.a.	10000	5000	0	0	2	200	200
temperature warmup	0	0	0	0	0	0	0	0	10000	0
polyak weight	n.a.	n.a.	n.a.	5e-3	1.0	n.a.	n.a.	5e-3	5e-3	1.0 (hard copy)
SDE sampling frequency	n.a.	4	n.a.	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	0
Normalization and clipping										
normalized observations	True	True	False	False	False	True	False	False	False	False
normalized rewards	True	True	0.05	False	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.	n.a.
reward clip	10.0	10.0	10.0	n.a.	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.	n.a.
importance ratio clip	0.2	lin_0.3	0.1	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
Architecture										
hidden layers	$[128, 128]$	$[128, 128]$	n.a.	$[256, 256]$	$[256, 256]$	$[128, 128]$	$[32, 32]$	$[128, 128]$	$[128, 128]$	variance: $[128, 128]$
hidden layers critic	$[128, 128]$	$[128, 128]$	n.a.	$[256, 256]$	$[2048, 2048]$	$[128, 128]$	$[32, 32]$	n.a.	n.a.	n.a.
hidden activation	tanh	tanh	relu	relu	relu	relu	relu	leaky_relu	leaky_relu	gelu / leaky_relu
orthogonal initialization	Yes	No	xavier	fanin	fanin	Yes	Yes	Yes	fanin	GPT / fanin
initial std	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	state-dependent
number of heads	–	–	4	–	–	–	–	8	4	1 / 4
dims per head	–	–	16	–	–	–	–	16	32	32 / 32
number of attention layers	–	–	4	–	–	–	–	2	2	2 / 2