Action Chunking Proximal Policy Optimization with Feedback Correction

Sanghyun Hahn[*]
Cornell University
steve0221@cs.cornell.edu

Jonghyun Choi
Seoul National University[†]
jonghyunchoi@snu.ac.kr

Abstract

Action chunking provides temporal abstraction in reinforcement learning by selecting short action sequences instead of individual actions, but many existing approaches face two limitations in high-dimensional robotic control. First, many rely on value functions over action chunks, which can be difficult to learn as action dimensionality and chunk length grow. Second, executing chunks open-loop removes within-chunk feedback, limiting reactivity in contact-rich tasks. We present Action Chunking PPO (ACPPO), a PPO extension that uses a chunked actor while retaining a standard state-value critic, thereby avoiding chunked Q-functions. We further propose ACPPO-Corr, which augments the chunk planner with a stepwise feedback corrector that adjusts planned actions online within each chunk. Across 25 simulated robotics tasks from IsaacGym and Bi-DexHands, spanning locomotion, arm manipulation, and dexterous hand-object interaction, ACPPO-Corr achieves the strongest aggregate performance among evaluated methods and performs best on both decision-frequency-sensitive and decision-frequency-neutral task subsets. Ablations show that moderate chunk lengths work best and that corrector regularization is important for balancing chunk-level planning with local feedback. These results suggest that action chunking can be effective in online PPO when chunk-level planning is paired with closed-loop correction. The code is available at: https://github.com/hshhahn/ACPPO.git

1 Introduction

Action Chunking [1, 2, 3, 4, 5, 6] has been a popular design choice in robot imitation learning, where the policy predicts a sequence of actions instead of a single action at each decision step. This reduces the effective control horizon at inference, mitigates compounding errors in the continuous action space [7, 8], and captures non-Markovian behaviors inherent in human demonstrations [1].

Recent work has incorporated action chunking into reinforcement learning (RL), treating action chunks as temporally extended actions in a semi-Markov decision process [9, 10, 11, 12, 13, 14, 15, 16] or exploiting chunked critics [17, 18, 19]. Compared to stepwise control, action chunking RL can offer several advantages. First, multi-step actions shorten the decision horizon and accelerate reward propagation, which is beneficial in long-horizon problems [10, 12, 15]. Second, chunked policies allow temporally coherent exploration, which can lead to deeper state-space coverage and improved exploration efficiency [10, 14, 11].

However, these benefits come with two major challenges when applied to high-dimensional robotic control: chunked action-value function learning and the loss of within-chunk reactivity. First, most prior action-chunking RL methods rely on chunked Q-functions, $Q(s_t, a_{t:t+h-1})$, requiring the

[*]Work done at Seoul National University.
[†]JC is with ECE, IPAI and ASRI in SNU and a corresponding author.

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

critic to evaluate high-dimensional action sequences. Prior work shows that this can make value learning more difficult, cause overestimation and instability, and make policy extraction increasingly challenging as the chunk length grows [17, 15, 16]. As a result, some prior approaches rely on offline pre-training [10, 15, 17] or use heavier architectures, such as transformers, to model long action sequences [14, 18]. This limitation is particularly important in our setting, where the policy must operate in high-dimensional environments, and obtaining a reliable offline dataset is challenging.

Second, executing a predicted action chunk in an open-loop manner causes the loss of within-chunk reactivity: after the action sequence has been selected, the actions cannot adapt to environmental stochasticity, contact changes, or unexpected perturbations. This loss of within-chunk feedback is especially problematic in contact-rich robotic manipulation, including dexterous hand control [20], where contact dynamics can change rapidly and successful control often depends on timely closed-loop correction [21, 22].

In this work, we propose Action Chunking PPO (ACPPO), an extension of PPO that brings action chunking into a fully online, on-policy RL setting while keeping a standard state-value critic. ACPPO introduces temporal abstraction to the actor through a chunked planner that predicts short action sequences, while retaining the state-only critic of PPO [23]. The critic itself is not chunked: it remains a standard state-value function $V(s)$, while the PPO surrogate uses a chunk-level advantage computed from an h-step chunk return. Thus, ACPPO introduces multi-step actor credit assignment without learning a chunked Q-function.

Building on this formulation, we propose ACPPO-Corr, which adds a stepwise feedback corrector that adjusts the planned actions every step. This combines low-frequency chunk planning with high-frequency closed-loop control, recovering within-chunk reactivity while maintaining the benefits of action chunking. Crucially, instead of relying on prior demonstrations, offline pre-training, or a frozen pretrained chunked policy, we jointly train the planner and corrector within a single fully online, on-policy RL framework. Empirically, ACPPO-Corr achieves the strongest performance across 25 simulated robotics tasks from IsaacGym [24] and Bi-DexHands [25], including both decision-frequency-sensitive and decision-frequency-neutral subsets.

2 Related Work

Action Chunking in Reinforcement Learning. Multi-step returns have been a popular choice in RL for trading off the bias of one-step TD targets against the variance of Monte Carlo returns [26]. Recent action chunking RL methods extend this idea to temporally extended control by learning over short action sequences, often through chunked critics or action-sequence policies. One line of work focuses on critic-side temporal abstraction while keeping a stepwise policy or omitting an explicit actor [17, 19, 27, 18]. These methods move temporal abstraction to the critic by learning values over short action sequences, reducing the compounding errors from the bootstrapped value estimates [27]. A second line of work jointly learns chunked actors and chunked critics in offline, offline-to-online, or online settings [12, 10, 15, 13, 14]. These methods benefit from chunked actors, which produce temporally coherent action sequences and can encourage coherent exploration. Despite differences in their settings, a large fraction of these methods rely on chunked Q-functions and are based on off-policy reinforcement learning methods, often augmented with heavier architectures, such as transformers. This limitation is particularly important in our setting: in high-dimensional continuous control, collecting expert demonstrations is costly, whereas parallel simulators make online rollouts practical [24, 28]. Consequently, PPO-type on-policy methods remain strong baselines on high-DoF manipulation benchmarks [29, 25]. Our work studies action chunking in a fully online, on-policy PPO setting with a chunked actor and a standard state-value critic. This differs both from online chunked actor-critic methods based on chunked Q-functions [14, 13], and from VLA post-training that applies PPO-style updates to pretrained action-chunking policies while relying on expert demonstrations [11]. In contrast to chunked Q-learning, the critic input in ACPPO does not grow with the chunk length: the critic remains $V_\phi(s)$, and chunking is applied by the actor and the chunked advantage rather than through a chunked Q-function $Q(s, a_{t:t+h-1})$.

Residual correction and reactive execution. A related line of work addresses the loss of reactivity or execution mismatch in chunked policies, either by learning corrective residuals or by modifying real-time inference. Classical residual RL augments a frozen controller with a learned corrective policy [30]. In robot learning from demonstrations, Ankile et al. [31] learn a closed-loop residual

policy on top of a frozen behavior-cloning chunked planner, and Ankile et al. [32] study off-policy residual fine-tuning of such frozen base policies on high-DoF robots. For pretrained chunked policies, Xue et al. [21] combine a slow chunk-level planner with a fast tactile controller, while Liu et al. [33] improve reactivity through test-time closed-loop resampling of chunk predictions. For real-time deployment of pretrained chunked policies, Black et al. [34] generate the next chunk while executing the current one, Sendai et al. [35] add a per-step correction head to off-the-shelf VLA action chunks, and Wang et al. [36] learn corrective adjustments on top of a pretrained policy through masked action chunking. ACPPO-Corr is closest to this line of work, but differs in the learning setup: instead of correcting a pretrained, frozen chunked policy, we jointly train a chunk planner and a stepwise corrector within a fully online, on-policy RL formulation. Therefore, the corrector is not an add-on execution module, but part of the policy optimized jointly with the chunk planner from scratch.

3 Preliminaries

Problem setup. We consider a Markov decision process (MDP) [37] $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $s_t \in \mathcal{S}$ is the state, $a_t \in \mathcal{A}$ is the action, $r_t = r(s_t, a_t)$ is the reward, and $P(s_{t+1} \mid s_t, a_t)$ is the transition kernel. The behavior of the agent is determined by a stochastic policy $\pi_\theta(a_t \mid s_t)$, and the objective is to maximize the expected discounted return

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right].$$

For continuous action spaces, we assume $\mathcal{A} \subset \mathbb{R}^{d_a}$ and parameterize the policy as a diagonal Gaussian.

Proximal Policy Optimization. Given a batch of trajectories collected by a policy $\pi_{\theta_{\text{old}}}$, Proximal Policy Optimization (PPO) [23] performs multiple epochs of first-order updates on a clipped surrogate objective that controls the deviation of π_θ from $\pi_{\theta_{\text{old}}}$. Defining the importance ratio

$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},$$

PPO maximizes the surrogate:

$$J_{\text{PPO}}(\theta) = \mathbb{E}\left[\min\Big(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\right],$$

where $\hat{A}_t$ is the advantage estimate. To compute this advantage, a state-value critic $V_\phi(s_t)$ is trained jointly with the actor. We define the one-step temporal-difference residual as

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t).$$

$\hat{A}_t$ is obtained by the Generalized Advantage Estimate (GAE) [38]

$$\hat{A}_t = \hat{A}_t^{\text{GAE}} = \delta_t + \gamma\lambda\hat{A}_{t+1}^{\text{GAE}}.$$

Finally, the corresponding value target to train the critic is

$$\hat{R}_t = \hat{A}_t^{\text{GAE}} + V_\phi(s_t).$$

Multi-step returns and control. An h-step bootstrapped return aggregates rewards over a short horizon:

$$G_t^{(h)} = \sum_{j=0}^{h-1} \gamma^j r_{t+j} \ + \ \gamma^h V_\phi(s_{t+h}).$$

These multi-step returns interpolate between one-step TD targets and Monte Carlo returns, yielding a bias-variance tradeoff. In our setting, a chunk-level planner (chunked actor) predicts an open-loop action sequence of length h

$$\mathbf{u}_{t_0:t_0+h-1} = [u_{t_0,0}, \ldots, u_{t_0,h-1}] \in \mathbb{R}^{h \times d_a},$$

where $u_{t_0,k}$ denotes the action associated with the k-th step of the chunk. We use u for planned chunk actions and a for the action executed in the environment.

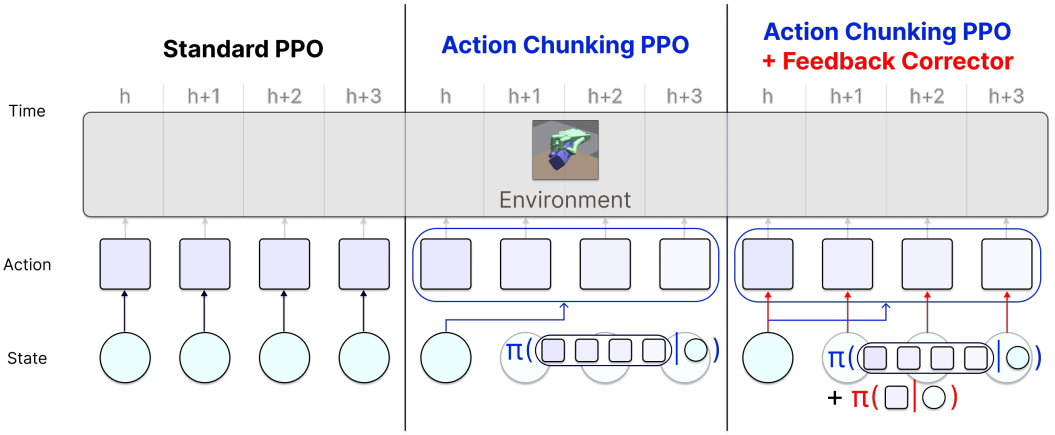


Figure 1: Stepwise comparison between PPO, ACPPO, and ACPPO-Corr. PPO runs the policy closed-loop. ACPPO runs the policy open-loop. ACPPO-Corr corrects the open-loop policy with a closed-loop feedback corrector. In ACPPO-Corr, the open-loop policy and the closed-loop corrector are jointly trained.

4 Method

We first introduce **Action Chunking PPO (ACPPO)**, which brings action chunking into PPO while keeping a standard state-value critic. This avoids chunked Q-functions and keeps the learning setup fully online and on-policy. We then extend ACPPO to **ACPPO-Corr** by adding a stepwise feedback corrector for tasks where within-chunk reactivity is important.

4.1 Action Chunking PPO (ACPPO)

For any step t, let

$$t_0 = \left\lfloor \frac{t}{h} \right\rfloor h, \qquad k = t - t_0 \in \{0, \dots, h-1\},$$

where t_0 is the start of the current chunk and k is the within-chunk index. At chunk boundaries, a chunked planner $\pi^{(h)}_{\theta_{\text{pl}}}$ predicts an h step action sequence

$$\mathbf{u}_{t_0:t_0+h-1} = \pi^{(h)}_{\theta_{\text{pl}}}(s_{t_0}) = [u_{t_0,0}, \dots, u_{t_0,h-1}] \in \mathbb{R}^{h \times d_a}.$$

At step $t = t_0 + k$, ACPPO uses the k-th planned action $u_{t_0,k}$ as the policy mean and samples the executed action a_t from a Gaussian. In this way, temporal abstraction is introduced at the actor side, while the critic remains state only. Accordingly, we also form the advantage and the importance sampling ratio at the chunk level. The critic is trained with the standard PPO value loss using stepwise GAE returns.

Chunked advantage. Instead of using separate advantages at each timestep, ACPPO calculates a single chunked advantage at the chunk-start [9]. We define the h-step chunked advantage as

$$A^{\pi}_{h}(s_{t_0}, \mathbf{a}_{t_0:t_0+h-1}) := \mathbb{E}\left[\sum_{j=0}^{h-1} \gamma^j r_{t_0+j} + \gamma^h V^{\pi}(s_{t_0+h}) - V^{\pi}(s_{t_0}) \,\middle|\, s_{t_0}, \mathbf{a}_{t_0:t_0+h-1} \right].$$

In practice, we estimate it as a combination of the h-step chunk head and a standard GAE tail at the next chunk boundary:

$$\hat{A}^{\text{chunk}}_{t_0} = \sum_{j=0}^{h-1} \gamma^j \delta_{t_0+j} + \gamma^h \hat{A}^{\text{GAE}}_{t_0+h}, \tag{1}$$

where $\delta_t := r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ is the stepwise residual and $\hat{A}^{\text{GAE}}_t$ is the standard stepwise GAE. Appendix C shows that the h-step head telescopes intermediate value terms with a bias–variance tradeoff as the chunk length increases.

Algorithm 1 ACPPO-Corr

Require: planner $\pi_{\theta_{\mathrm{pl}}}$, corrector/value network $\pi_{\theta_{\mathrm{co}},\phi}$, chunk length h, rollout horizon T

for $e = 1, \ldots, N$ **do**
  $\textsc{freeze}_{\mathrm{pl}} \leftarrow \mathbb{1}\{e \le N_{\mathrm{warm}}\}$
  for $t = 0, \ldots, T-1$ **do**
    if $t \bmod h = 0$ **then**
      $t_0 \leftarrow t, \quad \mathbf{u}_{t_0:t_0+h-1} \leftarrow \pi_{\theta_{\mathrm{pl}}}(s_{t_0})$ ▷ plan an action chunk
    end if
    $k \leftarrow t - t_0$
    $(c_t, \sigma_t, V_t) \leftarrow \pi_{\theta_{\mathrm{co}},\phi}(s_t)$ ▷ add closed-loop correction
    Sample $a_t \sim \mathcal{N}(u_{t_0,k} + c_t, \mathrm{diag}(\sigma_t^2))$
    Execute a_t and store $(s_t, s_{t_0}, a_t, r_t, d_t, \log \pi(a_t), V_t, k)$
  end for
  Compute stepwise GAE $\hat{A}_t^{\mathrm{GAE}}$ and chunked advantages $\hat{A}_{t_0}^{\mathrm{chunk}}$ using Eq. 1
  For mid-chunk terminations, mask the remaining chunk suffix
  for each PPO minibatch **do**
    if not $\textsc{freeze}_{\mathrm{pl}}$ **then**
      Evaluate $\mathcal{L}_{\mathrm{total}}$ (Eq. 3) with planner-side stop-grads: $u_{t_0,k} + \mathrm{sg}(c_t),\ \mathrm{sg}(\sigma_t)$
      Update θ_{pl} with Adam
    end if
    Evaluate $\mathcal{L}_{\mathrm{total}}$ using corrector-side stop-gradients: $\mathrm{sg}(u_{t_0,k}) + c_t,\ \sigma_t$
    Update $(\theta_{\mathrm{co}}, \phi)$ with Adam
  end for
end for

Projected state critic. When executing chunked actions, the exact value of a mid-chunk step may depend on active chunk variables z_t such as the chunk-start state, planned chunk, and within-chunk index. Let $\widetilde{V}^{\pi}(s_t, z_t)$ denote the exact augmented-state value. Under a squared value loss, the best state-only predictor is

$$V_*^{\pi}(s) = \mathbb{E}\Big[\widetilde{V}^{\pi}(s_t, z_t) \mid s_t = s\Big],$$

where the expectation is over chunk variables induced by the current policy. Our critic $V_\phi(s)$ approximates this projection. In Eq. 1, the h-step actor advantage uses only the boundary evaluations $V_\phi(s_{t_0})$ and $V_\phi(s_{t_0+h})$, with intermediate value terms telescoping as shown in Appendix C.

Chunked importance sampling ratio. ACPPO uses the importance ratio of the joint chunk distribution rather than per-step ratios. Let $\mathcal{T}_{t_0} \subset \{0, \ldots, h-1\}$ denote the valid chunk prefix [39] before any mid-chunk episode termination.

$$\rho_{t_0}^{(h)}(\theta) = \frac{\pi_\theta^{(h)}(\mathbf{a}_{t_0:t_0+h-1} \mid s_{t_0})}{\pi_{\theta_{\mathrm{old}}}^{(h)}(\mathbf{a}_{t_0:t_0+h-1} \mid s_{t_0})} = \prod_{k \in \mathcal{T}_{t_0}} \frac{\pi_{\theta,k}^{(h)}(a_{t_0+k} \mid s_{t_0})}{\pi_{\theta_{\mathrm{old}},k}^{(h)}(a_{t_0+k} \mid s_{t_0})}.$$

The clipped chunk-level surrogate becomes

$$J_{\mathrm{chunk}}(\theta) = \mathbb{E}_{t_0}\left[\min\Big(\rho_{t_0}^{(h)}(\theta)\hat{A}_{t_0}^{\mathrm{chunk}},\ \mathrm{clip}\Big(\rho_{t_0}^{(h)}(\theta),\, 1-\epsilon,\, 1+\epsilon\Big)\,\hat{A}_{t_0}^{\mathrm{chunk}}\Big)\right].$$

If the action at offset j terminates the episode, then $\mathcal{T}_{t_0} = \{0, \ldots, j\}$ and the remaining offsets are masked. In vectorized rollouts, a mid-chunk terminated environment is held as inactive padding until the next global chunk boundary, where a new planner is called.

4.2 ACPPO-Corr: ACPPO With a Stepwise Feedback Corrector

Adding the stepwise corrector. ACPPO addresses the first part of the problem: it brings action chunking into the PPO framework with a standard value critic, avoiding the need to train $Q(s_t, a_{t:t+h-1})$. However, ACPPO still executes the planned chunk open loop once the chunk has started. In dexterous manipulation and locomotion, this can be a limitation since contacts and object motion can change quickly [16, 33, 21]. In such settings, low-frequency planning can be useful, but low-frequency reacting is not always sufficient.

To recover within-chunk reactivity, we keep the ACPPO planner for chunk-level means and add a corrector $\pi_{\theta_{\mathrm{co}}}$ that observes the current state at every control step:

$$(c_t, \sigma_t) = \pi_{\theta_{\mathrm{co}}}(s_t).$$

Here, c_t is a stepwise correction to the planner mean, while σ_t is the stepwise exploration scale. The executed Gaussian policy becomes

$$\pi_\theta(a_t \mid s_t, s_{t_0}, k) = \mathcal{N}\big(u_{t_0,k} + c_t,\ \mathrm{diag}(\sigma_t^2)\big)\,,$$

where $\theta = (\theta_{\mathrm{pl}}, \theta_{\mathrm{co}})$. Here, the planner provides chunk-level structure, while the corrector provides stepwise feedback and stepwise exploration inside the chunk.

Corrected importance sampling ratio. ACPPO-Corr uses the same chunked advantage in Eq. 1, but the chunked importance sampling ratio is now also conditioned on the current state. For a valid chunk prefix $\mathcal{T}_{t_0}$ [39], the importance ratio is

$$\rho_{t_0}^{\mathrm{corr}}(\theta) = \prod_{k \in \mathcal{T}_{t_0}} \frac{\pi_\theta(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k)}{\pi_{\theta_{\mathrm{old}}}(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k)}. \tag{2}$$

In practice, the adaptive KL scheduling of PPO [23] and short chunk lengths keep the empirical clip fraction close to PPO. The full diagnostics are reported in Appendix D.

Decoupled planner and corrector updates. We decouple planner and corrector optimization with stop gradients. Let $\mathrm{sg}(\cdot)$ denote stop gradient. For each minibatch, we evaluate the chunk surrogate with two policy parameterizations:

$$\pi_\theta^m(a_t \mid s_t, s_{t_0}, k) = \begin{cases} \mathcal{N}\big(u_{t_0,k} + \mathrm{sg}(c_t),\ \mathrm{diag}(\mathrm{sg}(\sigma_t)^2)\big)\,, & m = \mathrm{pl}, \\ \mathcal{N}\big(\mathrm{sg}(u_{t_0,k}) + c_t,\ \mathrm{diag}(\sigma_t^2)\big)\,, & m = \mathrm{co}. \end{cases}$$

Using the chunk-level surrogate with $\rho_{t_0}^{\mathrm{corr}}$, we write a single training objective:

$$\mathcal{L}_{\mathrm{total}} = -J_{\mathrm{chunk}}^\theta + \underbrace{c_v \mathcal{L}_{\mathrm{value}} - c_e \mathcal{H}[\pi_\theta^{\mathrm{co}}] + \lambda_b \mathcal{L}_{\mathrm{bound}}}_{\text{Standard PPO Regularizers}} + \underbrace{\lambda_r \mathbb{E}_t\big[\|c_t\|_2^2\big]}_{\substack{\text{Corrector} \\ \text{Regularizer}}}. \tag{3}$$

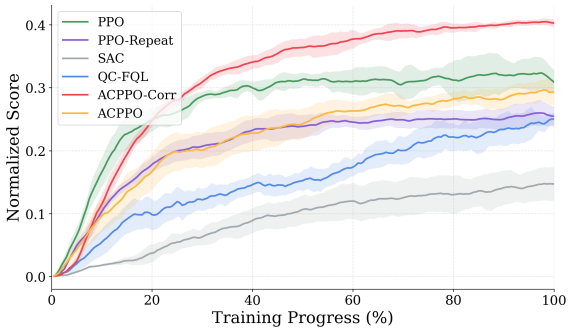
The critic loss is $\mathcal{L}_{\mathrm{value}} = \mathbb{E}_t[(V_\phi(s_t) - \hat{R}_t)^2]$, $\mathcal{H}$ denotes the policy entropy, $\mathcal{L}_{\mathrm{bound}}$ is the action-bound regularizer [24], and the corrector regularizer term penalizes large corrections. In the planner pass, only $u_{t_0,k}$ receives actor gradients; in the corrector/value pass, c_t, σ_t, and V_ϕ are updated, using the two Adam updates shown in Algorithm 1.

Value warm-up. In all experiments, we use a short warm-up stage before full joint training. Specifically, we freeze the chunk planner for the first $N/20$ epochs and continue optimizing the corrector/value branch; after warm-up, we unfreeze the planner and train all modules jointly with separate Adam optimizers. This allows the value function to create stable advantage estimates before the chunk-level planner begins to learn from it.

5 Experiments

Benchmark. We evaluate ACPPO-Corr on 25 simulated robotics tasks, comprising 9 tasks from IsaacGym [24] and 16 tasks from Bi-DexHands [25]. The benchmark spans locomotion, arm manipulation, and dexterous hand-object interaction with different horizons and contact dynamics. This diversity is particularly important in our setting, since the effect of action chunking and reduced decision frequency can vary across different tasks.

Algorithms. We compare ACPPO-Corr against five baselines: PPO [23], the stepwise on-policy reference; PPO-Repeat [40], PPO with repeated actions, which isolates the effect of reduced decision frequency; SAC [41], a standard off-policy actor-critic; QC-FQL [10], the action chunking RL comparator; and ACPPO, which removes the proposed feedback corrector. The chunk length and action repetition frequency are fixed to $h = 4$. In implementation, ACPPO-Corr uses a reduced-width planner/corrector design to keep model size comparable to PPO. Appendix A.2 reports the model sizes and the computational efficiency of each algorithm.



(a) Training Curve (25 Tasks)

Method	Sensitive	Neutral
PPO	1.00 [1.00, 1.00]	1.00 [1.00, 1.00]
PPO-Repeat	0.61 [0.55, 0.68]	0.97 [0.92, 1.02]
SAC	0.33 [0.21, 0.44]	0.59 [0.49, 0.68]
QC-FQL	0.76 [0.64, 0.87]	0.77 [0.66, 0.87]
ACPPO	0.92 [0.87, 0.97]	0.97 [0.85, 1.10]
ACPPO-Corr	**1.33 [1.24, 1.43]**	**1.24 [1.15, 1.34]**

(b) Frequency-Sensitivity Split Performance

Figure 2: Benchmark results. **(a)** Normalized IQM Scores over training across all 25 tasks. Shaded regions show pointwise 95% seed-bootstrap intervals with tasks fixed. ACPPO-Corr achieves best aggregated performance, improving over PPO by 30.4% in final normalized IQM. **(b)** Final relative IQM with 95% seed-bootstrap intervals with tasks fixed on the frequency-sensitive and frequency-neutral subsets, computed as the ratio to PPO's IQM on the same task set.

Hyperparameters. For PPO-family baselines and SAC, we use the published task-specific hyperparameters from prior benchmark implementations [42, 25]. ACPPO-Corr inherits the PPO hyperparameters and introduces the corrector regularization weight λ_r. We select $\lambda_r \in \{0.03, 0.1, 0.3\}$ using preliminary per-task tuning runs, then fix the selected value for the reported evaluation. To match the tuning budget, we tune the QC-FQL behavior regularization coefficient $\alpha \in \{1, 3, 10\}$ using the same protocol. Details are given in Appendix A.3, and full hyperparameters are given in Appendix A.7.

Metrics. Based on the task success indicator in IsaacGym and Bi-DexHands, we use two different metrics. For goal reaching tasks, where performance is defined by completion of the target behavior, we use the final success rate $(0 - 1)$. For progress-based control tasks, where performance is captured by the accumulated reward, we use episodic return. We then normalize each task score as $R_{\text{norm}} = (R - R_{\text{low}})/(R_{\text{high}} - R_{\text{low}})$, where R_{low} and R_{high} are the task-specific lower and upper bound values reported at Appendix A.7. Each method is evaluated with 5 random seeds on every task, and we report the interquartile mean (IQM) [43] across all tasks. For the final relative IQM, we compute the ratio between a method's final IQM and PPO's final IQM on the same task set.

5.1 Benchmark Results

Overall Performance. Figure 2a summarizes the main benchmark results. ACPPO-Corr achieves best aggregated performance, improving over PPO by 30.4% in final normalized IQM. Both ACPPO and PPO-Repeat underperform PPO on the full benchmark, indicating that the gains of ACPPO-Corr cannot be explained by reduced decision frequency alone. ACPPO-Corr also substantially outperforms QC-FQL, our main chunked-action RL baseline, suggesting that actor-side chunking with a standard state-value critic is effective when paired with stepwise feedback correction in this online PPO setting. Soft Actor-Critic [41] performs poorly overall, consistent with prior reports on Bi-DexHands [25]; this reflects the difficulty of off-policy Q-learning in high-dimensional, contact-rich continuous action spaces. The full learning curves are shown in Appendix A.6, and the visualizations are displayed in Figure 3.

Frequency Sensitivity. Prior work [40, 44] has shown that performance can depend strongly on interaction frequency and action repetition. Therefore, we partition tasks according to their sensitivity to reduced decision frequency. For each task e, we compute

$$\Delta_e^{\text{freq}} = \left| \frac{S_e^{\text{PPO}} - S_e^{\text{PPO-Repeat}}}{S_e^{\text{PPO}}} \right|,$$

where S_e denotes the final normalized score (without IQM) on task e. A larger Δ_e^{freq} indicates that the task is more sensitive to changes in decision frequency. We exclude four tasks with near-

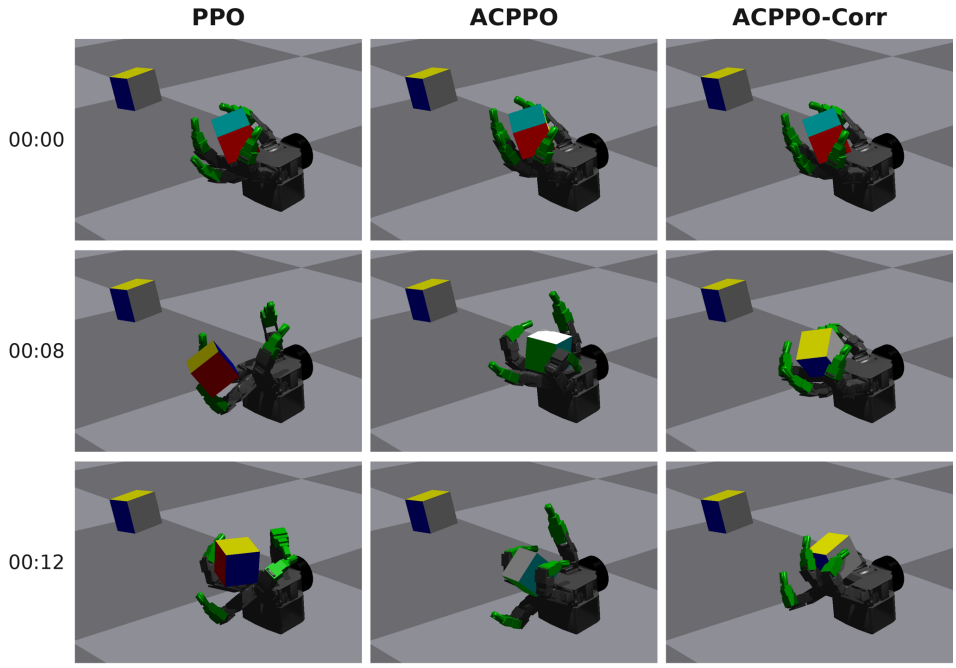
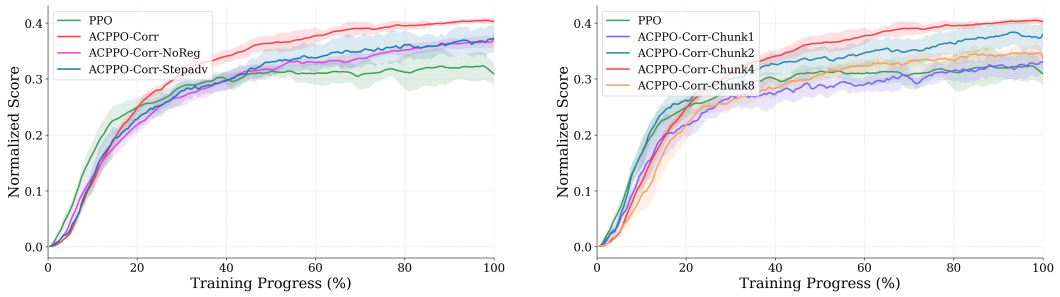


Figure 3: Rollout snapshots of PPO, ACPPO, and ACPPO-Corr on `AllegroHand`. ACPPO-Corr reaches the target orientation the fastest, where ACPPO drops the cube and PPO requires additional 128 environment interactions to reach the target.



(a) Advantage estimate and corrector regularization

(b) Sensitivity to chunk length

Figure 4: Ablations on chunked advantage, corrector regularization, and chunk length. (a) Chunked GAE and corrector regularization improve both performance and stability. (b) $h = 1$ removes multi-step chunking while retaining the ACPPO-Corr pipeline. Moderate chunk lengths perform best, with $h = 4$ giving the strongest aggregate performance.

zero PPO scores to ensure numerical stability of this ratio. Using a threshold of $\Delta_e^{\text{freq}} > 0.3$, we split the remaining 21 tasks into 11 frequency-sensitive tasks and 10 frequency-neutral tasks. The frequency-sensitivity split results are elaborated in Appendix A.1.

Figure 2b reports the final relative IQM under this split, computed as the ratio of IQMs within each subset. On frequency-neutral tasks, PPO and PPO-Repeat perform similarly, as expected when reduced decision frequency has limited effect. On frequency-sensitive tasks, PPO-Repeat suffers a large performance drop, confirming that naive action repetition can be harmful when tasks require frequent feedback. Using the win/tie/loss defined in Appendix A.1, ACPPO-Corr is within the top group on 10/10 frequency-neutral tasks and 9/11 frequency-sensitive tasks. This suggests that ACPPO-Corr preserves the benefits of chunk-level planning while avoiding the performance degradation caused by open-loop execution: the stepwise corrector provides within-chunk reactivity on tasks where reduced decision frequency is harmful.

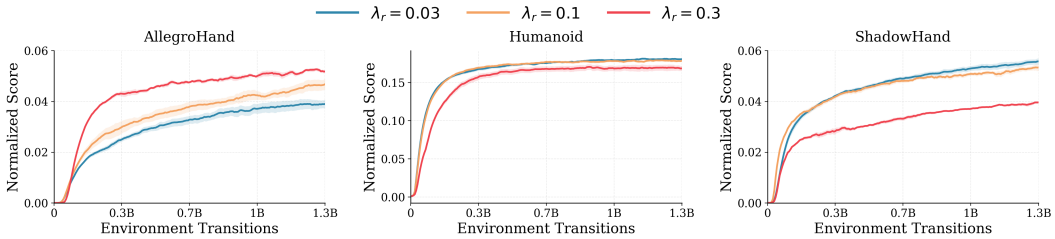


Figure 5: Sensitivity to the corrector regularizer λ_r on three representative tasks. Different tasks prefer different regularization strengths: λ_r is a relevant method-specific hyperparameter.

5.2 Ablation Study

Effect of the corrector regularizer. The corrector regularizer in Eq. 3 prevents the corrector from dominating the action plans. Without this term, the corrector can absorb most of the action signal, causing the chunked planner to contribute little and making ACPPO-Corr behave like a mostly stepwise reactive policy. ACPPO-Corr-NoReg of Figure 4a compares ACPPO-Corr with and without the corrector regularizer. The regularized variant achieves stronger aggregate performance, showing that balancing the planner and corrector is crucial for effective chunked control.

Effect of within-chunk feedback. To ablate the observations received during chunk execution, we replace the corrector's current-state input s_t with the chunk-start state s_{t_0}, keeping the architecture unchanged. Applying this change only at evaluation retains 84.0% of ACPPO-Corr's performance, and training from scratch retains 88.1% of ACPPO-Corr's performance. Retraining partially recovers performance, but the remaining gap supports the benefit of within-chunk feedback.

Effect of the chunked advantage. We next ablate the chunked advantage used to train the ACPPO-Corr policy. We compare ACPPO-Corr, which uses Eq. 1 as the advantage estimate against a variant that replaces the chunked advantage with standard stepwise GAE. As shown in Figure 4a, ACPPO-Corr achieves stronger aggregate performance than ACPPO-Corr-Stepadv. This suggests that the policy-gradient signal should match the temporal abstraction of the actor: because the planner selects an entire action chunk, assigning a single temporally extended advantage to the chunk decision provides more coherent credit assignment than using separate stepwise advantages. This result is also consistent with the analysis in Appendix C, where the chunked estimator reduces reliance on intermediate value estimates, but uses a coarser and potentially higher-variance chunk-level credit signal.

Sensitivity to the chunk length. To study the effect of the chunk length h, we evaluate chunk lengths $h \in \{1, 2, 4, 8\}$ and compare it to the performance of PPO. Since the rollout horizon length is 8 for most environments in the task set, we restrict the comparison to chunk lengths that divide this horizon. The $h = 1$ setting removes chunking while retaining the ACPPO-Corr pipeline: it analyzes whether the gains are explained by decomposing the policy into two branches. Figure 4b shows that $h = 1$ improves only marginally over PPO and remains below the chunked variants. Performance improves substantially for $h > 1$, with $h = 4$ achieving the strongest aggregate performance. This suggests that the main gains of ACPPO-Corr come from combining chunk-level planning with stepwise feedback correction, rather than from the corrector alone.

Sensitivity to the corrector regularizer hyperparameter. To study the effect of the corrector regularization coefficient λ_r, we select three representative tasks and sweep $\lambda_r \in \{0.03, 0.1, 0.3\}$. Figure 5 shows that the best-performing value is task-dependent: stronger regularization works better on AllegroHand, while weaker regularization performs better on Humanoid and ShadowHand. This sweep shows that the appropriate strength of the corrector regularizer varies across tasks.

To visualize how λ_r changes the division between the planner and the corrector, we also report the final correction ratio $r_{\text{corr}} = \|c_{t_0+k}\|_2 / \|u_{t_0,k}\|_2$. Table 1 shows that larger λ_r consistently reduces r_{corr}, confirming that the regularizer prevents the feedback corrector from dominating the chunk planner.

Table 1: Final correction ratios across tasks and regularization strengths λ_r.

Task	$\lambda_r = 0.03$	$\lambda_r = 0.1$	$\lambda_r = 0.3$
AllegroHand	0.222	0.053	0.007
Humanoid	0.021	0.009	0.002
ShadowHand	0.163	0.029	0.009

Table 2: Longer-horizon evaluation. Entries are the means of per-task score ratios to PPO.

Method	$h = 4$	$h = 8$	$h = 16$
ACPPO-Corr	1.39	1.32	1.21
ACPPO	1.14	1.07	0.68
QC-FQL	0.88	0.87	0.29

Longer chunk horizons. We extend the chunk-length evaluation to $h \in \{4, 8, 16\}$ on the five tasks whose rollout horizons support these lengths. As shown in Table 2, ACPPO-Corr degrades less as h increases, retaining a mean score ratio to PPO of 1.21 at $h = 16$, compared with 0.68 for ACPPO and 0.29 for QC-FQL. We additionally evaluate $h = 32$ on FrankaCubeStack and Humanoid in Table 8. ACPPO-Corr achieves nontrivial scores, while ACPPO and QC-FQL each obtain at most 0.02 on either task. These results support the benefit of feedback correction over longer execution windows, although moderate chunk lengths remain preferable.

6 Conclusion & Limitations

We presented ACPPO, an action-chunking variant of PPO with a standard state-value critic, and ACPPO-Corr, which augments chunk-level planning with a stepwise feedback corrector. This design brings temporal abstraction into a fully online, on-policy PPO framework while providing within-chunk reactivity which is important in contact-rich robotic control. Across 25 simulated robotics tasks spanning IsaacGym and Bi-DexHands, ACPPO-Corr achieved the strongest overall benchmark performance and remained best on both frequency-sensitive and frequency-neutral task subsets, suggesting the impact of action chunking and closed-loop action corrections. The open-loop ACPPO baseline underperforms PPO on the full benchmark, indicating that actor-side chunking alone is not sufficient for these contact-rich tasks. Our ablations further indicate that moderate chunk lengths are most effective and that regularizing the corrector helps balance the division between the chunk planner and the feedback corrector. Overall, these results suggest that action chunking can be made effective in PPO when open-loop temporal structure is paired with closed-loop correction.

However, several limitations remain. First, our study is limited to continuous robotics control tasks: ACPPO-Corr assumes a continuous action space where additive corrections and magnitude penalties are meaningful, and therefore does not directly apply to discrete or semantically discontinuous action spaces such as token-level NLP. Furthermore, our experiments are conducted only in simulated robotics environments with state-based observations. Finally, the chunked importance sampling ratio may become less stable as the chunk length increases. While clipping remained stable for short chunk lengths, increased clipping was observed in $h = 8$. Applying ACPPO-Corr to substantially longer chunks may require additional stabilization. Important directions for future work include adaptive chunk lengths, sim-to-real transfer, and extensions to latent-action formulations for application beyond continuous control.

Acknowledgements

This work was partly supported by the IITP grants (RS-2022-II220113, RS-2022-II220959, RS-2022-II220871, RS-2026-25507282, RS-2021-II211343 (SNU AI), RS-2025-25442338 (AI Star Fellowship-SNU)) funded by the Korea government (MSIT), grants (RS-2025-25462891 (US-KOR BARI)) funded by MOTIR and the BK21 FOUR program, SNU in 2025.

References

[1] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[2] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4788–4795. IEEE, 2024.

[3] 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.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *Robotics: Science and Systems*, 2025.

[5] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *Robotics: Science and Systems*, 2025.

[6] Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Boyang Li, Shuo Liu, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. In *Workshop on Making Sense of Data in Robotics: Composition, Curation, and Interpretability at Scale at CoRL 2025*, 2025.

[7] Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. In *The Thirty Eighth Annual Conference on Learning Theory*, pages 5248–5351. PMLR, 2025.

[8] Thomas T Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and exploratory data collection yield exponential improvements in behavior cloning for continuous control. *arXiv preprint arXiv:2507.09061*, 2025.

[9] 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): 181–211, 1999.

[10] Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=XUks1Y96NR`.

[11] Si-Cheng Wang, Tian-Yu Xiang, Xiao-Hu Zhou, Mei-Jiang Gui, Xiao-Liang Xie, Shi-Qi Liu, Shuang-Yi Wang, Ao-Qun Jin, and Zeng-Guang Hou. Vla model post-training via action-chunked ppo and self behavior cloning. *arXiv preprint arXiv:2509.25718*, 2025.

[12] Kwanyoung Park, Seohong Park, Youngwoon Lee, and Sergey Levine. Scalable offline model-based RL with action chunks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=WXGb9unEHo`.

[13] 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. *arXiv preprint arXiv:2508.11143*, 2025.

[14] 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.

[15] Changyeon Kim, Haeone Lee, Younggyo Seo, Kimin Lee, and Yuke Zhu. DEAS: DEtached value learning with action sequence for scalable offline RL. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=bVTaAXeBmE`.

[16] Qiyang Li, Seohong Park, and Sergey Levine. Decoupled q-chunking. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aqGNdZQL9l.

[17] Younggyo Seo and Pieter Abbeel. Coarse-to-fine q-network with action sequence for data-efficient reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VoFXUNc9Zh.

[18] Ge Li, Dong Tian, Hongyi Zhou, Xinkai Jiang, Rudolf Lioutikov, and Gerhard Neumann. TOP-ERL: Transformer-based off-policy episodic reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=N4NhVN30ph.

[19] Dong Tian, Onur Celik, and Gerhard Neumann. Chunking the critic: A transformer-based soft actor-critic with n-step returns. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rb5eTktqbc.

[20] Roland S Johansson and J Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, 2009.

[21] Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[22] Yuhang Zheng, Songen Gu, Weize Li, Yupeng Zheng, Yujie Zang, Shuai Tian, Xiang Li, Ce Hao, Chen Gao, Si Liu, Haoran Li, Yilun Chen, Shuicheng Yan, and Wenchao Ding. Omnivta: Visuo-tactile world modeling for contact-rich robotic manipulation, 2026. URL https://arxiv.org/abs/2603.19201.

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

[24] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[25] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

[26] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[27] Gwanwoo Song, Kwanyoung Park, and Youngwoon Lee. Chunk-guided q-learning. *arXiv preprint arXiv:2603.13971*, 2026.

[28] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pages 91–100. PMLR, 2022.

[29] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems XIV*, 2018.

[30] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[31] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–08. IEEE, 2025.

[32] Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025.

[33] Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[34] Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

[35] Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks. *arXiv preprint arXiv:2509.23224*, 2025.

[36] Haoxuan Wang, Gengyu Zhang, Yan Yan, Yuzhang Shang, Ramana Rao Kompella, and Gaowen Liu. Real-time robot execution with masked action chunking. *arXiv preprint arXiv:2601.20130*, 2026.

[37] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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

[39] Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025.

[40] Sahil Sharma, Aravind S Lakshminarayanan, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *International Conference on Learning Representations*, 2017.

[41] 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*, pages 1861–1870. Pmlr, 2018.

[42] Chi-Chang Lee, Zhang-Wei Hong, and Pulkit Agrawal. Going beyond heuristics by imposing policy improvement as a constraint. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=vBGMbFgvsX`.

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

[44] Amirmohammad Karimi, Jun Jin, Jun Luo, A Rupam Mahmood, Martin Jagersand, and Samuele Tosatto. Dynamic decision frequency with continuous options. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7545–7552. IEEE, 2023.

A Experiment Details

A.1 Benchmarks

We use the composition of 9 tasks from IsaacGym [24], and 16 tasks from Bi-DexHands [25]. For the frequency-sensitivity analysis only, we exclude four tasks whose PPO scores are near zero, since the relative frequency-sensitivity ratio Δ_e^{freq} becomes numerically unstable. The total task sets and the frequency-sensitivity splits are displayed in Table 3.

We also report a per-task win/tie/loss status for ACPPO-Corr. For each task e, let $S_{e,m}$ denote the final normalized score of method m, averaged over five seeds. We compare ACPPO-Corr against the strongest baseline method on the same task:

$$\Delta_e^{\text{WTL}} = S_{e,\text{ACPPO-Corr}} - \max_{m \neq \text{ACPPO-Corr}} S_{e,m}.$$

We mark a task as a win if $\Delta_e^{\text{WTL}} > 0.03$, a tie if $|\Delta_e^{\text{WTL}}| \leq 0.03$, and a loss if $\Delta_e^{\text{WTL}} < -0.03$.

Table 3: Task assignment for the frequency-sensitivity split and per-task win/tie/loss status of ACPPO-Corr. Superscripts denote ACPPO-Corr's status against the strongest competing method on each task: W = win, T = tie, L = loss.

Group	Tasks	#	W/T/L
Neutral	FrankaCabinet^W, Ingenuity^T, ShadowHandDoorOpenInward^W, ShadowHandDoorCloseInward^T, ShadowHandDoorOpenOutward^T, ShadowHandScissors^W, QuadCopter^T, ShadowHandGraspAndPlace^W, ShadowHandOver^T, ShadowHandBlockStack^W	10	5/5/0
Sensitive	ShadowHandSpin^W, ShadowHandBottleCap^L, Humanoid^W, Anymal^T, ShadowHand^W, AllegroHand^W, Ant^W, ShadowHandUpsideDown^L, ShadowHandCatchOver2Underarm^W, ShadowHandCatchAbreast^W, FrankaCubeStack^T	11	7/2/2
Excluded	ShadowHandLiftUnderarm^W, ShadowHandDoorCloseOutward^L, ShadowHandKettle^L, ShadowHandPushBlock^T	4	1/1/2
All tasks	–	25	13/8/4

A.2 Model Size and Computational Overhead

The MLP widths are task-dependent and follow the standard configurations used in Lee et al. [42]. For PPO and PPO-Repeat, we inherit the default task-specific PPO actor and state-value critic architectures. ACPPO keeps the same PPO architecture, except that the actor output dimension is expanded from d_a to hd_a to predict an action chunk.

For methods with multiple actor or critic branches, we use reduced-width MLPs, with the first layer of the MLP divided in half to keep the total model size comparable. Table 4 summarizes the network components used by each method. Table 5 reports representative parameter counts and training throughput under our benchmark implementation. All experiments were conducted on a single RTX A6000 GPU.

Table 4: Network components used per algorithm. Halved MLP configuration is used for methods with multiple actor or critic branches.

Algorithm	**Network components**
PPO	1 full-size actor + 1 full-size value critic
ACPPO	1 full-size chunked actor + 1 full-size value critic
ACPPO-Corr	1 chunk planner + 1 corrector + 1 full-size value critic
SAC	1 actor + 2 Q-functions + 2 target Q-functions
QC-FQL	1 chunked actor + 2 Q-functions + 2 target Q-functions + 1 flow network

Table 5: Training throughput and model size. Steps/sec includes rollout and optimization. Parameter counts include policy/value networks and, for off-policy methods, target critic networks used during training.

Algorithm	Steps/sec	Parameters
PPO	47,894	1,463,785
ACPPO	45,579	1,484,065
ACPPO-Corr	41,714	1,557,258
SAC	39,049	2,024,428
QC-FQL	34,595	3,287,492

A.3 Hyperparameter Selection

Unless otherwise stated, all chunked methods use chunk length $h = 4$ in the main benchmark. The effect of varying h is studied separately in Section 5.2.

For PPO and SAC, we inherit the task-specific hyperparameters from Lee et al. [42] and Chen et al. [25]. PPO-Repeat and ACPPO use the same hyperparameters and network architecture as PPO.

ACPPO-Corr introduces one additional hyperparameter, the corrector regularization coefficient λ_r. For each environment, we select λ_r from $\{0.03, 0.1, 0.3\}$ using preliminary tuning runs, then use the selected value for the reported five seed runs.

For QC-FQL, we use the same three-value tuning budget and select the behavior regularization coefficient from $\alpha \in \{1, 3, 10\}$ following the same procedure as ACPPO-Corr. We find that $\alpha = 1$ performs best in most environments. This is expected in our fully online setting: early rollouts are noisy and non-expert, so strong behavior cloning to the replay buffer can constrain policy improvement. Reducing the behavior regularization allows the policy to deviate from early low-quality behavior while still retaining the stabilizing effect of the regularizer. The selected coefficient for each environment was chosen using preliminary tuning runs and then fixed for the reported five-seed evaluation. The selected values per environment for both algorithms are reported in Appendix A.7.

A.4 Action Chunking RL Baselines

Several related chunk-correction and real-time execution methods are not included as direct baselines because they assume a different training interface from our benchmark. In particular, many methods operate on frozen behavior-cloned chunked policies, require demonstrations or pretrained VLA/diffusion policies, use visual or tactile observations not available in our state-based benchmark, or modify test-time chunk resampling rather than online RL training from scratch. Including such methods would require adding offline data, pretrained policies, or different observation modalities, which would break the online state-based comparison. We therefore focus the main benchmark on methods that can be trained from scratch under a matched online protocol, and include QC-FQL as the closest chunked-action RL comparator.

A.5 Aggregation

For each method, task, and seed, we first normalize the evaluation curve using the task-specific bounds in Appendix A.7, smooth the curve, and interpolate it onto a common training-progress grid. If multiple runs share the same method-task-seed tuple, we average them before aggregation. Let $x_{m,e,s}(p)$ denote the resulting normalized score for method m, task e, seed s, and training progress p.

At each progress point p, we aggregate performance using the interquartile mean (IQM) [43] over the set of task-seed scores

$$\mathcal{X}_m(p) = \{x_{m,e,s}(p) : e \in \mathcal{E},\ s \in \mathcal{S}\}.$$

The solid learning curve reports

$$\mathrm{IQM}_m(p) = \mathrm{IQM}(\mathcal{X}_m(p)),$$

where IQM is the mean of the middle 50% of scores after sorting.

Uncertainty intervals are computed by seed bootstrap with the task set fixed. For each bootstrap replicate, we resample the five seed slots with replacement, recompute $\mathrm{IQM}_m(p)$ at every progress point using the resampled task-seed scores, and report the pointwise 2.5 and 97.5 percentiles. Thus, the shaded regions reflect variability due to random seeds under the fixed benchmark suite, not uncertainty over a broader distribution of tasks.

Final performance is computed analogously from the last finite value of each normalized task-seed curve. For PPO-normalized performance on a task subset $\mathcal{E}'$, we recompute the method IQM and PPO IQM under the same bootstrap seed resample and report

$$\mathrm{RelIQM}_m = \frac{\mathrm{IQM}_m}{\mathrm{IQM}_{\mathrm{PPO}}}.$$

Bootstrap intervals for relative IQM are obtained from the corresponding bootstrap distribution of this ratio.

A.6 Full Training Curve

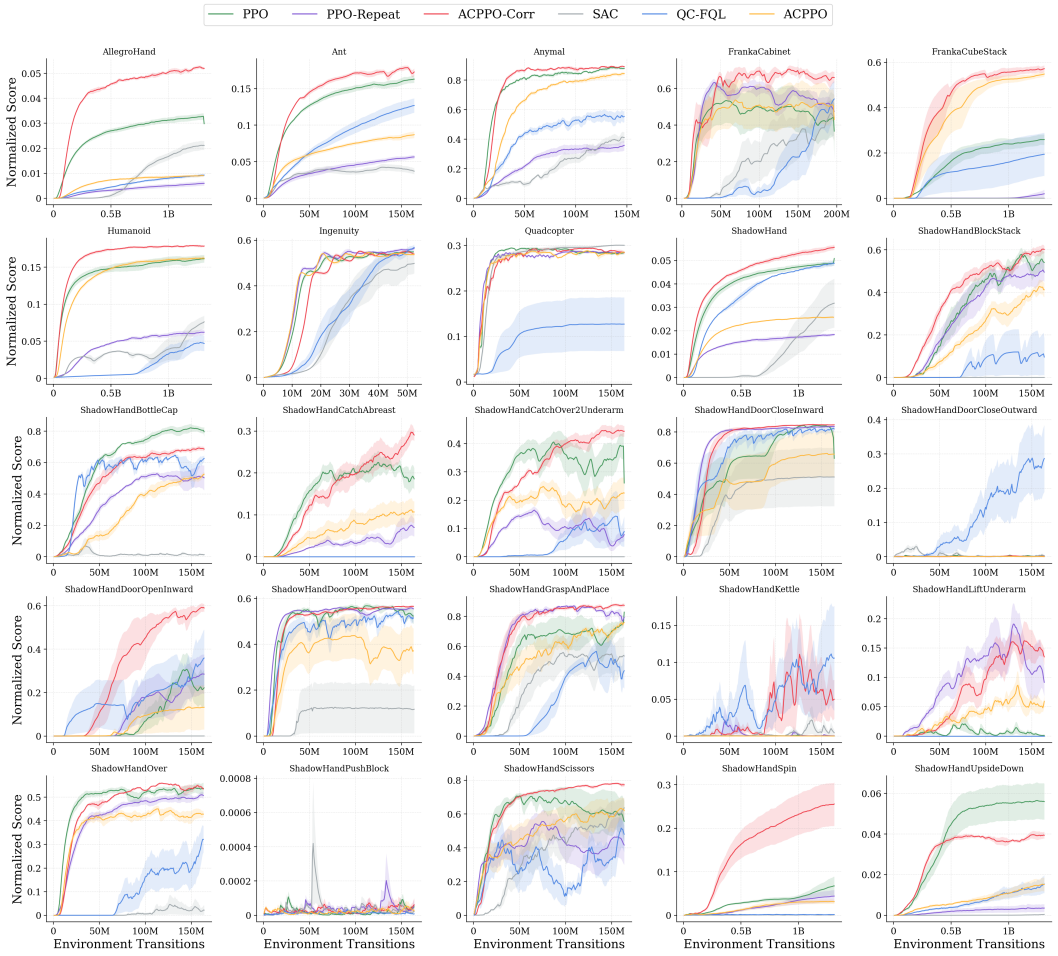


Figure 6: Full training curves on IsaacGym and Bi-DexHands.

A.7 Per Task Hyperparameters

Table 6: Per-task selected hyperparameters and normalization bounds. λ_r denotes the corrector regularization coefficient for ACPPO-Corr at each chunk length, α denotes the behavior regularization coefficient for QC-FQL, and $R_{\text{low}}, R_{\text{high}}$ are the task-specific bounds used in normalized IQM computation.

Task	λ_r $(h=1)$	λ_r $(h=2)$	λ_r $(h=4)$	λ_r $(h=8)$	QC-FQL α	R_{low}	R_{high}
`FrankaCabinet`	0.1	0.1	0.1	0.1	3	0	1
`ShadowHandDoorOpenInward`	0.1	0.3	0.1	0.1	1	0	1
`ShadowHandDoorCloseInward`	0.3	0.1	0.1	0.1	1	0	1
`ShadowHandDoorOpenOutward`	0.3	0.3	0.3	0.3	1	0	1
`ShadowHandScissors`	0.3	0.3	0.3	0.3	1	0	1
`Ingenuity`	0.1	0.1	0.1	0.1	1	0	14000
`QuadCopter`	0.3	0.3	0.3	0.3	1	0	1500
`ShadowHandGraspAndPlace`	0.1	0.1	0.1	0.1	1	0	1
`ShadowHandOver`	0.1	0.1	0.1	0.03	1	0	1
`ShadowHandBlockStack`	0.3	0.3	0.3	0.1	1	0	1
`ShadowHandSpin`	0.3	0.3	0.3	0.3	1	0	1
`ShadowHandBottleCap`	0.03	0.03	0.03	0.03	1	0	1
`Anymal`	0.3	0.3	0.3	0.3	1	0	75
`ShadowHand`	0.03	0.03	0.03	0.03	1	0	1
`Humanoid`	0.1	0.1	0.1	0.1	10	-1	62841
`Ant`	0.3	0.3	0.3	0.3	1	-2	61341
`ShadowHandUpsideDown`	0.1	0.03	0.03	0.03	1	0	1
`ShadowHandCatchOver2Underarm`	0.3	0.3	0.3	0.3	1	0	1
`AllegroHand`	0.1	0.1	0.3	0.1	1	0	1
`FrankaCubeStack`	0.3	0.3	0.3	0.3	1	0	1
`ShadowHandCatchAbreast`	0.03	0.03	0.03	0.03	1	0	1
`ShadowHandLiftUnderarm`	0.1	0.1	0.3	0.3	1	0	1
`ShadowHandKettle`	0.3	0.3	0.3	0.3	1	0	1
`ShadowHandPushBlock`	0.3	0.3	0.3	0.3	1	0	1
`ShadowHandDoorCloseOutward`	0.3	0.3	0.3	0.3	1	0	1

A.8 Simulation and Control Frequencies

Table 7: Simulation and control timescales. The ShadowHand row includes all ShadowHand-prefixed tasks.

Task	Physics (Hz)	Action repeat	Control (Hz)	$h = 4$ (s)
AllegroHand	120	4	30	0.133
Ant	120	2	60	0.067
Anymal	100	2	50	0.080
FrankaCabinet	60	1	60	0.067
FrankaCubeStack	120	2	60	0.067
Humanoid	120	2	60	0.067
Ingenuity	200	2	100	0.040
QuadCopter	200	2	100	0.040
ShadowHand tasks	120	2	60	0.067

A.9 Longer-horizon evaluation.

Ingenuity, FrankaCabinet, and Ant use rollout horizon $T = 16$; FrankaCubeStack and Humanoid use $T = 32$. All five tasks are evaluated at $h \in \{4, 8, 16\}$, and the latter two also support $h = 32$. Table 8 gives the per-task score ratios to PPO.

Table 8: Per-task score ratios to PPO at longer chunk lengths. A dash denotes an unevaluated setting due to the rollout horizon.

Method	Task	$h = 4$	$h = 8$	$h = 16$	$h = 32$
ACPPO-Corr	FrankaCubeStack	2.21	2.17	1.72	1.09
	Humanoid	1.11	1.07	0.97	0.65
	Ingenuity	0.99	1.00	0.99	–
	FrankaCabinet	1.56	1.39	1.48	–
	Ant	1.06	0.97	0.88	–
ACPPO	FrankaCubeStack	2.11	1.90	1.37	0.00
	Humanoid	1.00	0.71	0.52	0.02
	Ingenuity	0.99	0.57	0.37	–
	FrankaCabinet	1.08	1.48	0.72	–
	Ant	0.53	0.68	0.43	–
QC-FQL	FrankaCubeStack	0.75	1.59	1.12	0.00
	Humanoid	0.29	0.53	0.20	0.01
	Ingenuity	1.06	0.78	0.01	–
	FrankaCabinet	1.53	0.98	0.00	–
	Ant	0.78	0.48	0.10	–

A.10 Task dependence and failures.

The benefit of ACPPO-Corr varies with the need for temporally coherent actions and within-chunk feedback. On `AllegroHand` and `ShadowHandSpin`, ACPPO-Corr achieves 1.45 and 3.76 times PPO's score, respectively. Both tasks require coherent multi-step in-hand manipulation while continuously adapting to object slip and changing finger contacts.

On `FrankaCabinet` and `FrankaCubeStack`, open-loop ACPPO already performs well: their lower-dimensional manipulation and relatively stable parallel-gripper contacts reduce the need for within-chunk correction.

ACPPO-Corr underperforms PPO on `ShadowHandUpsideDown` and `ShadowHandBottleCap`, achieving 0.706 and 0.829 times PPO's score. We hypothesize that failures in these tasks require abandoning the current manipulation plan. A slipping pen may require an immediate reconfiguration of all fingers, and a failed cap grasp may require releasing and establishing a new contact pose. However, since ACPPO-Corr generates actions by correcting the planned sequence, it may remain biased toward the current manipulation mode and therefore underperform PPO.

For `ShadowHandPushBlock`, all evaluated methods obtained zero success, consistent with the behavior observed in the original benchmark [42]. We suspect that the existing reward signal and exploration budget provide insufficient guidance for discovering the coordinated two-hand pushing behavior. However, because none of the evaluated methods solves the task, we cannot conclusively attribute the failure to a particular component of the algorithms.

For `ShadowHandDoorCloseOutward`, the case is different: QC-FQL shows nonzero performance, whereas the other evaluated methods remain near zero. One possibility is that QC-FQL's flow-based actor better represents useful chunk distributions, or that its off-policy replay retains rare successful trajectories more effectively. However, QC-FQL differs in its policy class, replay-based data usage, Q-function optimization, and objective, making it difficult to explain it as a single factor.

B Closed-loop Chunked Importance Ratio

Closed-loop chunk likelihood. Although the planner is evaluated only at chunk boundaries, ACPPO-Corr defines a valid stochastic policy at every control step. For a chunk starting at t_0, let $k = t - t_0$ and let $\mathcal{T}_{t_0} \subseteq \{0, \dots, h-1\}$ denote the valid set of executed offsets before any mid-chunk termination. The planner outputs the chunk mean sequence

$$\mathbf{u}_{\theta,t_0} = \left[u_{\theta,t_0,0}, \dots, u_{\theta,t_0,h-1}\right] = \pi_\theta^{\mathrm{pl}}(s_{t_0}),$$

and the corrector outputs a state-dependent correction and exploration scale

$$(c_\theta(s_t), \sigma_\theta(s_t)) = \pi_\theta^{\mathrm{co}}(s_t).$$

The executed policy inside the chunk is therefore

$$\pi_\theta(a_t \mid s_t, s_{t_0}, k) = \mathcal{N}\big(a_t; u_{\theta,t_0,k} + c_\theta(s_t), \mathrm{diag}(\sigma_\theta(s_t)^2)\big)\,.$$

For a realized chunk prefix, the probability density of the sampled segment under π_θ is

$$p_\theta(\tau_{t_0} \mid s_{t_0}) = \prod_{k \in \mathcal{T}_{t_0}} \pi_\theta(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k) P(s_{t_0+k+1} \mid s_{t_0+k}, a_{t_0+k}), \tag{4}$$

where the transition kernel P is independent of the policy parameters. Hence, for the same sampled trajectory segment, the likelihood ratio between the current policy and the behavior policy is

$$\frac{p_\theta(\tau_{t_0} \mid s_{t_0})}{p_{\theta_{\mathrm{old}}}(\tau_{t_0} \mid s_{t_0})} = \prod_{k \in \mathcal{T}_{t_0}} \frac{\pi_\theta(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k)}{\pi_{\theta_{\mathrm{old}}}(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k)}.$$

Therefore, the product ratio in Eq. 2 is the correct likelihood ratio for the sampled closed-loop chunk, even though the corrector observes the current state inside the chunk. If the episode terminates inside a chunk, the product is taken only over the executed prefix $\mathcal{T}_{t_0}$ and the remaining offsets are masked.

Unclipped on-policy interpretation. For a chunk-start state s_{t_0}, define the h-step bootstrapped target

$$G_{t_0}^{(h)} = \sum_{k=0}^{h-1} \gamma^k r_{t_0+k} + \gamma^h V^\pi(s_{t_0+h}),$$

and the corresponding chunk-level advantage

$$A_h^\pi(s_{t_0}, \tau_{t_0}) = G_{t_0}^{(h)} - V^\pi(s_{t_0}).$$

Since the transition terms in Eq. 4 do not depend on θ, the score of the sampled chunk prefix is

$$\nabla_\theta \log p_\theta(\tau_{t_0} \mid s_{t_0}) = \sum_{k \in \mathcal{T}_{t_0}} \nabla_\theta \log \pi_\theta(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k).$$

Therefore, in the unclipped on-policy actor update, and treating the value target as fixed as in standard actor–critic methods, the chunk-level likelihood-ratio contribution takes the form

$$\left(\sum_{k \in \mathcal{T}_{t_0}} \nabla_\theta \log \pi_\theta(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k) \right) A_h^\pi(s_{t_0}, \tau_{t_0}).$$

C Chunked Advantage: Value-Error Cancellation and Tradeoff

Setup. Let V_*^π denote the projected state value from Section 4.1. For notational simplicity, we write it as V^π in this appendix, and define the critic approximation error $e(s) := V_\phi(s) - V^\pi(s)$. Let

$$\delta_t := r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \qquad \tilde{\delta}_t := r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t).$$

Then

$$\delta_t - \tilde{\delta}_t = \gamma e(s_{t+1}) - e(s_t). \tag{5}$$

Stepwise GAE. Let $\alpha := \gamma\lambda$. Standard stepwise GAE and its oracle counterpart are

$$\hat{A}_t^{\mathrm{GAE}} := \sum_{j=0}^{\infty} \alpha^j \delta_{t+j}, \qquad \tilde{A}_t^{\mathrm{GAE}} := \sum_{j=0}^{\infty} \alpha^j \tilde{\delta}_{t+j}.$$

Using Eq. 5, their difference is

$$\begin{aligned} \hat{A}_t^{\mathrm{GAE}} - \tilde{A}_t^{\mathrm{GAE}} &= \sum_{j=0}^{\infty} \alpha^j \left(\gamma e(s_{t+j+1}) - e(s_{t+j})\right) \\ &= -e(s_t) + \gamma(1-\lambda) \sum_{j=1}^{\infty} (\gamma\lambda)^{j-1} e(s_{t+j}). \end{aligned} \tag{6}$$

The first term is the start-state value error. The second term is the recursive bootstrap error from future critic errors. Define

$$F_t := \gamma(1-\lambda)\sum_{j=1}^{\infty}(\gamma\lambda)^{j-1}e(s_{t+j}).$$

Then

$$\hat{A}_t^{\mathrm{GAE}} - \tilde{A}_t^{\mathrm{GAE}} = -e(s_t) + F_t.$$

If $\|e\|_\infty \le \epsilon$, the recursive term is bounded by

$$|F_t| \le \frac{\gamma(1-\lambda)}{1-\gamma\lambda}\epsilon. \tag{7}$$

Chunked advantage. For a chunk beginning at t_0, ACPPO uses

$$\hat{A}_{t_0}^{\mathrm{chunk}} = \sum_{j=0}^{h-1}\gamma^j\delta_{t_0+j} + \gamma^h\hat{A}_{t_0+h}^{\mathrm{GAE}},$$

with oracle counterpart

$$\tilde{A}_{t_0}^{\mathrm{chunk}} = \sum_{j=0}^{h-1}\gamma^j\tilde{\delta}_{t_0+j} + \gamma^h\tilde{A}_{t_0+h}^{\mathrm{GAE}}.$$

Subtracting the two estimators gives

$$\hat{A}_{t_0}^{\mathrm{chunk}} - \tilde{A}_{t_0}^{\mathrm{chunk}} = \sum_{j=0}^{h-1}\gamma^j\left(\gamma e(s_{t_0+j+1}) - e(s_{t_0+j})\right) + \gamma^h\left(\hat{A}_{t_0+h}^{\mathrm{GAE}} - \tilde{A}_{t_0+h}^{\mathrm{GAE}}\right). \tag{8}$$

The first term telescopes:

$$\sum_{j=0}^{h-1}\gamma^j\left(\gamma e(s_{t_0+j+1}) - e(s_{t_0+j})\right) = \gamma^h e(s_{t_0+h}) - e(s_{t_0}).$$

Using Eq. 6 at time t_0+h,

$$\hat{A}_{t_0+h}^{\mathrm{GAE}} - \tilde{A}_{t_0+h}^{\mathrm{GAE}} = -e(s_{t_0+h}) + F_{t_0+h}.$$

Substituting this expression into Eq. 8 yields

$$\begin{aligned}\hat{A}_{t_0}^{\mathrm{chunk}} - \tilde{A}_{t_0}^{\mathrm{chunk}} &= \left(\gamma^h e(s_{t_0+h}) - e(s_{t_0})\right) + \gamma^h\left(-e(s_{t_0+h}) + F_{t_0+h}\right)\\ &= -e(s_{t_0}) + \gamma^h F_{t_0+h}.\end{aligned}$$

The intermediate value errors $e(s_{t_0+1}), \ldots, e(s_{t_0+h})$ cancel. If $\|e\|_\infty \le \epsilon$, then

$$\left|\gamma^h F_{t_0+h}\right| \le \frac{\gamma^{h+1}(1-\lambda)}{1-\gamma\lambda}\epsilon.$$

Compared with the recursive term in stepwise GAE in Eq. 7, the remaining recursive bootstrap-error contribution is discounted by an additional factor of γ^h.

Tradeoff. This cancellation does not imply that the chunked advantage is uniformly better. Inside the first h steps, the chunked estimator uses weights γ^j rather than the stepwise GAE weights $(\gamma\lambda)^j$. For $\lambda < 1$, these weights are larger, making the estimator closer to an h-step Monte Carlo target within the chunk. This can increase variance when rewards or transitions are noisy, especially for larger h. In addition, all actions in the chunk receive the same chunk-level advantage, which coarsens within-chunk credit assignment. These tradeoffs motivate using short chunks and comparing chunked/stepwise advantages.

Table 9: Chunk-level clipping diagnostics.

Method	Chunk length	Clip fraction
PPO	–	0.365
ACPPO-Corr	2	0.364
ACPPO-Corr	4	0.369
ACPPO-Corr	8	0.408

D Chunked Importance Sampling Ratio Clipping

Chunked importance sampling ratio. The ACPPO chunked importance ratio can be written as

$$\rho_{t_0}^{(h)}(\theta) = \frac{\pi_\theta^{(h)}(\mathbf{a}_{t_0:t_0+h-1} \mid s_{t_0})}{\pi_{\theta_{\text{old}}}^{(h)}(\mathbf{a}_{t_0:t_0+h-1} \mid s_{t_0})} = \prod_{k \in \mathcal{T}_{t_0}} \frac{\pi_{\theta,k}^{(h)}(a_{t_0+k} \mid s_{t_0})}{\pi_{\theta_{\text{old}},k}^{(h)}(a_{t_0+k} \mid s_{t_0})},$$

where $\pi_{\theta,k}^{(h)}$ denotes the marginal Gaussian for the k-th action in the chunk and $\mathcal{T}_{t_0}$ is the valid prefix of the chunk before any mid-chunk termination. PPO clipping is applied to the total chunked importance sampling ratio $\rho_{t_0}^{(h)}(\theta)$, not independently to each factor.

For ACPPO-Corr, the corrector observes the current state inside the chunk, so the likelihood factorizes along the realized closed-loop trajectory:

$$\rho_{t_0}^{\text{corr}}(\theta) = \prod_{k \in \mathcal{T}_{t_0}} \frac{\pi_\theta(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k)}{\pi_{\theta_{\text{old}}}(a_{t_0+k} \mid s_{t_0+k}, s_{t_0}, k)},$$

If the episode terminates inside a chunk, the product is truncated at the termination boundary and the remaining steps are masked from the regularizers.

Stability of the chunked importance sampling ratio. Although the chunk ratio is a product of per-step likelihood ratios, its log-ratio is a sum:

$$\log \rho_{t_0}^{(h)} = \sum_{k=0}^{h-1} \ell_{t_0+k}, \qquad \ell_t = \log \pi_\theta(a_t \mid \cdot) - \log \pi_{\theta_{\text{old}}}(a_t \mid \cdot).$$

For intuition, consider Gaussian policies with fixed covariance and small policy updates. The mean and the variance can be formulated as the KL divergence between likelihoods:

$$\mathbb{E}[\ell_t] = -\text{KL}(\pi_{\theta_{\text{old}}}(\cdot \mid \cdot) \,\|\, \pi_\theta(\cdot \mid \cdot)), \qquad \text{Var}(\ell_t) = 2\,\text{KL}(\pi_{\theta_{\text{old}}}(\cdot \mid \cdot) \,\|\, \pi_\theta(\cdot \mid \cdot)).$$

The variability of the chunk log-ratio is controlled by the accumulated KL over the chunk:

$$\text{Var}\left[\log \rho_{t_0}^{(h)}\right] \approx 2 \sum_{k=0}^{h-1} \text{KL}_{t_0+k}.$$

Following common PPO implementations, we adapt the learning rate based on the measured KL divergence:

$$\eta \leftarrow \begin{cases} \eta/1.5, & \text{if } \frac{1}{h}\sum_{k=0}^{h-1} \text{KL}_{t_0+k} > 2\kappa, \\ 1.5\eta, & \text{if } \frac{1}{h}\sum_{k=0}^{h-1} \text{KL}_{t_0+k} < 0.5\kappa, \\ \eta, & \text{otherwise.} \end{cases}$$

where η is the learning rate and κ is the KL threshold. Since the KL is controlled during PPO updates and our chunks are short, the chunked importance sampling ratio remains stable in practice. The clip fractions per chunk length are displayed in Table 9. For the main setting $h = 4$, ACPPO-Corr has a clip fraction of 0.369, close to PPO's 0.365. The clip fraction increases for $h = 8$, consistent with a larger accumulated chunk KL, but remains stable in our experiments.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the two main contributions: ACPPO, which introduces action chunking into PPO while retaining a standard state-value critic, and ACPPO-Corr, which adds a step-wise feedback corrector for within-chunk reactivity. Our experiments on 25 simulated continuous-control robotics tasks from Isaac Gym and Bi-DexHands demonstrate the effect of these methods.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the main limitations of the work. First, the method is evaluated only in simulated, state-based continuous-control robotics environments. Second, ACPPO-Corr assumes continuous action spaces where additive feedback corrections and magnitude regularization are meaningful, and therefore does not directly apply to discrete or semantically discontinuous action spaces. Third, performance depends on design choices such as chunk length and corrector regularization. Finally, the chunked importance-sampling ratio can become less stable for longer chunks, suggesting that substantially longer action chunks may require additional stabilization.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results. Appendices B–D provide derivations and diagnostic analyses used to motivate the proposed objective and implementation choices. These sections support the algorithmic design but are not claimed as formal theoretical contributions.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix A.1 details the benchmark setup, model sizes, network architectures, and hyperparameter selection procedures necessary for reproduction. We also provide the source code in our Supplementary Materials.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide anonymized source code in the supplementary material, including training scripts, configuration files, and commands for running the proposed method and baselines.. The simulation benchmarks are existing publicly available assets, and the paper specifies the environments, hyperparameters, normalization bounds, and evaluation protocol.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Hyperparameters, including the tuning ranges for the corrector regularization coefficient and the behavior regularization coefficient, as well as the use of the Adam optimizer, are detailed in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Main experimental results are reported over five random seeds. Figures and aggregate results include 95% confidence intervals across seeds. The reported intervals capture variability due to random seed initialization and stochastic training.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: We partially report the network parameter counts and used GPU devices. While we used a single RTX A6000 GPU to run multiple seeds in parallel, a single seed training can be performed on any RTX GPU device with 12GB+ VRAM. The required computational resources for all the simulation benchmarks are listed on their respective websites.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The research involves standard simulated robotics benchmarks and aligns to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work focuses on reinforcement learning algorithms evaluated in simulated environments and does not present direct societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper does not release high-risk foundational models, scraped datasets, or image generators

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Our experiments use existing simulation environments and learning-code assets. Isaac Gym Preview 4 is obtained separately from NVIDIA and used subject to the NVIDIA Isaac Gym License Agreement; we do not redistribute the Isaac Gym simulator. Several training and utility files are derived from NVIDIA IsaacGymEnvs/HEPO code and retain the original NVIDIA BSD-3-Clause copyright headers. The supplementary code is built on the HEPO repository, which is released under the MIT License and also includes NVIDIA BSD-3-Clause components. We also use and modify components from rl-games, which is released under the MIT License. Bi-DexHands is used as an external benchmark and is released under the Apache-2.0 License; users are instructed to install it from the original repository.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will create comprehensive documentation for our code, providing information about the original source.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not contain crowdsourcing experiments nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not contain crowdsourcing experiments nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Large Language Models are not an important, original, or non-standard component of the core reinforcement learning methods proposed in this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.