VALUE-GUIDED ADAPTIVE ACTION CHUNKING FOR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Action chunking emerged as a pivotal technique in imitation learning, enabling policies to predict cohesive action sequences rather than single actions. Recently, this approach has expanded to reinforcement learning (RL), enhancing behavioral consistency and reducing bootstrapping errors in value function estimation. However, existing methods rely on a fixed chunk length, creating a performance bottleneck as the optimal length varies across states and tasks. In this paper, we propose **A**daptive Action **CH**unking (ACH), a novel offline-to-online RL algorithm that dynamically modulates chunk length during both training and inference. For each state, ACH takes the action value of each candidate chunk length as its optimality criterion. We estimate the action values of all candidate chunk lengths simultaneously in a single forward pass by employing a causal Transformer-based value function. Based on these values, the agent adaptively selects the most effective chunk length for the current state. Evaluated on 34 challenging tasks, ACH outperforms fixed-length baselines in most environments, demonstrating enhanced learning efficiency. Moreover, ACH improves the performance of vision-language-action (VLA) models, demonstrating its applicability to large-scale models.

1 INTRODUCTION

Action chunking (Zhao et al., 2023) is a technique introduced in imitation learning (IL), where a policy predicts a sequence of consecutive actions, called an action chunk, instead of a single action per step. Since an action chunk is executed in an open-loop manner, the policy makes decisions once every several steps rather than at every step, enabling efficient inference. Moreover, while individually sampled single actions often result in redundant or inconsistent behaviors, an action chunk is generated as one cohesive sequence, thereby promoting behavioral consistency. Driven by these advantages, action chunking has emerged as a pivotal technique within the imitation domain (Zhao et al., 2023; Black et al., 2025b; Chi et al., 2025; Bjorck et al., 2025).

Recently, action chunking has expanded beyond IL into the realm of reinforcement learning (RL) (Sutton et al., 1998a). Specifically, in offline RL (Levine et al., 2020), several methods have successfully integrated action chunking (Li et al., 2026b;a; Kim et al., 2026; Song et al., 2026). These approaches focus on effective value function training for policies with a fixed chunk length. By aligning value estimation with action sequences rather than individual actions, these methods effectively mitigate the bootstrapping errors inherent in single-action value estimation. Consequently, they demonstrate the advantage of action chunking in offline RL, particularly in long-horizon tasks.

Building on these advantages, action chunking has been further extended to offline-to-online (Off2On) RL (Lee et al., 2022; Zhang et al., 2023; Nakamoto et al., 2023; Li et al., 2026b). In Off2On RL, a policy pre-trained via offline RL is fine-tuned through online interaction with the environment, allowing it to surpass the performance limit imposed by the quality of the given dataset. In this setting, the behavioral consistency of action chunking enables efficient exploration during online fine-tuning, complementing its benefits in value function learning (Li et al., 2026b).

However, these methods keep the chunk length fixed throughout training and inference, so their performance hinges on a single value that is rarely optimal for every task, or even for every state within a single task. Consider, for example, the antmaze navigation task (Park et al., 2025a), where an ant must traverse a complex, large-scale maze to reach a goal, as illustrated in Figure 1. When the ant is in the segment shown in Subfigure 1, the appropriate action changes rapidly, so an excessively

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

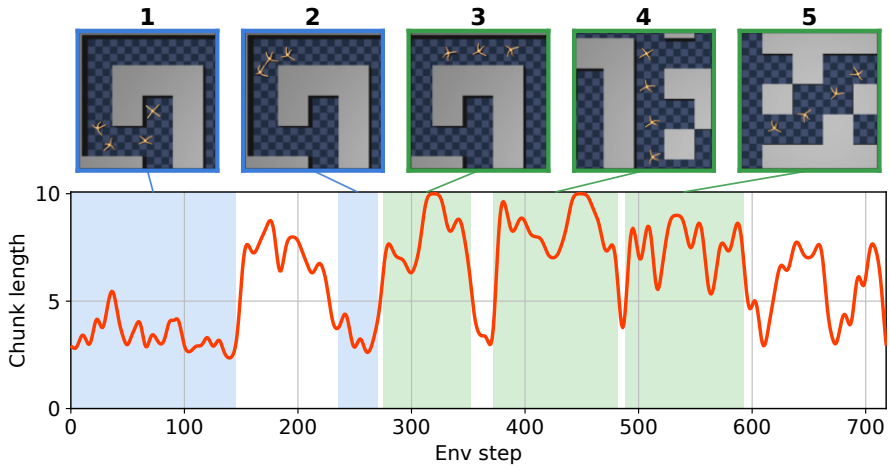


Figure 1: Adaptive Action Chunking: Chunk lengths selected by our proposed algorithm, ACH, over environment timesteps during an `antmaze-giant` episode. Shaded regions correspond to the subfigures above. ACH selects shorter chunks (blue) in segments where the required action changes rapidly, such as around corners, and longer chunks (green) where the trajectory is predictable. A similar pattern in a manipulation task is shown in Appendix D.1.

large chunk length may cause frequent collisions or make the agent overshoot turning points, thereby degrading performance. In such scenarios, a shorter chunk length is preferred. Conversely, in the segment shown in Subfigure 4, where the trajectory is more predictable, the agent should ideally maintain its momentum to reach the goal efficiently. In this case, a larger chunk length is more advantageous. These variations underscore the need to move beyond rigid, fixed-length approaches toward a more adaptive action chunking strategy.

In this paper, we propose **A**daptive Action **CH**unking (ACH), a novel Off2On RL algorithm designed to address these limitations. Our method enables the policy to optimally adjust its chunk length based on the current state, as illustrated in Figure 1. As the criterion for this adjustment, we use the action value of each candidate chunk length, which naturally reflects the effectiveness of the corresponding chunk. These values are estimated simultaneously in a single forward pass by employing a causal Transformer-based value function, allowing ACH to select the most suitable chunk length efficiently. We evaluate our approach on 34 diverse and challenging tasks, where experimental results demonstrate that ACH successfully modulates chunk length across various environments, outperforming existing Off2On RL methods in most environments. We further apply ACH to a vision-language-action (VLA) model, where it outperforms all baselines on two robot embodiments, showing that ACH extends to large-scale models.

2 Preliminaries

Offline-to-online RL In this paper, we consider a Markov Decision Process (MDP) (Sutton et al., 1998a) defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \rho, \mathcal{P}, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the reward function, $\rho \in \Delta(\mathcal{S})$ is the initial state distribution, $\mathcal{P}(s'|s,a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, and $\gamma \in [0,1]$ is the discount factor. The objective of RL is to find a policy $\pi(a|s)$ that maximizes the expected return: $\mathbb{E}_\pi \left[\sum_k \gamma^k r(s_k, a_k)\right]$.

Off2On RL is a two-stage learning framework that performs offline pre-training and subsequent online learning (Lee et al., 2022; Nakamoto et al., 2023; Zhou et al., 2025; Shin et al., 2026). In the offline pre-training stage, the agent is trained on a pre-collected dataset $D = \{(s, a, r, s')\}$ without any interaction with the environment. During the subsequent online learning stage, the trained policy interacts directly with the environment, further refining its behavior.

Action Chunking in RL Action Chunking, originally introduced in IL to improve robotic control efficiency (Zhao et al., 2023), has become a key technique in robotics research (Black et al., 2025b; Chi et al., 2025; Bjorck et al., 2025). By grouping a sequence of actions into a single unit, referred to as an action chunk, this approach reduces decision-making frequency while promoting consistent

execution. Recently, the action chunking approach has been extended to RL (Li et al., 2025a; 2026b;a; Yang et al., 2026). Instead of learning the value of a single action a_t as in conventional approaches, the value of an action chunk $\boldsymbol{a}_{t:t+h} := (a_t, a_{t+1}, \cdots, a_{t+h})$ is learned to enable faster value propagation and unbiased value estimation, thereby facilitating effective and stable temporal-difference (TD) learning (Li et al., 2026b;a; Kim et al., 2026).

In the context of Off2On RL, action chunking can be further leveraged to enhance online exploration through consistent behavior. Q-Chunking (QC) (Li et al., 2026b) combines this exploration benefit with the value learning advantages of chunking, employing a flow-based policy as its backbone. To this end, a flow-based behavior policy β_ψ is trained by learning a velocity field v_ψ in a chunk-wise manner:

$$\mathcal{L}_\beta(\psi) = \mathbb{E}_{\substack{(s_t, \boldsymbol{a}_{t:t+h}) \sim D, \\ u \sim \text{Unif}([0,1]), \\ \mathbf{z} \sim \mathcal{N}(0, I_{A(h+1)})}} \left[\left\| v_\psi(s_t, \boldsymbol{a}_u, u) - (\boldsymbol{a}_{t:t+h} - \mathbf{z}) \right\|_2^2 \right] \tag{1}$$

where $\boldsymbol{a}_u = (1-u)\mathbf{z} + u\boldsymbol{a}_{t:t+h}$ and A denotes the action space dimension. Unlike conventional single-action learning, the velocity field here is defined over action chunks. By integrating the trained velocity field v_ψ from initial noise $\mathbf{z} \sim \mathcal{N}(0, I_{A(h+1)})$, the behavior policy samples an action chunk $\boldsymbol{a}_{t:t+h}$ for a given state s_t. Subsequently, behavior-regularized Q-maximization is applied to derive a one-step flow policy (Park et al., 2025b):

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{\substack{s_t \sim D, \\ \mathbf{z} \sim \mathcal{N}(0, I_{A(h+1)})}} \Big[-Q_\phi(s_t, \pi_\theta(s_t, \mathbf{z})) + \alpha \left\| \pi_\theta(s_t, \mathbf{z}) - \beta_\psi(s_t, \mathbf{z}) \right\|_2^2 \Big] \tag{2}$$

where α is a hyperparameter controlling the strength of the regularization. Beyond policy action generation, the action-value function is also learned in a chunk-wise manner via temporal-difference (TD) learning:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{\substack{(s_t, \boldsymbol{a}_{t:t+h}, \boldsymbol{r}_{t:t+h}, s_{t+h+1}) \sim D, \\ \boldsymbol{a}'_{t+h+1:t+2h+1} \sim \pi_\theta(\cdot | s_{t+h+1})}} \Big[\big(Q_\phi(s_t, \boldsymbol{a}_{t:t+h}) - R_{t:t+h} - \gamma^{h+1} Q_{\bar{\phi}}(s_{t+h+1}, \boldsymbol{a}'_{t+h+1:t+2h+1}) \big)^2 \Big] \tag{3}$$

where $R_{t:t+h} = \sum_{k=0}^{h} \gamma^k r_{t+k}$ and $\bar{\phi}$ denotes the target network parameters. In this way, both the policy and the value function operate on action chunks rather than individual actions.

3 METHOD

In this section, we present ACH, a novel framework for Off2On RL. We first introduce a value-based criterion for selecting the chunk length (Section 3.1), then describe a Transformer-based value function that estimates the values of all candidate chunk lengths in parallel (Section 3.2), and finally present the adaptive chunk length sampling mechanism (Section 3.3).

3.1 VALUE-BASED CHUNK LENGTH SELECTION

To adapt the chunk length at each state s_t, we need a criterion that reflects how the chunk length affects performance. Intuitively, the chunk length involves a trade-off between two opposing factors. The first is *the uncertainty of predicted future states*: since an action chunk is sampled conditioned solely on s_t and executed in an open-loop manner, its actions become suboptimal when the actual trajectory deviates from the states implicitly predicted at time t, which favors shorter chunks. The second is *policy stochasticity jittering loss*: when future states are highly correlated with s_t, resampling at every step interrupts the continuity of behaviors that a longer chunk would provide (Li et al., 2026b), and in stochastic policies such as flow-based models, each independent sampling introduces jitter that hinders the agent's progress toward the goal, which favors longer chunks. Thus, a chunk length of one is not always optimal even when computational cost is disregarded, and the best chunk length depends on how these two factors balance at each state.

One way to balance this trade-off is to design heuristic criteria, such as bounding the estimated entropy of predicted future states or actions, possibly combined with a penalty on action jittering. A recent study by Liang et al. (2026) falls into this category, adjusting the chunk length at inference time based on the average entropy of the predicted actions and its maximum gap. However, such criteria affect the return only indirectly, so the resulting chunk length may be suboptimal, and they require multiple hand-designed functions and thresholds.

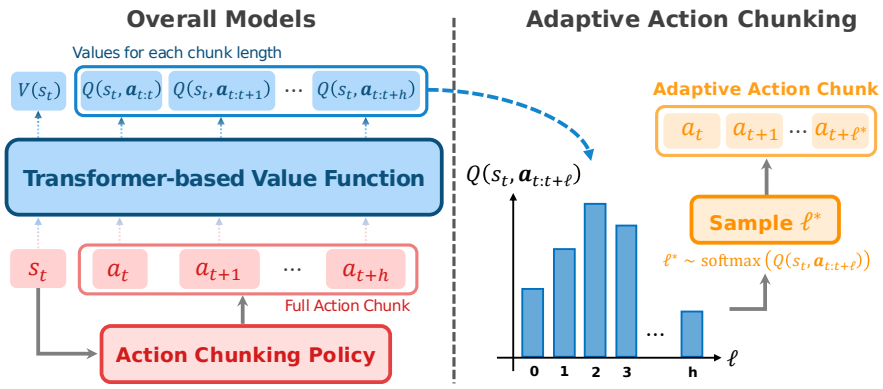


Figure 2: The overall architecture of ACH. (left) Causal Transformer-based joint value learning: from $(s_t, \boldsymbol{a}_{t:t+h})$, the value function outputs $V(s_t)$ and the action values of all prefix chunks in a single forward pass. (right) Chunk length sampling: the chunk length is drawn from the softmax distribution over these action values.

Instead of relying on such heuristics, we take a different perspective and directly employ the values of action chunks as our optimality criterion, aligning with the fundamental goal of RL: return maximization. Concretely, we construct a set $\mathcal{A}_h$ of action chunks with lengths up to h, all originating from s_t, as follows:

$$\mathcal{A}_h = \{(s_t, a_t), (s_t, a_t, a_{t+1}), \cdots, (s_t, a_t, a_{t+1}, \cdots, a_{t+h})\}, \tag{4}$$

where each element includes the preceding element as its prefix. To compare these chunks, we evaluate each of them by its value under a policy π, i.e., the expected return of executing the chunk $(a_t, \cdots, a_{t+n})$ from s_t and following π afterward:

$$Q^\pi(s_t, a_t, \cdots, a_{t+n}) = \mathbb{E}_{a_{t+n+1}, a_{t+n+2}, \cdots \sim \pi}\left[\sum_k \gamma^k r_{t+k} \,\middle|\, s_t, a_t, \cdots, a_{t+n}\right] \tag{5}$$

Evaluating every chunk in $\mathcal{A}_h$ in this way yields the value sequence

$$Q^\pi(s_t, a_t), Q^\pi(s_t, a_t, a_{t+1}), \cdots, Q^\pi(s_t, a_t, a_{t+1}, \cdots, a_{t+h}). \tag{6}$$

This sequence progressively evaluates the value of action chunks $(a_t, \cdots, a_{t+n})$ for given s_t, with n ranging up to h. In other words, the sequence describes how the expected return changes as the agent commits to a longer chunk at s_t before making its next decision, as illustrated in Appendix D.2.

The key of this approach lies in the prefix structure of the chunk sequence. Specifically, if we were to compare $Q^\pi(s_t, a_t, a_{t+1})$ and $Q^\pi(s_t, a'_t, a'_{t+1}, a_{t+2})$ where $(a_t, a_{t+1}) \neq (a'_t, a'_{t+1})$, the difference in Q-values could stem from the initial action mismatch rather than the chunk length itself. By maintaining a prefix-consistent sequence, we can isolate the effect of length and determine the optimal chunk length within the range $[1, h+1]$ by selecting the index that maximizes the value in the sequence in (6).

3.2 Transformer-based Multi-Chunk Value Estimation

The main hurdle to realizing our adaptive chunk length selection described above is how to learn the Q-values of multiple increasing chunks in (6) simultaneously. To solve this problem, we adopt a *causal Transformer* as our value function architecture. Building on previous work (Li et al., 2025a), which utilized a Transformer-based value function for episodic RL, we design a value function that takes $(s_t, \boldsymbol{a}_{t:t+h})$ as input and outputs $(V(s_t), Q(s_t, \boldsymbol{a}_{t:t}), Q(s_t, \boldsymbol{a}_{t:t+1}), \cdots, Q(s_t, \boldsymbol{a}_{t:t+h}))$, as illustrated in Figure 2, where the state value function $V(s)$ is additionally included for efficient bootstrapping. With a causal Transformer as the value function, we can estimate the values of all possible action chunks in a single forward pass, while ensuring that action chunks do not attend to future steps.

To train the value function under multiple chunk lengths, we follow the training scheme of the previous work (Li et al., 2025a) and modify it for adaptive chunk lengths, which yields the following

objective that trains the entire value sequence in (6) at once under the current policy:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{\substack{(\boldsymbol{s}_{t:t+h}, \boldsymbol{a}_{t:t+h}, \\ \boldsymbol{r}_{t:t+h}, s_{t+h+1}) \sim D, \\ \boldsymbol{a}'_{t:t+h} \sim \pi_\theta(\cdot|s_t)}} \Bigg[\frac{1}{h+1} \sum_{n=0}^{h} \Bigg(Q_\phi(s_t, \boldsymbol{a}_{t:t+n}) - \big(R_t^n + \gamma^{n+1} V_{\bar{\phi}}(s_{t+n+1}) \big) \Bigg)^2 + \Bigg(V_\phi(s_t) - \mathbb{E}_{\ell \sim \pi_L(\cdot|s_t, \boldsymbol{a}'_{t:t+h})} \big[Q_{\bar{\phi}}(s_t, \boldsymbol{a}'_{t:t+\ell}) \big] \Bigg)^2 \Bigg] \tag{7}$$

where $R_t^n = \sum_{k=0}^{n} \gamma^k r_{t+k}$, $\boldsymbol{a}'_{t:t+\ell}$ is a truncation (up to ℓ) of full chunk $\boldsymbol{a}'_{t:t+h}$ sampled from $\pi_\theta(\cdot|s_t)$, and $\pi_L(\ell|s_t, \boldsymbol{a}'_{t:t+h})$ denotes the chunk length selection policy over $\ell \in \{0, 1, \ldots, h\}$, where ℓ corresponds to executing the prefix $\boldsymbol{a}'_{t:t+\ell}$, i.e., a chunk length of $\ell + 1$. This policy will be explained shortly in Section 3.3.

The loss above consists of two terms. The first term learns the action value for each chunk length n, which corresponds to the n-th element of (6), using the n-step target. By estimating the value of the action chunk $\boldsymbol{a}_{t:t+n}$ covering the n-step reward $\boldsymbol{r}_{t:t+n}$, the proposed method yields less biased value estimation. Moreover, unlike the previous approach (Li et al., 2026b) that requires the state-action value of the next n-step chunk $\boldsymbol{a}_{t+n+1:t+2n+1}$ when applying bootstrapping, our method instead builds the target from the value of the state s_{t+n+1}, thereby avoiding the variance from action-based bootstrapping. The second term learns the state value from the current policy's behavior. In doing so, it performs value estimation according to the chunk length selection distribution. We further discuss the connections of Eq. (7) to multi-step TD learning and the options framework in Appendix B.

Based on the learned value function, we train the policy π_θ by jointly optimizing distillation from the behavior policy β_ψ learned via flow matching and action-value maximization:

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{\substack{s_t \sim D, \\ \mathbf{z} \sim \mathcal{N}(0, I_{A(h+1)}), \\ \boldsymbol{a}'_{t:t+h} \sim \pi_\theta(s_t, \mathbf{z})}} \Big[- \mathbb{E}_{\ell \sim \pi_L(\cdot|s_t, \boldsymbol{a}'_{t:t+h})} \big[Q_\phi(s_t, \boldsymbol{a}'_{t:t+\ell}) \big] + \alpha \left\| \boldsymbol{a}'_{t:t+h} - \beta_\psi(s_t, \mathbf{z}) \right\|_2^2 \Big], \tag{8}$$

where α is a behavior cloning coefficient. Note that the value-maximization term is taken in expectation over $\ell \sim \pi_L$, so the policy is optimized for the chunk lengths that it will actually execute rather than for a single fixed length. This keeps the training objective aligned with the adaptive execution, guiding the policy to learn action chunks with high action values at the lengths that π_L selects during environment interaction.

3.3 Adaptive Chunk Length Sampling

We generate a full action chunk $\boldsymbol{a}'_{t:t+h}$ for current state s_t using the policy $\pi_\theta(\cdot|s_t)$, and feed the full sequence $(s_t, \boldsymbol{a}'_{t:t+h})$ to the transformer-based value function. This yields the Q-values of all $h+1$ prefix chunks: $(s_t, \boldsymbol{a}'_{t:t}), (s_t, \boldsymbol{a}'_{t:t+1}), \cdots, (s_t, \boldsymbol{a}'_{t:t+h})$. One could then select the optimal chunk length for s_t as $h^* = 1 + \arg\max_{n \in \{0,1,\cdots,h\}} Q_\phi(s_t, \boldsymbol{a}'_{t:t+n})$, resulting in a greedy selection policy. However, inspired by ϵ-greedy strategies in Q-learning, we further refine this selection process to incorporate exploration capabilities during online learning. Consequently, we define the chunk length selection policy, π_L , as a soft policy following a sampling distribution:

$$\pi_L(\ell) = \frac{\exp(\beta \cdot Q_\phi(s_t, \boldsymbol{a}'_{t:t+\ell}))}{\sum_{n=0}^{h} \exp(\beta \cdot Q_\phi(s_t, \boldsymbol{a}'_{t:t+n}))}, \quad \forall \ell \in \{0, 1, \cdots, h\} \tag{9}$$

where β denotes the inverse temperature parameter. Since both the policy and the value function are trained to prioritize chunks with high action values in the offline dataset via Eqs. (7) and (8), this method naturally selects action chunks that align with the training dynamics. This probabilistic sampling encourages the agent to explore diverse chunk lengths, thereby enabling it to identify the most effective one. At evaluation, we select the chunk length with the highest estimated value to obtain deterministic behavior.

4 Experiments

In this section, we empirically evaluate the proposed method, ACH, across challenging environments involving long-horizon and diverse tasks in comparison with several key baselines.

Table 1: Aggregated results of ACH and baselines on OGBench. All methods are evaluated after 1M steps of offline learning followed by 1M steps of online learning. We report the mean and 95% confidence intervals across 10 seeds. Results for each environment are averaged over five tasks.

Task	RLPD	ReBRAC	BFN	FQL	FINO	QC	ACH
antmaze-giant-navigate	$0_{\pm 0} \rightarrow 69_{\pm 9}$	$30_{\pm 6} \rightarrow \mathbf{99}_{\pm 0}$	$0_{\pm 0} \rightarrow 22_{\pm 6}$	$10_{\pm 5} \rightarrow 74_{\pm 7}$	$2_{\pm 2} \rightarrow 76_{\pm 4}$	$0_{\pm 0} \rightarrow 47_{\pm 8}$	$1_{\pm 1} \rightarrow 72_{\pm 2}$
antsoccer-arena-navigate	$0_{\pm 0} \rightarrow 64_{\pm 4}$	$1_{\pm 1} \rightarrow 0_{\pm 0}$	$51_{\pm 2} \rightarrow 74_{\pm 7}$	$35_{\pm 2} \rightarrow 70_{\pm 5}$	$40_{\pm 3} \rightarrow 71_{\pm 5}$	$10_{\pm 2} \rightarrow 67_{\pm 2}$	$0_{\pm 0} \rightarrow \mathbf{84}_{\pm 1}$
cube-quadruple-play-10M	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$2_{\pm 1} \rightarrow 10_{\pm 2}$	$0_{\pm 0} \rightarrow 2_{\pm 1}$	$1_{\pm 0} \rightarrow 7_{\pm 3}$	$1_{\pm 1} \rightarrow 33_{\pm 3}$	$0_{\pm 0} \rightarrow \mathbf{52}_{\pm 6}$
cube-triple-play	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$4_{\pm 1} \rightarrow 22_{\pm 3}$	$2_{\pm 1} \rightarrow 19_{\pm 2}$	$2_{\pm 1} \rightarrow 24_{\pm 3}$	$4_{\pm 2} \rightarrow 57_{\pm 4}$	$1_{\pm 1} \rightarrow \mathbf{67}_{\pm 3}$
humanoidmaze-medium-navigate	$0_{\pm 0} \rightarrow 29_{\pm 9}$	$34_{\pm 4} \rightarrow 4_{\pm 2}$	$59_{\pm 3} \rightarrow 87_{\pm 6}$	$36_{\pm 3} \rightarrow 56_{\pm 3}$	$38_{\pm 5} \rightarrow \mathbf{92}_{\pm 4}$	$15_{\pm 2} \rightarrow 32_{\pm 4}$	$45_{\pm 2} \rightarrow 82_{\pm 1}$
puzzle-4x4-play	$0_{\pm 0} \rightarrow 64_{\pm 12}$	$14_{\pm 2} \rightarrow 32_{\pm 6}$	$32_{\pm 2} \rightarrow 60_{\pm 7}$	$14_{\pm 2} \rightarrow 54_{\pm 6}$	$15_{\pm 3} \rightarrow 53_{\pm 7}$	$11_{\pm 2} \rightarrow 38_{\pm 4}$	$13_{\pm 3} \rightarrow \mathbf{90}_{\pm 8}$

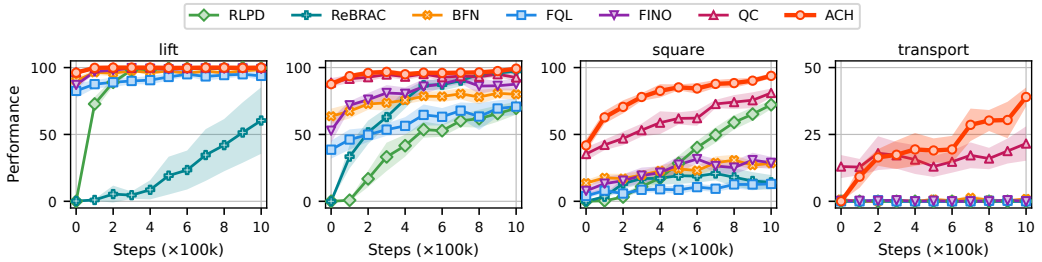


Figure 3: Learning curves over 1M online steps across robomimic tasks. The performance is averaged over 10 random seeds, with the 95% confidence interval as the shaded region.

Benchmarks and Tasks We evaluate ACH on 30 tasks from six environments of OGBench (Park et al., 2025a), spanning locomotion and manipulation, and on four robotic manipulation tasks from robomimic (Mandlekar et al., 2021). Details of the tasks and datasets are provided in Appendix C.1.

Baselines We compare ACH against the following Off2On RL methods: (1) RLPD (Ball et al., 2023), an online RL algorithm that starts with an offline dataset and achieves high performance without an offline learning phase. (2) ReBRAC (Tarasov et al., 2023), an offline RL method built on a Gaussian policy, which has shown strong performance in both offline and Off2On RL. (3) BFN is a method introduced in the QC paper (Li et al., 2026b), which draws multiple action candidates from a behavior policy and selects the one with the highest estimated action value. (4) FQL (Park et al., 2025b), an offline RL method that learns a behavior policy via flow matching and distills it into a one-step policy, demonstrating strong performance across offline and Off2On RL. (5) FINO (Shin et al., 2026), an algorithm targeting Off2On RL with a flow-based policy, which builds upon FQL and injects noise into the flow matching for strong Off2On performance. (6) QC (Li et al., 2026b), as discussed in Section 2, extends the conventional single-action approach to action chunks, and serves as our backbone algorithm.

Implementation Details We build ACH upon QC (Li et al., 2026b), using its variant that trains a one-step policy for efficient inference, and set the maximum chunk length to 10 ($h = 9$) in most environments. Further implementation details, hyperparameters, and the pseudo-code are provided in Appendix C.4.

Evaluations For all baselines, we apply the same protocol to obtain experimental results. Training consists of 1M steps for offline learning followed by 1M steps for online learning, and the final results are based on the performance evaluated at the last step. All experiments are conducted with 10 random seeds per task, and we report the mean and 95% confidence intervals. To fairly compare the performance improvement during online learning, we also report the offline performance. We bold all results within 95% of the top-performing algorithm after online learning.

Results Table 1 summarizes the performance across 30 OGBench tasks, aggregated by environment. The results demonstrate that ACH achieves superior performance across diverse domains. Specifically, RLPD, which conducts online RL without a prior offline phase, exhibits clear limitations inherent in its learning paradigm. Although ReBRAC achieves strong performance in `antmaze`, it struggles in more complex environments such as `antsoccer` and `humanoidmaze`, where it fails to improve or even collapses during online learning, which we attribute to the limited expressiveness of its unimodal policy. BFN employs a sampling strategy similar to the proposed method, yet fails to achieve strong performance due to the inherent limitation of single-action prediction. Moreover, the performance

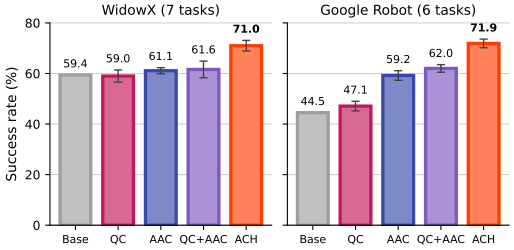


Figure 4: Success rates on SimplerEnv, averaged over 7 WidowX tasks and 6 Google Robot tasks. Error bars denote 95% confidence intervals. Per-task results are reported in Appendix E.2.

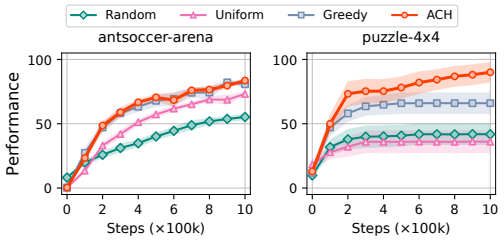


Figure 5: Experimental comparison of ACH against alternative training and sampling strategies. Each plot presents the aggregated performance across five distinct tasks.

of FQL suggests that simply extending offline RL methods to online fine-tuning is insufficient for addressing the challenges of the Off2On transition. While FINO performs competitively in navigation-heavy tasks such as `humanoidmaze`, its performance in manipulation tasks (e.g., `cube`) reveals the limitations of single-action prediction in scenarios requiring precise manipulation. Lastly, while QC demonstrates that action chunking yields competitive results in most tasks, its comparison with ACH underscores that our proposed method facilitates more efficient learning. This gain is primarily attributed to ACH's ability to employ adaptive chunk lengths, as opposed to the rigid, fixed-length approach of the baseline.

Figure 3 illustrates the learning curves for four environments within robomimic during the online fine-tuning. The results demonstrate that ACH consistently achieves the highest performance across all tasks. This performance gain is particularly pronounced in the `square` and `transport` tasks, where the backbone algorithm with a fixed chunk length exhibits only marginal improvements. These findings, together with the results in Table 1, validate ACH as a robust framework that outperforms conventional methods by adaptively modulating chunk lengths across challenging domains.

4.1 Scaling to Vision-Language-Action Models

Vision-language-action (VLA) models have been actively studied in robotics, and many of them adopt action chunking (Black et al., 2025b; Bjorck et al., 2025). To examine whether ACH also improves such large-scale models, we apply it to a VLA model. We evaluate on SimplerEnv (Li et al., 2025b), covering seven tasks with the WidowX robot and six tasks with the Google Robot, and all methods build upon the same GR00T N1.7 model (Bjorck et al., 2025), which serves as the base policy. We compare ACH against the following methods: (1) Base, the base policy without any additional training; (2) QC (Li et al., 2026b), the backbone algorithm of ACH, which is trained and executed with a fixed chunk length; (3) AAC (Liang et al., 2026), which adapts the chunk length of the base policy at inference time based on the entropy of the predicted actions; (4) QC+AAC, which trains the policy with QC and additionally applies AAC at inference time.

Following the base policy, which predicts action chunks of up to 8 steps, we set the chunk length to 8 for all methods, which serves as the maximum chunk length for the adaptive methods. For QC and ACH, we keep the VLA model frozen and train only lightweight one-step policy and value networks through offline-to-online RL. For offline training, we collect 1,000 episodes per task with the base policy and use them to train the one-step policy and the value function of each method, followed by online training for 100K environment steps. Each task is evaluated over 200 episodes, and we report the mean and 95% confidence intervals across 4 seeds. Further details are provided in Appendix C.5.

As shown in Figure 4, ACH achieves the highest success rates on both robots, improving the average success rate of the base policy from 59.4% to 71.0% on WidowX and from 44.5% to 71.9% on the Google Robot. Although QC is trained through offline-to-online RL, it performs similarly to the base policy, indicating that training alone cannot compensate for a rigid execution horizon. AAC improves the base policy on the Google Robot without additional training, but its gain on WidowX is marginal, as its entropy-based heuristic reflects the resulting return only indirectly. QC+AAC yields only modest gains over AAC, indicating that merely coupling a policy trained under a fixed chunk length with an inference-time heuristic cannot fully exploit adaptive chunking, since the chunk length is never taken into account during training. In contrast, ACH adapts the chunk length during both

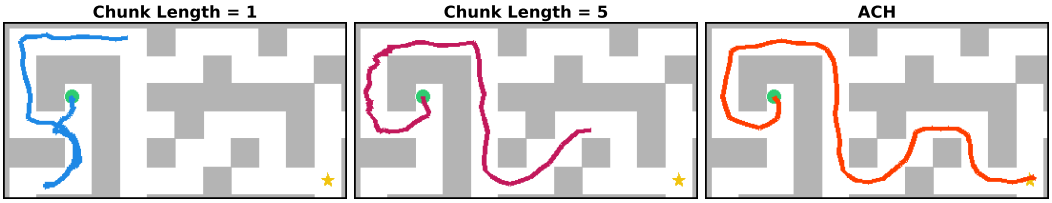


Figure 6: Behavioral comparison of ACH and fixed-length baselines in the `antmaze-giant` environment. Starting from a shared initial state, all algorithms execute the same number of total action steps. Agent trajectories are constructed by connecting sequential positions at each timestep. The initial state and goal state are marked by a green circle and a yellow star, respectively.

training and inference, aligning the training objective with the chunk lengths actually executed, and thus outperforms all baselines. These results demonstrate that ACH is also applicable to large-scale models that take visual observations as input.

4.2 Further Analysis

In this section, we present an ablation study on the core components of ACH and a behavior analysis that shows how adaptive action chunking shapes the behavior of the agent. Additional analyses are provided in Appendix D.

Ablation Study Section 3 presents the rationale for the value-based chunk length selection criterion (Section 3.1), describes the training of the value function and policy (Section 3.2), and details the adaptive chunk length sampling procedure (Section 3.3). To assess the impact of these core components, we conduct an ablation study by replacing each with an alternative approach. For the chunk length selection criterion, we examine a variant that randomly selects a length among all candidates instead of value-based sampling (denoted as `Random`). Regarding the training procedure, we evaluate a variant that assigns a uniform weight to all chunk length candidates, i.e., $\ell \sim \text{Unif}\{0, 1, \dots, h\}$ in Eqs. (7) and (8) rather than prioritizing the distribution π_L within the training losses (denoted as `Uniform`). To evaluate the sampling mechanism, we replace our soft sampling strategy Eq. (9) with a greedy approach that selects the chunk length with the highest estimated value during online interaction (denoted as `Greedy`).

As shown in Figure 5, replacing any of these components leads to a performance decline, confirming that no substitution matches the effectiveness of our proposed method. Simply selecting the chunk length at random does not consider the expected return of executing the corresponding action chunk, resulting in inferior performance across all environments. Training uniformly across all chunk lengths results in significant degradation, which we attribute to the misalignment between the training objective and the action selection policy used during online learning. The greedy alternative achieves competitive performance in some environments due to its behavior similarity to our method; however, since it solely exploits the Q-values, it fails to explore sufficiently. Consequently, it exhibits suboptimal sample efficiency compared to ACH. These findings confirm that the synergy between our training and sampling mechanisms is essential to enabling sample-efficient learning, which is consistently observed on robomimic (Appendix D.6).

Behavior Analysis To verify the efficacy of the adaptive chunk length selection, we conducted a comparative analysis in the challenging `antmaze-giant` environment. In Figure 6, we compare the trajectories of our proposed method against two baselines: a model that does not utilize action chunking (employing a fixed chunk length of 1) and a backbone model using a static, fixed chunk length of 5. All baselines are trained within the same learning timestep, and the trajectories are recorded by sampling from the final trained policies.

The results show that the model without action chunking exhibits redundant, oscillatory (i.e., jittering) behaviors near the initial state. This is because the policy predicts only a single action at each step, leading to frequent inconsistencies in decision making and highlighting the necessity of action chunking. While the fixed-length action chunking demonstrates improved behavior compared to single-action prediction, it still suffers from suboptimal efficiency. For instance, it fails to execute efficient maneuvers in areas requiring precise control, such as navigating corners, and ultimately fails

to reach the goal within the timestep limit. In contrast, our method selects the most appropriate chunk length for every state, resulting in efficient and smooth trajectories that successfully reach the goal. These behavioral patterns are accumulated as high-quality transitions during online learning, which facilitates sample-efficient training and enhanced performance. We present the results on the full map of Figure 6 and additional behavioral patterns discovered for chunk length 5 in Appendix E.3.

5 Related Work

Offline-to-online RL Off2On RL is a framework that addresses the inability of offline RL to learn beyond the given dataset and the safety and cost issues of applying online RL from scratch (Lee et al., 2022; Zhang et al., 2023; Nakamoto et al., 2023; Yu & Zhang, 2023; Wang et al., 2023; Luo et al., 2024; Zhang et al., 2024; Shin et al., 2025; Zhou et al., 2025; Shin et al., 2026). The objective is to leverage the offline dataset to prepare a policy, then refine it through direct environment interaction. The most straightforward approach is to extend existing offline RL methods (Kumar et al., 2020; Kostrikov et al., 2022; Wu et al., 2022; Tarasov et al., 2023) to online learning, but such approaches fail to fully exploit the interaction due to the inherited conservatism of offline RL. Prior works (Lee et al., 2022; Nakamoto et al., 2023; Yu & Zhang, 2023; Wang et al., 2023; Luo et al., 2024; Zhang et al., 2024; Shin et al., 2025) proposed easing conservatism, yet these methods still retain it, limiting sample efficiency. Recently, several works (Hansen-Estruch et al., 2023; Park et al., 2025b; Dong et al., 2026; Shin et al., 2026; Chae et al., 2026) have sought to leverage the multimodality of expressive generative models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Albergo & Vanden-Eijnden, 2022; Lipman et al., 2023) as policies, but since they rely on single-action prediction, they tend to produce redundant or inconsistent behaviors, resulting in limited sample efficiency. A recently proposed action chunking-based method (Li et al., 2026b) achieves strong performance through consistent behavior and value learning benefits, yet remains suboptimal due to its fixed chunk length. Our method employs adaptive action chunking to select the optimal chunk length for each state, enabling sample-efficient learning within a limited online budget.

Action Chunking for RL Action chunking has become a key technique in imitation learning by addressing compounding errors and dataset non-stationarity arising from single-action prediction (Zhao et al., 2023; Black et al., 2025b; Chi et al., 2025; Bjorck et al., 2025; Black et al., 2025a; Kim et al., 2025). This technique has been extended to reinforcement learning: in offline RL, various studies (Li et al., 2026b; Seo & Abbeel, 2026; Li et al., 2026a; Song et al., 2026; Kim et al., 2026; Park et al., 2026a) have leveraged action chunking, as the resulting horizon reduction facilitates value learning, particularly for long-horizon tasks, while in online RL, prior works (Li et al., 2025a; Tian et al., 2026; Nagy et al., 2026) have proposed Transformer-based value functions using action chunks to mitigate bias from n-step returns. However, these approaches suffer from limited sample efficiency due to fixed chunk length. Although prior work (Liang et al., 2026) in the imitation domain introduces adaptive chunk length selection, it targets only inference-time adjustment without incorporating adaptation into training. In contrast, our method leverages action chunking while adaptively adjusting its length during both training and inference, leading to strong sample efficiency in Off2On RL. We further discuss the connections to multi-step TD learning, the options framework, and hierarchical RL in Appendix B.

6 Conclusion

In this paper, we introduced ACH, a novel offline-to-online RL algorithm that dynamically modulates chunk length during both training and inference. To determine optimal chunk length for a current state, we adopt chunk value as an optimality criterion and employ a causal Transformer-based critic to simultaneously evaluate all candidate chunk lengths in a single forward pass. To enhance exploration during online fine-tuning, we developed a soft policy that adaptively samples chunk lengths based on these values. Our evaluation across 34 challenging tasks demonstrates that ACH outperforms fixed-length baselines in most environments with enhanced learning efficiency. Furthermore, experiments with a vision-language-action model show that ACH is applicable to large-scale models. We believe that our work opens a new avenue for leveraging adaptive action chunking in reinforcement learning, and that this direction will play an important role in advancing efficient learning in sequential decision making.

AI USE STATEMENT

In this work, we used generative AI tools for only writing-related tasks. We have not used generative AI tools for any of the tasks with required disclosure. Additionally, we used generative AI tools for tasks with recommended disclosure, namely editing the paper to improve readability (e.g., correcting typos and grammatical errors). We have reviewed all AI-assisted work: every suggested edit was manually checked by the authors against the original content before being incorporated. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or any personally identifiable information. All experiments are conducted in simulated environments using only publicly available benchmarks and datasets, namely OGBench (Park et al., 2025a) and robomimic (Mandlekar et al., 2021), whose sources are cited throughout the paper.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of our results. The proposed method is described in Section 3, and its full procedure is summarized in Algorithm 1. The benchmarks, tasks, and datasets are described in Section 4 and Appendix C.1, and the evaluation protocol, including the number of random seeds and evaluation episodes, is detailed in Section 4 and Appendix C.2. Implementation details of all baselines, which are built upon publicly available official code, and our hyperparameter tuning protocol are provided in Appendix C.3. The hyperparameters shared across methods and the task-specific BC coefficients are listed in Tables 2 and 3, and the Transformer architecture and chunk length settings of ACH are specified in Appendix C.4. For the VLA experiments, we use publicly released GR00T N1.7 checkpoints, and the benchmark, training protocol, network architecture, and hyperparameters are described in Section 4.1 and Appendix C.5. Finally, per-task results for all experiments are reported in Appendix E.

REFERENCES

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

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

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong,

Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025a.

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. In *Robotics: Science and Systems XXI*, 2025b.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

Steven Bradtke and Michael Duff. Reinforcement learning methods for continuous-time markov decision problems. *Advances in neural information processing systems*, 7, 1994.

Jongseong Chae, Jongeui Park, Yongjae Shin, Gyeongmin Kim, Seungyul Han, and Youngchul Sung. Flow actor-critic for offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

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.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. In *International Conference on Learning Representations*, volume 2026, pp. 131379–131394, 2026.

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

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Anna Harutyunyan, Will Dabney, Diana Borsa, Nicolas Heess, Remi Munos, and Doina Precup. The termination critic. *arXiv preprint arXiv:1902.09996*, 2019.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Ahad Jawaid. Offline rl with hierarchical action chunking. *arXiv preprint arXiv:2607.20834*, 2026.

Michael J Kearns and Satinder Singh. Bias-variance error bounds for temporal difference updates. In *COLT*, pp. 142–147, 2000.

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Martin Klissarov and Doina Precup. Flexible option learning. *Advances in Neural Information Processing Systems*, 34:4632–4646, 2021.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

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

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

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

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

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

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728, 2025b.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20802–20811, June 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Qin-Wen Luo, Ming-Kun Xie, Ye-Wen Wang, and Sheng-Jun Huang. Optimistic critic reconstruction and constrained fine-tuning for general offline-to-online rl. *Advances in Neural Information Processing Systems*, 37:108167–108207, 2024.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518:529–533, 2015.

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

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

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.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Kwanyoung Park, Seohong Park, Youngwoon Lee, and Sergey Levine. Scalable offline model-based rl with action chunks. In *International Conference on Learning Representations*, volume 2026, pp. 9599–9623, 2026a.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. In *International Conference on Machine Learning (ICML)*, 2025b.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *Advances in Neural Information Processing Systems*, 38:8350–8389, 2026b.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

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

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.

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

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

Yongjae Shin, Jeonghye Kim, Whiyoung Jung, Sunghoon Hong, Deunsol Yoon, Youngsoo Jang, Geon-Hyeong Kim, Jongseong Chae, Youngchul Sung, Kanghoon Lee, and Woohyung Lim. Online pre-training for offline-to-online reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

Yongjae Shin, Jongseong Chae, Jongeui Park, and Youngchul Sung. Flow matching with injected noise for offline-to-online reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Gwanwoo Song, Kwanyoung Park, and Youngwoon Lee. Chunk-guided q-learning. In *Forty-third International Conference on Machine Learning*, 2026.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44, 1988.

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

Richard S Sutton, Doina Precup, and Satinder Singh. Intra-option learning about temporally abstract actions. In *ICML*, volume 98, pp. 556–564, 1998b.

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

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023.

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

Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. In *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*, pp. 177–184. IEEE, 2009.

Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. *Advances in neural information processing systems*, 29, 2016.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Shenzhi Wang, Qisen Yang, Jiawei Gao, Matthieu Lin, Hao Chen, Liwei Wu, Ning Jia, Shiji Song, and Gao Huang. Train once, get a family: State-adaptive balances for offline-to-online reinforcement learning. *Advances in Neural Information Processing Systems*, 36:47081–47104, 2023.

Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 31278–31291, 2022.

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

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

Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In *International Conference on Machine Learning*, pp. 40452–40474. PMLR, 2023.

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

Haichao Zhang, Wei Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Jiyao Zhang, Zimu Han, Junhan Wang, Xionghao Wu, Shihong Lin, Jinzhou Li, Hongwei Fan, Ruihai Wu, Dongjiang Li, and Hao Dong. Hipolicy: Hierarchical multi-frequency action chunking for policy learning. In *European Conference on Computer Vision*, pp. 359–376. Springer, 2026.

Yinmin Zhang, Jie Liu, Chuming Li, Yazhe Niu, Yaodong Yang, Yu Liu, and Wanli Ouyang. A perspective of q-value estimation on offline-to-online reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 16908–16916, 2024.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX*, 2023.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *International Conference on Learning Representations*, volume 2025, pp. 32343–32368, 2025.

A LIMITATIONS

Since the proposed method employs a Transformer as the value function, which is relatively heavier than the MLP used in prior methods, training requires comparatively more time. We report this in Appendix D.4. A potential direction for future work is to identify a lighter alternative to the Transformer that still enables multiple chunk value estimation. Additionally, as with existing methods, the proposed method executes the sampled action chunk in an open-loop manner, which may limit its applicability in environments with rapidly changing dynamics. In such cases, replanning during action chunk execution (Sendai et al., 2025; Black et al., 2026) would be necessary, and this constitutes another promising direction for future research.

B ADDITIONAL RELATED WORK

In this section, we discuss the connections between ACH and prior research on temporally extended decision making, including multi-step temporal-difference (TD) learning, the options framework, action repetition, and hierarchical RL.

Multi-step TD learning Multi-step TD methods construct the learning target from the rewards over the next n steps and the value estimate of the state reached after n steps. The choice of n controls a trade-off between the bias of one-step bootstrapping and the variance of Monte Carlo returns (Sutton, 1988; Sutton et al., 1998a; Kearns & Singh, 2000). However, the n-step return becomes biased when computed from off-policy data, since the actions taken within the n steps are not drawn from the policy being evaluated. Prior works correct this bias through importance sampling or truncated importance weights (Precup et al., 2000; Munos et al., 2016), while many practical algorithms simply use uncorrected n-step returns for faster learning (Hessel et al., 2018; Barth-Maron et al., 2018). More recently, several works avoid this bias by conditioning the critic on the executed action sequence (Li et al., 2025a; 2026b; Tian et al., 2026; Nagy et al., 2026). ACH follows this line of work.

From the perspective of multi-step TD learning, the first term of Eq. (7) applies n-step TD policy evaluation to all $h+1$ prefixes of an action chunk simultaneously, where each $Q(s_t, \boldsymbol{a}_{t:t+n})$ is regressed onto the realized n-step return plus the bootstrapped value $\gamma^{n+1}V(s_{t+n+1})$. Since each Q is conditioned on the executed actions, this target avoids the off-policy bias of standard n-step returns. The bootstrapped value V in this target is learned by the second term as the expected Q under the current policy and π_L, analogous to Expected Sarsa (Van Seijen et al., 2009). Combined with the policy improvement step in Eq. (8), Eq. (7) forms an actor-critic scheme that generalizes the chunk-wise TD objective of QC (Li et al., 2026b) to all chunk lengths.

Options and semi-Markov decision processes The options framework formalizes temporally extended actions as options, each consisting of an initiation set, an intra-option policy, and a termination condition (Sutton et al., 1999; Precup, 2000). Decision making over options is then modeled as a semi-Markov decision process (SMDP), in which each decision can span multiple time steps (Bradtke & Duff, 1994; Sutton et al., 1999). Many works learn options and their termination conditions end-to-end (Bacon et al., 2017; Harb et al., 2018; Harutyunyan et al., 2019; Klissarov & Precup, 2021), typically over a small set of discrete options.

From the perspective of the options framework, an action chunk can be viewed as an option that executes a sequence of actions determined at its start for a fixed number of steps, without responding to the states encountered along the way. Options of this kind are known as macro-actions (Vezhnevets et al., 2016). The composition of π_θ and π_L then plays the role of the policy over options. For an option given by the chunk $\boldsymbol{a}_{t:t+n}$ of duration $n+1$, the only remaining randomness is the environment transition, and the SMDP Bellman expectation equation is

$$Q(s_t, \boldsymbol{a}_{t:t+n}) = \mathbb{E}\Big[\textstyle\sum_{k=0}^{n} \gamma^k r_{t+k} + \gamma^{n+1} V(s_{t+n+1})\Big], \tag{10}$$

where V denotes the state value under the policy over options. The first term of Eq. (7) regresses $Q(s_t, \boldsymbol{a}_{t:t+n})$ onto exactly this target for every duration, and the second term corresponds to the SMDP relation between V and Q under the composite policy over options. Thus, the two terms of Eq. (7) jointly instantiate the SMDP Bellman expectation equations for Q and V. Moreover, since every prefix of an executed chunk is itself a valid shorter option, a single trajectory provides valid

targets for all durations at once, which is analogous to intra-option value learning (Sutton et al., 1998b). Unlike prior option learning methods, the options in ACH are not chosen from a predefined set; instead, the policy generates a new continuous action sequence at every state, so the option space is uncountable. Even in this setting, ACH efficiently determines how long to execute each generated sequence by comparing the values of all its prefixes, which the causal Transformer-based value function evaluates in a single forward pass.

Action repetition Action repetition methods study how long an agent should keep executing the same action, which is closely related to selecting the chunk length. Frame skipping, which repeats each action for a fixed number of steps, is a common practice (Mnih et al., 2015), and subsequent works learn the repetition length adaptively (Lakshminarayanan et al., 2017; Sharma et al., 2017; Biedenkapp et al., 2021), select a persistence level for offline RL (Metelli et al., 2020), or decide at each step whether to keep the previous action (Yu et al., 2021). Other works generate multi-step action plans and decide when to replace them, for example by learning when to update the plan (Vezhnevets et al., 2016) or by switching to a newly generated plan when its value is higher than that of the current plan (Zhang et al., 2022). In contrast to these methods, which decide whether to keep or replace the current plan, ACH compares the values of all prefixes of a generated chunk to select how long to execute it.

Hierarchical RL Hierarchical RL decomposes decision making into multiple levels of temporal abstraction (Dayan & Hinton, 1992; Dietterich, 2000). Typically, a high-level policy sets subgoals or latent goals every few steps, and a low-level policy selects primitive actions to reach them (Vezhnevets et al., 2017; Nachum et al., 2018). Such hierarchies have been shown to benefit mainly from improved exploration (Nachum et al., 2019). Other works extract reusable skills from offline data (Ajay et al., 2020; Pertsch et al., 2021), and hierarchical methods that decompose tasks into subgoals achieve strong performance on long-horizon offline goal-conditioned tasks (Park et al., 2023; 2026b). More recently, several works have combined hierarchy with action chunking, for example by pairing a high-level planner that predicts latent subgoals with a low-level critic conditioned on fixed-length action chunks (Jawaid, 2026), or by predicting action chunks at multiple frequencies and balancing them during execution with an entropy-guided mechanism in imitation learning (Zhang et al., 2026). Like ACH, these approaches aim to combine long-horizon structure with fine-grained control, but they either fix the chunk length or select it with an uncertainty-based heuristic rather than value estimation. In contrast, ACH keeps a single-level policy and uses the values of all chunk prefixes to select the chunk length at each state, adapting it during both training and inference.

C Experimental Details

In Section 4, we validate the efficacy of the proposed method through comprehensive evaluations across 34 distinct tasks. This section provides the specific details of the experiments used to obtain these results.

C.1 Benchmarks and Tasks

We conduct experiments on 30 tasks from OGBench (Park et al., 2025a) and 4 tasks from robomimic (Mandlekar et al., 2021). Following standard protocols in offline-to-online RL (Park et al., 2025b; Shin et al., 2026), we utilize the single-task variant across six environments: `antmaze-giant`, `antsoccer-arena`, `cube-quadruple`, `cube-triple`, `humanoidmaze-medium`, `puzzle-4x4`. Each environment includes five distinct tasks.

- Antmaze and Antsoccer: The objective is to control a robot with an 8-dimensional action space to reach a designated goal.
- Cube: A robotic arm with a 5-dimensional action space performs various pick-and-place tasks.
- Humanoidmaze: This task involves navigating a humanoid robot in a 21-dimensional action space to a goal.
- Puzzle: A robotic arm with a 5-dimensional action space presses buttons in a specific manner to solve the puzzle.

For the datasets, we employ the navigate dataset for antmaze, antsoccer, and humanoidmaze, which consist of 1M, 1M, and 2M transitions, respectively. The play dataset is utilized for cube and puzzle, containing 3M transitions for cube-triple, 10M transitions for cube-quadruple, and 1M transitions for puzzle.

We further evaluate the method on four specific tasks within the robomimic benchmark. All experiments in this domain utilize the multi-human dataset, which consists of 300 successful trajectories for each task.

- lift: A single robotic arm with a 7-dimensional action space lifts a small cube.
- can: A single robotic arm with a 7-dimensional action space picks up a can and places it into a target bin.
- square: A single robotic arm with a 7-dimensional action space places a square nut onto a rod.
- transport: Two robotic arms with a combined 14-dimensional action space perform a collaborative maneuver. One arm lifts a hammer and passes it to the other arm, which then places the object into a target bin.

C.2 Evaluation

We evaluate all baselines every 100K environment steps. Each evaluation measures the success rate over 50 episodes. Regarding the training budget, 1M offline steps are sufficient for offline pre-training to converge on datasets with 1M or more transitions. The 1M online steps match the scale of the offline data and are long enough for each method to meaningfully surpass the performance limited by the dataset. This choice is also consistent with prior studies (Park et al., 2025b; Li et al., 2026b).

C.3 Baselines

For the comparative analysis, we implement the baselines by leveraging the experimental settings of FINO (Shin et al., 2026) and QC (Li et al., 2026b). Specifically, the implementations of RLPD, ReBRAC, FQL, and FINO are based on the paper and official code of FINO[1], while the BFN and QC follow the paper and official code of QC[2]. In accordance with previous studies (Park et al., 2025b;

[1] https://github.com/CTID282/FINO

[2] https://github.com/ColinQiyangLi/qc

Shin et al., 2026), RLPD is configured with an update-to-data ratio of 1 and two value functions. The number of sampled actions for BFN is fixed at 4. The BC coefficients α for ReBRAC, FQL, FINO, and QC are summarized in Table 3, and the common hyperparameters utilized across all baselines are presented in Table 2.

For hyperparameter tuning, we follow two rules. First, when the original paper of a baseline covers the corresponding task, we use its reported hyperparameters. Second, when a task is not covered, we construct three candidate values for the main hyperparameter by referring to the values used for similar tasks and report the best result. In particular, since the original papers of RLPD, ReBRAC, and BFN do not cover OGBench (Park et al., 2025a), we refer to the settings of FQL (Park et al., 2025b) and QC (Li et al., 2026b). The selection criterion is the final performance after online learning, averaged over seeds. We apply the same protocol to ACH for a fair comparison.

Table 2: Shared Hyperparameters

Hyperparameter	Value
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2014)
Batch size	256
Hidden dimensions	[512, 512, 512, 512]
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Target network update rate	0.005
Discount factor γ	0.99 (default), 0.995 (`antmaze, humanoidmaze, antsoccer`)
Flow steps	10

Table 3: Task-specific BC coefficient α for each method. ReBRAC follows the loss normalization of its original implementation, so its coefficients are not directly comparable in scale to those of the other methods.

Task	ReBRAC	FQL	FINO	QC	ACH
antmaze-giant-navigate	0.003	10	10	30	30
antsoccer-arena-navigate	0.01	30	30	30	30
cube-quadruple-play-10M	0.1	300	300	100	300
cube-triple-play	0.1	300	300	100	300
humanoidmaze-medium-navigate	0.01	100	100	100	100
puzzle-4x4-play	0.3	1000	1000	300	300
lift	0.03	10000	10000	10000	10000
can	0.03	10000	10000	10000	10000
square	0.1	10000	10000	10000	10000
transport	0.1	10000	10000	10000	1000

C.4 ACH

The proposed method, ACH, is built upon QC as the backbone model and incorporates a transformer architecture into the value function. This transformer configuration consists of an embedding dimension of 128, 4 heads, and 2 attention layers. As with other baselines, ACH utilizes an environment-specific BC coefficient α, which is provided in Table 3. We set the maximum chunk length to 10 for all environments except `humanoidmaze`; for `humanoidmaze`, where the action dimension is 21 and relatively large, learning with a maximum chunk length of 10 becomes difficult for the policy, so we reduce it to 5. For adaptive chunk length sampling at inference, the action chunk with the highest action value is selected to ensure consistent evaluation, thereby rendering the policy deterministic. The pseudo-code for ACH is presented in Algorithm 1.

C.5 VLA Experiments

Benchmark We evaluate on SimplerEnv (Li et al., 2025b) with two embodiments. For the WidowX robot, we use seven tasks: `spoon_on_towel`, `carrot_on_plate`,

Algorithm 1 ACH: Adaptive Action CHunking for Offline-to-Online RL

Inputs: offline dataset D, behavior policy β_ψ, one-step policy π_θ, transformer-based value function Q_ϕ, horizon length h, action chunk buffer $\mathcal{B} \leftarrow \emptyset$
while in offline learning **do**
  Update ψ, θ, ϕ based on Equations 1, 8, 7.
end while
while in online learning **do**
  if $\mathcal{B}$ is empty **then**
    Sample full-length action chunk $\boldsymbol{a}_{t:t+h} \sim \pi_\theta(\cdot|s_t)$
    Compute values $\{Q_\phi(s_t, \boldsymbol{a}_{t:t}), \ldots, Q_\phi(s_t, \boldsymbol{a}_{t:t+h})\}$
    Calculate sampling probability $\pi_L(\ell)$ from Equation 9
    Select $\ell \sim \pi_L(\ell)$ {chunk length $\ell + 1$}
    Store $\{a_t, a_{t+1}, \ldots, a_{t+\ell}\}$ in $\mathcal{B}$
  end if
  Get a_t from $\mathcal{B}$ and execute a_t
  Update ψ, θ, ϕ based on Equations 1, 8, 7.
end while

`put_eggplant_in_basket`, `stack_cube`, `put_eggplant_in_sink`, `close_drawer`, and `open_drawer`. For the Google Robot, we use six tasks under the visual matching setting: `pick_coke_can`, `pick_object`, `move_near`, `open_drawer`, `close_drawer`, and `place_in_closed_drawer`.

Base policy We use GR00T N1.7 (Bjorck et al., 2025) as the base VLA, with the publicly released checkpoints for the WidowX[3] and Google Robot[4] embodiments. The base VLA predicts action chunks of 8 steps by default, and we use this chunk length for all methods, which also serves as the maximum chunk length for the adaptive methods. The VLA is kept frozen throughout training.

QC and ACH For QC and ACH, we add a one-step policy on top of the frozen VLA. The one-step policy takes the observation and the action chunk produced by the VLA as input and outputs a new action chunk. Observations consist of an image and the proprioceptive state. The image is processed by a four-layer convolutional network with 32, 64, 128, and 256 channels and a stride of 2, and its output is combined with the proprioceptive state into a 256-dimensional embedding. This encoder is shared by the policy and the value function, which use the same network architecture. QC is trained with Eqs. (2) and (3), and ACH is trained with Eqs. (7) and (8) and selects the chunk length with Eq. (9) using $\beta = 1$. Both methods use a BC coefficient of $\alpha = 1000$, a batch size of 128, and the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 3×10^{-4}.

We train a separate model for each task. For offline training, we collect 1,000 episodes per task with the base policy and train for 100K gradient steps. Online training then proceeds for 100K environment steps with an update-to-data ratio of 5, where each batch is sampled symmetrically from the offline dataset and the online replay buffer following RLPD (Ball et al., 2023). We use a discount factor of $\gamma = 0.99$, a target network update rate of $\tau = 0.005$, and an ensemble of two Q-functions.

AAC and QC+AAC For AAC (Liang et al., 2026), we use $N = 20$ samples and select its movement floor hyperparameter for each embodiment from $\{0, 0.05, 0.1, 0.15, 0.3, 0.6\}$. For QC+AAC, we sample 20 action chunks from the one-step policy of QC and determine the chunk length based on the rule of AAC.

[3] `https://huggingface.co/nvidia/GR00T-N1.7-SimplerEnv-Bridge`
[4] `https://huggingface.co/nvidia/GR00T-N1.7-SimplerEnv-Fractal`

D Additional Experiments

D.1 Adaptive Chunk Length in Manipulation

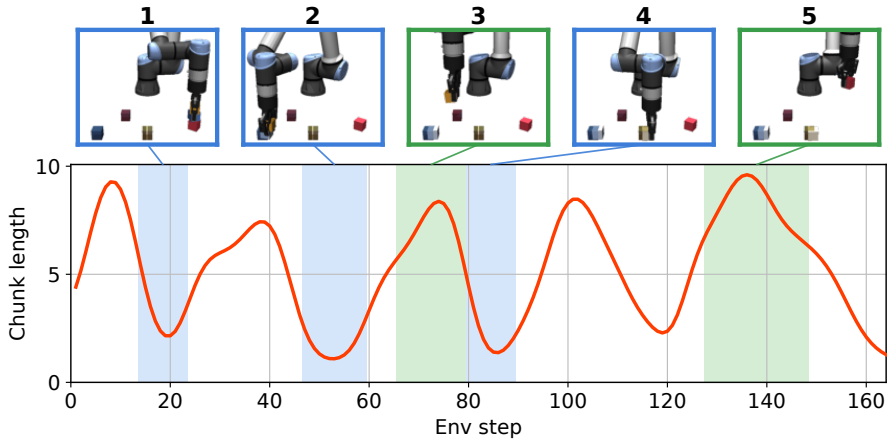


Figure 7: Chunk lengths selected by ACH over environment timesteps during a `cube-triple-task3` episode. Shaded regions correspond to the subfigures above. ACH selects shorter chunks (blue) when grasping or placing a cube and longer chunks (green) while moving between cubes.

To examine whether the behavior in Figure 1 extends beyond navigation, we analyze the chunk lengths selected by ACH during an episode of `cube-triple-task3`, where the agent must pick and place multiple cubes. As shown in Figure 7, ACH repeatedly alternates between short and long chunks over the course of the episode. When the gripper interacts with a cube, such as grasping or placing it, ACH reduces the chunk length to one or two and re-selects its actions from the latest observations during precise manipulation. In contrast, when the gripper simply moves toward the next cube or carries a grasped cube to its target location, ACH selects long chunks, up to the maximum length, as these motions do not require precise control. These results indicate that ACH adapts the chunk length to the precision required at each state not only in navigation but also in manipulation.

D.2 Action Values across Chunk Lengths

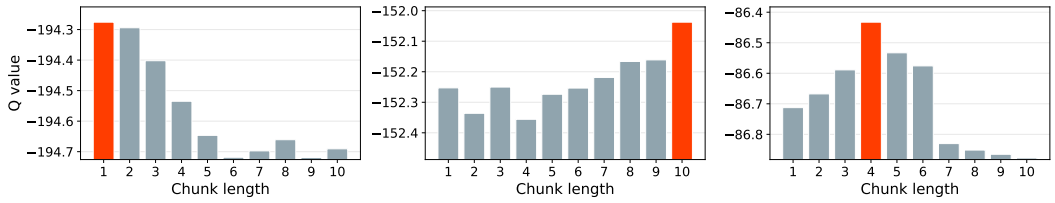


Figure 8: Action value as a function of chunk length (`antmaze-giant-navigate` with maximum chunk length set to 10): (left) fast-changing states, (middle) strongly correlated states over time, and (right) trade-off situation, which correspond to Subfigures 1, 3, and 5 in Figure 1, respectively. The minimal value difference stems from the reward design of the environment: a constant reward of -1 per step, complemented by a sparse reward (1 or 0).

Figure 8 shows the action values of chunks with different lengths at three representative states in `antmaze-giant-navigate`, estimated by the Transformer-based value function trained with ACH (Section 3.2). In fast-changing states (left), where the uncertainty of future states dominates, the value is highest at a chunk length of one and decreases as the chunk becomes longer. In states that are strongly correlated over time (middle), where policy stochasticity jittering dominates, the value is highest at the maximum chunk length. When both factors are present (right), the value peaks at an intermediate chunk length. These patterns confirm that the optimal chunk length varies across states and is determined by the trade-off discussed in Section 3.1.

D.3 Comparison with Fixed-Length Baselines

Table 4: Comparison of ACH with fixed chunk length baselines on OGBench. Hn denotes the backbone trained and executed with a fixed chunk length of n, where H1 and H5 correspond to FQL and QC, respectively. All methods are evaluated after 1M steps of offline learning followed by 1M steps of online learning. We report the mean and 95% confidence intervals across 10 seeds. Results for each environment are averaged over five tasks.

Task	H1 (FQL)	H3	H5 (QC)	H10	ACH
antmaze-giant-navigate	$10_{\pm 5} \rightarrow \mathbf{74}_{\pm 7}$	$0_{\pm 0} \rightarrow \mathbf{71}_{\pm 7}$	$0_{\pm 0} \rightarrow 47_{\pm 8}$	$0_{\pm 0} \rightarrow 20_{\pm 6}$	$1_{\pm 1} \rightarrow \mathbf{72}_{\pm 2}$
antsoccer-arena-navigate	$35_{\pm 2} \rightarrow 70_{\pm 5}$	$50_{\pm 3} \rightarrow \mathbf{82}_{\pm 2}$	$10_{\pm 2} \rightarrow 67_{\pm 2}$	$0_{\pm 0} \rightarrow 12_{\pm 2}$	$0_{\pm 0} \rightarrow \mathbf{84}_{\pm 1}$
cube-quadruple-play-10M	$0_{\pm 0} \rightarrow 2_{\pm 1}$	$1_{\pm 0} \rightarrow 8_{\pm 5}$	$1_{\pm 1} \rightarrow 33_{\pm 3}$	$1_{\pm 0} \rightarrow 41_{\pm 3}$	$0_{\pm 0} \rightarrow \mathbf{52}_{\pm 6}$
cube-triple-play	$2_{\pm 1} \rightarrow 19_{\pm 2}$	$5_{\pm 1} \rightarrow 35_{\pm 3}$	$4_{\pm 2} \rightarrow 57_{\pm 4}$	$3_{\pm 1} \rightarrow \mathbf{70}_{\pm 4}$	$1_{\pm 1} \rightarrow \mathbf{67}_{\pm 3}$
humanoidmaze-medium-navigate	$36_{\pm 3} \rightarrow 56_{\pm 3}$	$53_{\pm 3} \rightarrow 61_{\pm 7}$	$15_{\pm 2} \rightarrow 32_{\pm 4}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$45_{\pm 2} \rightarrow \mathbf{82}_{\pm 1}$
puzzle-4x4-play	$14_{\pm 2} \rightarrow 54_{\pm 6}$	$20_{\pm 3} \rightarrow 50_{\pm 8}$	$11_{\pm 2} \rightarrow 38_{\pm 4}$	$0_{\pm 0} \rightarrow 20_{\pm 5}$	$13_{\pm 3} \rightarrow \mathbf{90}_{\pm 8}$

A natural question is whether the gains of ACH could be obtained simply by tuning a fixed chunk length for each environment. To answer this question, we evaluate fixed-length baselines with four chunk lengths: 1, 3, 5, and 10. As summarized in Table 4, their performance fluctuates significantly across environments depending on the chunk length. Although certain fixed lengths perform comparably to ACH in some environments, they degrade substantially in others. A fixed chunk length performs well only when that length is close to optimal for most states in an environment; otherwise, no single fixed length can cover the whole task. For example, on `puzzle-4x4`, ACH achieves a success rate of 90, whereas the best fixed length reaches only 54, a gap that cannot be explained by a task-level choice of the chunk length. In contrast, ACH achieves strong performance across all environments without tuning the chunk length for each environment. Since no single chunk length suits all states, these results highlight the necessity of adapting the chunk length to each state.

D.4 Computational Cost

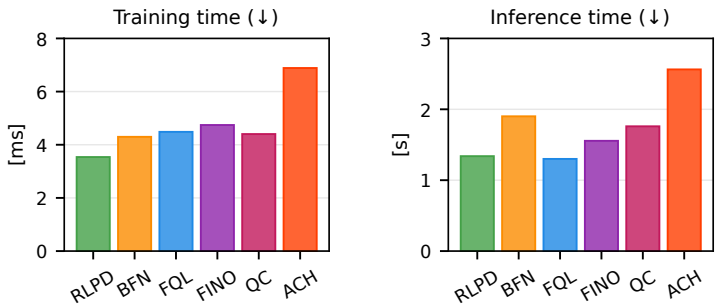


Figure 9: Comparison of computational costs between the proposed method and baselines on the `antmaze-giant-navigate` task. Training time represents the duration per step during online learning, and inference time denotes the time required for a single episode rollout. All measurements are conducted on a single NVIDIA RTX 3090 GPU.

In Figure 9, we provide the computational costs of the proposed method and the baselines during both the training and inference phases. Since ACH utilizes a relatively heavy transformer architecture and requires value estimation across all chunk lengths during inference, the computational time for ACH is higher than that of other baselines. However, the increase is not prohibitive, as it remains 1.3 to 1.4 times the processing time of the baselines. Considering the significant performance gains achieved by our method, this additional overhead is well within a justifiable limit.

D.5 Evolution of Chunk Length

In Off2On RL, the primary role of online learning is to enable the agent to refine its behavior through direct environment interaction. While methods with a fixed chunk length are limited to

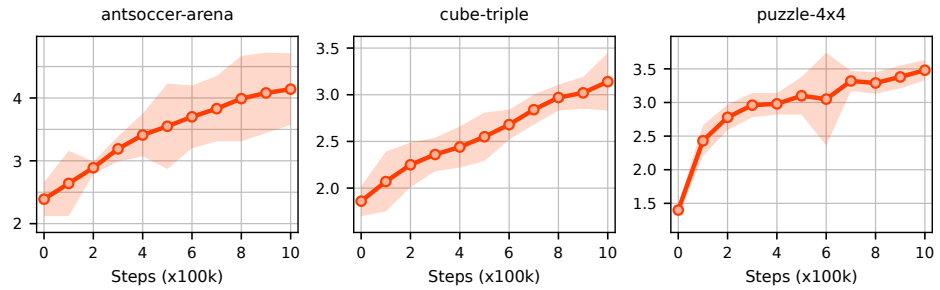


Figure 10: Evolution of average chunk length during the online learning phase on `antsoccer-arena`, `cube-triple`, and `puzzle-4x4`. The shaded region denotes one standard deviation across seeds.

adjusting individual actions, our approach facilitates both behavioral correction and chunk length optimization. To investigate this, we analyze the evolution of the average chunk length during online learning. As illustrated in Figure 10, ACH exhibits a clear tendency to gradually increase the average chunk length as training progresses, and this trend holds consistently across both navigation (`antsoccer-arena`) and manipulation (`cube-triple` and `puzzle-4x4`) environments. This trend is attributed to the stitching of trajectories within the value function. As the policy optimizes based on this value function, it becomes capable of executing behaviors as single, extended sequences that previously required multiple short segments. These results indicate that our approach goes beyond merely refining individual actions; by leveraging value-based trajectory stitching to progressively increase chunk lengths, it ultimately facilitates sample-efficient learning.

D.6 Ablation Study on Robomimic

Table 5: Ablation results on the `square` and `transport` tasks of robomimic, showing the performance immediately after 1M steps of offline learning and 1M steps of online learning. Results represent the mean ± 95% confidence interval.

Task	Uniform	Greedy	ACH
square	$44_{\pm 4} \rightarrow 74_{\pm 3}$	$40_{\pm 6} \rightarrow 80_{\pm 5}$	$42_{\pm 5} \rightarrow \mathbf{94}_{\pm 2}$
transport	$0_{\pm 0} \rightarrow 11_{\pm 1}$	$0_{\pm 0} \rightarrow 19_{\pm 2}$	$0_{\pm 0} \rightarrow \mathbf{39}_{\pm 4}$

To examine whether the core components of ACH remain effective beyond OGBench, we additionally conduct the `Uniform` and `Greedy` ablations from Section 4.2 on the two most challenging robomimic tasks, `square` and `transport`. As shown in Table 5, the results are consistent with those in Figure 5. `Uniform` suffers from the misalignment between the training objective and the chunk length selection policy, and `Greedy` fails to explore sufficiently, which is particularly evident on the difficult `transport` task. These results confirm that the core components of ACH are jointly necessary on robomimic as well.

E Complete Results

E.1 Main Experiments

We present the per-task results corresponding to Table 1 in Table 7, and the corresponding learning curves in Figure 12. For the ablation study shown in Figure 5, we report the results across all tasks in Table 8.

E.2 VLA Experiments

We present the per-task results corresponding to Figure 4 in Table 6.

Table 6: Complete results for Figure 4, showing the success rates (%) of each method on SimplerEnv. Results represent the mean ± 95% confidence interval calculated across 4 seeds, with 200 evaluation episodes per task.

Robot	Task	Base	QC	AAC	QC+AAC	ACH
WidowX	spoon_on_towel	81.5	77.9±2.0	80.0±4.7	79.6±3.6	**91.1**±4.7
	carrot_on_plate	68.5	68.8±12.5	62.6±4.9	67.0±17.0	**86.1**±10.2
	put_eggplant_in_basket	25.0	26.1±11.1	52.0±3.2	49.2±13.8	**67.6**±6.5
	stack_cube	41.0	39.6±2.9	34.9±3.3	35.0±5.8	**46.9**±6.6
	put_eggplant_in_sink	2.0	1.0±0.6	0.8±0.5	1.0±1.3	**5.3**±2.1
	close_drawer	**98.0**	**100.0**±0.0	**98.0**±0.9	**99.6**±0.8	**100.0**±0.0
	open_drawer	**100.0**	**99.9**±0.4	**99.6**±0.4	**100.0**±0.0	**100.0**±0.0
	Average	59.4	59.0±2.4	61.1±1.2	61.6±3.3	**71.0**±2.1
Google Robot	pick_coke_can	64.0	59.7±5.3	90.1±1.5	92.8±2.9	**98.2**±3.8
	pick_object	38.5	42.4±4.6	67.9±4.7	72.8±3.5	**87.0**±4.0
	move_near	54.0	54.2±2.9	80.7±0.8	86.1±3.9	**94.0**±4.8
	open_drawer	35.0	39.5±7.0	42.5±8.4	43.7±5.6	**55.0**±7.0
	close_drawer	69.5	**78.9**±2.3	65.6±6.0	68.8±3.1	**80.5**±1.7
	place_in_closed_drawer	6.0	7.9±3.6	8.7±1.4	7.9±2.9	**16.6**±2.1
	Average	44.5	47.1±1.9	59.2±1.9	62.0±1.5	**71.9**±1.7

E.3 Behavior Analysis

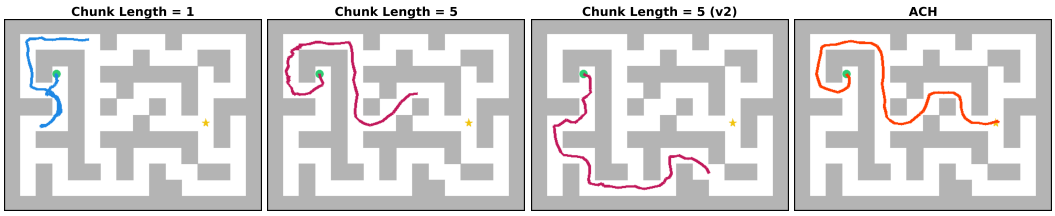


Figure 11: Behavioral comparison of ACH and fixed-length baselines in the antmaze-giant environment. Starting from a shared initial state, all algorithms execute the same number of total action steps. Agent trajectories are constructed by connecting sequential positions at each timestep. The initial state and goal state are marked by a green circle and a yellow star, respectively.

In Figure 6, we validate the efficacy of the proposed method by analyzing the behaviors required to solve tasks in a practical environment, comparing our approach with fixed-length baselines. In this section, Figure 11 presents the full layout of the environment and an additional behavioral pattern observed when the chunk length is fixed at 5 (v2). The trajectory for this baseline reveals a significant detour instead of following the shortest path to the goal, which eventually leads to a failure to reach the goal within the time limit. This outcome stems from the rigid nature of the fixed-length constraint. If the agent deviates from the optimal path during the early stages, the lack of flexibility prevents

it from recovering and forces it to follow a suboptimal route. These findings highlight the inherent limitation of conventional methods that rely on a static chunking strategy.

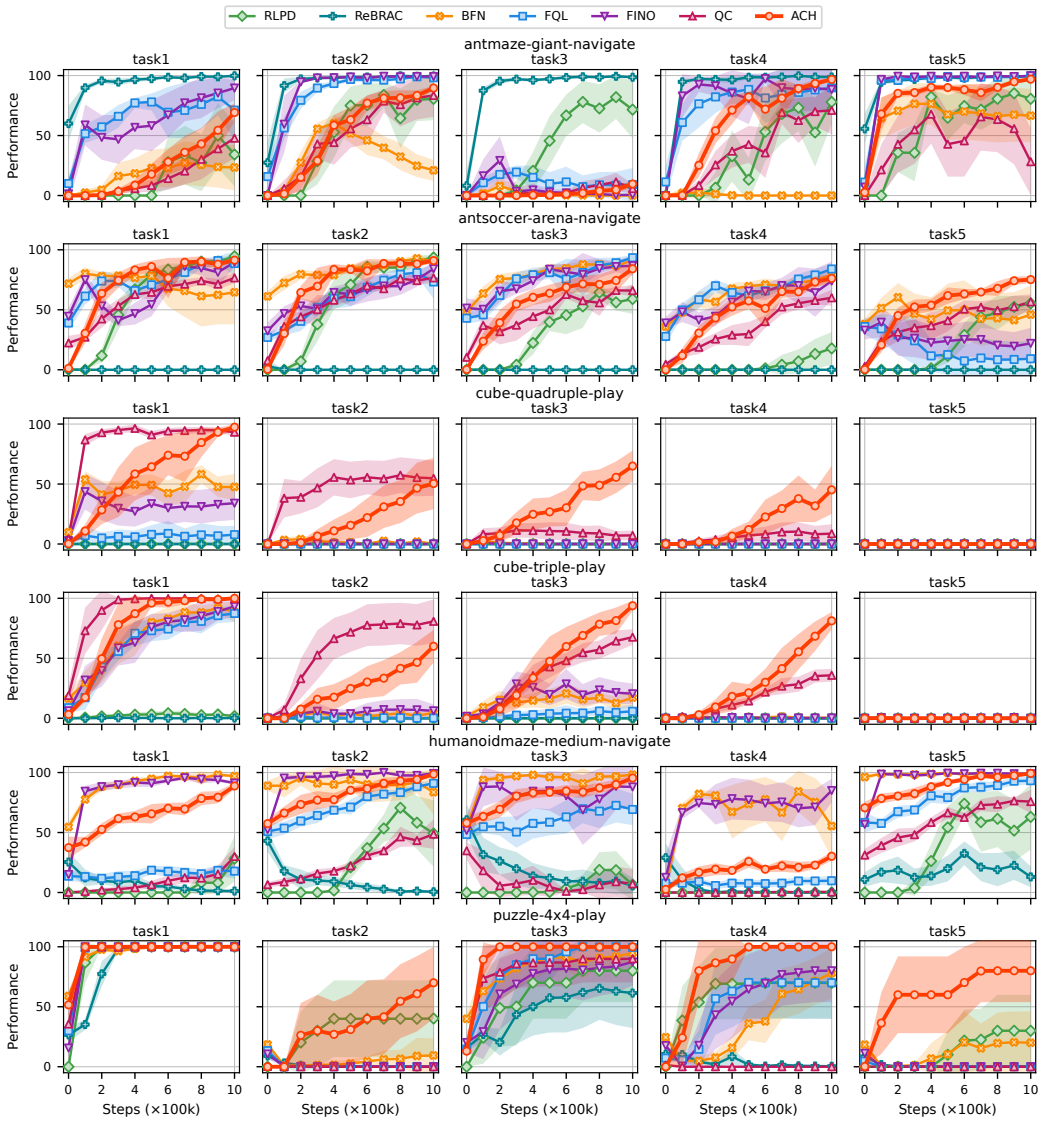


Figure 12: Learning curves during online learning for all tasks in OGBench. Solid lines represent the mean across 10 seeds, and shaded regions denote the 95% confidence interval.

Table 7: Complete results for Table 1 and Figure 3, showing the performance immediately after 1M steps of offline learning and 1M steps of online learning. Results represent the mean ± 95% confidence interval calculated across 10 seeds.

Environment	RLPD	ReBRAC	BFN	FQL	FINO	QC	ACH
OGBench antmaze-giant-navigate-task1	$0_{\pm 0} \rightarrow 34_{\pm 26}$	$60_{\pm 15} \rightarrow \mathbf{100}_{\pm 0}$	$0_{\pm 0} \rightarrow 23_{\pm 19}$	$10_{\pm 6} \rightarrow 71_{\pm 25}$	$1_{\pm 9} \rightarrow 90_{\pm 10}$	$0_{\pm 0} \rightarrow 48_{\pm 17}$	$0_{\pm 0} \rightarrow 69_{\pm 8}$
OGBench antmaze-giant-navigate-task2	$0_{\pm 0} \rightarrow 81_{\pm 18}$	$27_{\pm 18} \rightarrow \mathbf{100}_{\pm 1}$	$0_{\pm 0} \rightarrow 21_{\pm 9}$	$16_{\pm 12} \rightarrow \mathbf{98}_{\pm 1}$	$0_{\pm 0} \rightarrow \mathbf{99}_{\pm 1}$	$0_{\pm 0} \rightarrow 83_{\pm 18}$	$0_{\pm 0} \rightarrow 90_{\pm 2}$
OGBench antmaze-giant-navigate-task3	$0_{\pm 0} \rightarrow 72_{\pm 24}$	$8_{\pm 12} \rightarrow \mathbf{99}_{\pm 1}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 9_{\pm 14}$	$0_{\pm 0} \rightarrow 1_{\pm 1}$	$0_{\pm 0} \rightarrow 6_{\pm 4}$	$0_{\pm 0} \rightarrow 10_{\pm 3}$
OGBench antmaze-giant-navigate-task4	$0_{\pm 0} \rightarrow 78_{\pm 18}$	$0_{\pm 0} \rightarrow \mathbf{99}_{\pm 1}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$11_{\pm 14} \rightarrow 90_{\pm 16}$	$0_{\pm 0} \rightarrow 88_{\pm 19}$	$0_{\pm 0} \rightarrow 71_{\pm 20}$	$0_{\pm 0} \rightarrow \mathbf{97}_{\pm 1}$
OGBench antmaze-giant-navigate-task5	$0_{\pm 0} \rightarrow 81_{\pm 19}$	$56_{\pm 13} \rightarrow \mathbf{100}_{\pm 1}$	$2_{\pm 2} \rightarrow 67_{\pm 22}$	$11_{\pm 13} \rightarrow \mathbf{100}_{\pm 0}$	$7_{\pm 8} \rightarrow \mathbf{100}_{\pm 0}$	$0_{\pm 0} \rightarrow 28_{\pm 28}$	$3_{\pm 5} \rightarrow \mathbf{97}_{\pm 1}$
OGBench antsoccer-arena-navigate-task1	$0_{\pm 0} \rightarrow \mathbf{95}_{\pm 2}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$72_{\pm 5} \rightarrow 65_{\pm 27}$	$39_{\pm 6} \rightarrow 89_{\pm 10}$	$45_{\pm 7} \rightarrow 90_{\pm 4}$	$22_{\pm 6} \rightarrow 77_{\pm 4}$	$1_{\pm 1} \rightarrow \mathbf{91}_{\pm 2}$
OGBench antsoccer-arena-navigate-task2	$0_{\pm 0} \rightarrow \mathbf{94}_{\pm 2}$	$3_{\pm 5} \rightarrow 0_{\pm 0}$	$61_{\pm 4} \rightarrow \mathbf{91}_{\pm 2}$	$27_{\pm 4} \rightarrow 73_{\pm 14}$	$32_{\pm 7} \rightarrow 84_{\pm 6}$	$8_{\pm 2} \rightarrow 77_{\pm 4}$	$0_{\pm 1} \rightarrow \mathbf{91}_{\pm 2}$
OGBench antsoccer-arena-navigate-task3	$0_{\pm 0} \rightarrow 59_{\pm 12}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$48_{\pm 3} \rightarrow \mathbf{91}_{\pm 3}$	$43_{\pm 5} \rightarrow \mathbf{93}_{\pm 2}$	$51_{\pm 8} \rightarrow 87_{\pm 9}$	$10_{\pm 3} \rightarrow 66_{\pm 5}$	$0_{\pm 0} \rightarrow 84_{\pm 5}$
OGBench antsoccer-arena-navigate-task4	$0_{\pm 0} \rightarrow 18_{\pm 14}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$36_{\pm 6} \rightarrow 79_{\pm 5}$	$28_{\pm 6} \rightarrow \mathbf{84}_{\pm 4}$	$39_{\pm 5} \rightarrow 74_{\pm 16}$	$5_{\pm 2} \rightarrow 60_{\pm 7}$	$0_{\pm 0} \rightarrow 76_{\pm 3}$
OGBench antsoccer-arena-navigate-task5	$0_{\pm 0} \rightarrow 55_{\pm 6}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$38_{\pm 5} \rightarrow 46_{\pm 24}$	$36_{\pm 7} \rightarrow 9_{\pm 15}$	$33_{\pm 8} \rightarrow 22_{\pm 13}$	$3_{\pm 2} \rightarrow 57_{\pm 6}$	$0_{\pm 0} \rightarrow \mathbf{75}_{\pm 3}$
OGBench cube-quadruple-play-task1-10M	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$10_{\pm 5} \rightarrow 48_{\pm 11}$	$1_{\pm 1} \rightarrow 8_{\pm 7}$	$3_{\pm 2} \rightarrow 34_{\pm 14}$	$4_{\pm 3} \rightarrow \mathbf{93}_{\pm 3}$	$0_{\pm 0} \rightarrow \mathbf{97}_{\pm 2}$
OGBench cube-quadruple-play-task2-10M	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow \mathbf{55}_{\pm 15}$	$0_{\pm 0} \rightarrow 50_{\pm 21}$
OGBench cube-quadruple-play-task3-10M	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 7_{\pm 4}$	$0_{\pm 0} \rightarrow \mathbf{65}_{\pm 13}$
OGBench cube-quadruple-play-task4-10M	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 9_{\pm 8}$	$0_{\pm 0} \rightarrow \mathbf{45}_{\pm 20}$
OGBench cube-quadruple-play-task5-10M	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$
OGBench cube-triple-play-task1	$0_{\pm 0} \rightarrow 2_{\pm 1}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$16_{\pm 5} \rightarrow 89_{\pm 9}$	$9_{\pm 4} \rightarrow 87_{\pm 7}$	$6_{\pm 3} \rightarrow 93_{\pm 9}$	$19_{\pm 8} \rightarrow \mathbf{100}_{\pm 1}$	$3_{\pm 3} \rightarrow \mathbf{100}_{\pm 0}$
OGBench cube-triple-play-task2	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 1} \rightarrow 5_{\pm 4}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 6_{\pm 5}$	$0_{\pm 0} \rightarrow \mathbf{81}_{\pm 18}$	$0_{\pm 0} \rightarrow 60_{\pm 14}$
OGBench cube-triple-play-task3	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$2_{\pm 2} \rightarrow 17_{\pm 8}$	$0_{\pm 0} \rightarrow 6_{\pm 6}$	$2_{\pm 2} \rightarrow 20_{\pm 9}$	$0_{\pm 0} \rightarrow 68_{\pm 7}$	$0_{\pm 0} \rightarrow \mathbf{94}_{\pm 4}$
OGBench cube-triple-play-task4	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 1}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 36_{\pm 5}$	$0_{\pm 0} \rightarrow \mathbf{81}_{\pm 7}$
OGBench cube-triple-play-task5	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$
OGBench humanoidmaze-medium-navigate-task1	$0_{\pm 0} \rightarrow 28_{\pm 19}$	$25_{\pm 9} \rightarrow 1_{\pm 2}$	$55_{\pm 5} \rightarrow \mathbf{97}_{\pm 2}$	$14_{\pm 5} \rightarrow 18_{\pm 6}$	$15_{\pm 4} \rightarrow 91_{\pm 1}$	$0_{\pm 0} \rightarrow 30_{\pm 8}$	$37_{\pm 9} \rightarrow 89_{\pm 3}$
OGBench humanoidmaze-medium-navigate-task2	$0_{\pm 0} \rightarrow 49_{\pm 28}$	$43_{\pm 9} \rightarrow 1_{\pm 1}$	$89_{\pm 4} \rightarrow 84_{\pm 19}$	$51_{\pm 6} \rightarrow 91_{\pm 2}$	$52_{\pm 2} \rightarrow \mathbf{99}_{\pm 1}$	$7_{\pm 2} \rightarrow 49_{\pm 12}$	$57_{\pm 4} \rightarrow \mathbf{99}_{\pm 1}$
OGBench humanoidmaze-medium-navigate-task3	$0_{\pm 0} \rightarrow 7_{\pm 13}$	$61_{\pm 13} \rightarrow 6_{\pm 7}$	$54_{\pm 11} \rightarrow \mathbf{98}_{\pm 2}$	$48_{\pm 5} \rightarrow 69_{\pm 13}$	$52_{\pm 12} \rightarrow 88_{\pm 18}$	$35_{\pm 7} \rightarrow 7_{\pm 12}$	$58_{\pm 4} \rightarrow \mathbf{95}_{\pm 2}$
OGBench humanoidmaze-medium-navigate-task4	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$29_{\pm 12} \rightarrow 0_{\pm 0}$	$4_{\pm 7} \rightarrow 55_{\pm 26}$	$8_{\pm 3} \rightarrow 10_{\pm 4}$	$13_{\pm 20} \rightarrow \mathbf{85}_{\pm 10}$	$0_{\pm 0} \rightarrow 0_{\pm 1}$	$3_{\pm 2} \rightarrow 30_{\pm 4}$
OGBench humanoidmaze-medium-navigate-task5	$0_{\pm 0} \rightarrow 63_{\pm 27}$	$11_{\pm 6} \rightarrow 13_{\pm 9}$	$96_{\pm 3} \rightarrow \mathbf{99}_{\pm 1}$	$58_{\pm 8} \rightarrow 93_{\pm 4}$	$57_{\pm 1} \rightarrow \mathbf{99}_{\pm 1}$	$31_{\pm 5} \rightarrow 76_{\pm 10}$	$71_{\pm 5} \rightarrow \mathbf{99}_{\pm 1}$
OGBench puzzle-4x4-play-task1	$0_{\pm 0} \rightarrow \mathbf{100}_{\pm 0}$	$27_{\pm 6} \rightarrow \mathbf{100}_{\pm 0}$	$59_{\pm 4} \rightarrow \mathbf{100}_{\pm 0}$	$30_{\pm 6} \rightarrow \mathbf{100}_{\pm 0}$	$16_{\pm 9} \rightarrow \mathbf{100}_{\pm 0}$	$36_{\pm 8} \rightarrow \mathbf{100}_{\pm 0}$	$52_{\pm 11} \rightarrow \mathbf{100}_{\pm 0}$
OGBench puzzle-4x4-play-task2	$0_{\pm 0} \rightarrow 40_{\pm 32}$	$9_{\pm 2} \rightarrow 0_{\pm 0}$	$19_{\pm 2} \rightarrow 9_{\pm 14}$	$13_{\pm 3} \rightarrow 0_{\pm 0}$	$11_{\pm 4} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow \mathbf{70}_{\pm 30}$
OGBench puzzle-4x4-play-task3	$0_{\pm 0} \rightarrow 80_{\pm 26}$	$16_{\pm 3} \rightarrow 61_{\pm 29}$	$40_{\pm 5} \rightarrow \mathbf{95}_{\pm 5}$	$15_{\pm 3} \rightarrow \mathbf{99}_{\pm 1}$	$20_{\pm 8} \rightarrow 87_{\pm 20}$	$16_{\pm 4} \rightarrow 90_{\pm 20}$	$13_{\pm 7} \rightarrow \mathbf{100}_{\pm 0}$
OGBench puzzle-4x4-play-task4	$0_{\pm 0} \rightarrow 70_{\pm 30}$	$11_{\pm 2} \rightarrow 0_{\pm 1}$	$25_{\pm 3} \rightarrow 78_{\pm 21}$	$7_{\pm 2} \rightarrow 70_{\pm 30}$	$18_{\pm 5} \rightarrow 80_{\pm 26}$	$1_{\pm 1} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow \mathbf{100}_{\pm 0}$
OGBench puzzle-4x4-play-task5	$0_{\pm 0} \rightarrow 30_{\pm 30}$	$9_{\pm 3} \rightarrow 0_{\pm 0}$	$18_{\pm 5} \rightarrow 20_{\pm 26}$	$5_{\pm 2} \rightarrow 0_{\pm 0}$	$11_{\pm 6} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow \mathbf{80}_{\pm 26}$
robomimic lift-mh	$0_{\pm 0} \rightarrow \mathbf{100}_{\pm 1}$	$0_{\pm 0} \rightarrow 60_{\pm 25}$	$90_{\pm 3} \rightarrow \mathbf{98}_{\pm 1}$	$83_{\pm 5} \rightarrow 94_{\pm 3}$	$87_{\pm 3} \rightarrow \mathbf{99}_{\pm 1}$	$97_{\pm 2} \rightarrow \mathbf{100}_{\pm 0}$	$96_{\pm 1} \rightarrow \mathbf{100}_{\pm 0}$
robomimic can-mh	$0_{\pm 0} \rightarrow 70_{\pm 5}$	$0_{\pm 0} \rightarrow \mathbf{97}_{\pm 2}$	$64_{\pm 6} \rightarrow 80_{\pm 4}$	$39_{\pm 7} \rightarrow 71_{\pm 6}$	$53_{\pm 9} \rightarrow 88_{\pm 5}$	$89_{\pm 3} \rightarrow 93_{\pm 2}$	$88_{\pm 2} \rightarrow \mathbf{99}_{\pm 1}$
robomimic square-mh	$0_{\pm 0} \rightarrow 72_{\pm 3}$	$0_{\pm 0} \rightarrow 14_{\pm 5}$	$14_{\pm 2} \rightarrow 28_{\pm 3}$	$4_{\pm 2} \rightarrow 13_{\pm 3}$	$8_{\pm 3} \rightarrow 29_{\pm 4}$	$35_{\pm 5} \rightarrow 81_{\pm 3}$	$42_{\pm 5} \rightarrow \mathbf{94}_{\pm 2}$
robomimic transport-mh	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$1_{\pm 1} \rightarrow 1_{\pm 1}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$0_{\pm 0} \rightarrow 0_{\pm 0}$	$13_{\pm 4} \rightarrow 22_{\pm 6}$	$0_{\pm 0} \rightarrow \mathbf{39}_{\pm 4}$

Table 8: Complete results for Figure 5, showing the performance immediately after 1M steps of offline learning and 1M steps of online learning. Results represent the mean ± 95% confidence interval calculated across 10 seeds.

Environment	Random	Uniform	Greedy	ACH
OGBench antmaze-giant-navigate-task1	$0_{\pm 0} \to 25_{\pm 18}$	$0_{\pm 0} \to 26_{\pm 8}$	$0_{\pm 0} \to \mathbf{77}_{\pm 7}$	$0_{\pm 0} \to 69_{\pm 8}$
OGBench antmaze-giant-navigate-task2	$0_{\pm 0} \to 68_{\pm 18}$	$0_{\pm 0} \to \mathbf{88}_{\pm 2}$	$0_{\pm 0} \to \mathbf{86}_{\pm 4}$	$0_{\pm 0} \to \mathbf{90}_{\pm 2}$
OGBench antmaze-giant-navigate-task3	$0_{\pm 0} \to 13_{\pm 8}$	$0_{\pm 0} \to 2_{\pm 1}$	$0_{\pm 0} \to \mathbf{40}_{\pm 7}$	$0_{\pm 0} \to 10_{\pm 3}$
OGBench antmaze-giant-navigate-task4	$0_{\pm 0} \to 45_{\pm 25}$	$0_{\pm 0} \to 91_{\pm 3}$	$0_{\pm 0} \to \mathbf{92}_{\pm 3}$	$0_{\pm 0} \to \mathbf{97}_{\pm 1}$
OGBench antmaze-giant-navigate-task5	$0_{\pm 0} \to 27_{\pm 26}$	$4_{\pm 3} \to 91_{\pm 4}$	$6_{\pm 5} \to \mathbf{94}_{\pm 3}$	$3_{\pm 5} \to \mathbf{97}_{\pm 1}$
OGBench antsoccer-arena-navigate-task1	$23_{\pm 6} \to 70_{\pm 6}$	$1_{\pm 1} \to 83_{\pm 4}$	$1_{\pm 1} \to \mathbf{89}_{\pm 3}$	$1_{\pm 1} \to \mathbf{91}_{\pm 2}$
OGBench antsoccer-arena-navigate-task2	$7_{\pm 2} \to 73_{\pm 4}$	$0_{\pm 0} \to 80_{\pm 3}$	$0_{\pm 0} \to \mathbf{90}_{\pm 3}$	$0_{\pm 1} \to \mathbf{91}_{\pm 2}$
OGBench antsoccer-arena-navigate-task3	$1_{\pm 1} \to 23_{\pm 7}$	$0_{\pm 0} \to 73_{\pm 4}$	$0_{\pm 1} \to 78_{\pm 10}$	$0_{\pm 0} \to \mathbf{84}_{\pm 5}$
OGBench antsoccer-arena-navigate-task4	$6_{\pm 2} \to 55_{\pm 7}$	$0_{\pm 0} \to 63_{\pm 4}$	$0_{\pm 0} \to \mathbf{75}_{\pm 3}$	$0_{\pm 0} \to \mathbf{76}_{\pm 3}$
OGBench antsoccer-arena-navigate-task5	$4_{\pm 2} \to 55_{\pm 6}$	$0_{\pm 0} \to 67_{\pm 4}$	$0_{\pm 0} \to \mathbf{72}_{\pm 6}$	$0_{\pm 0} \to \mathbf{75}_{\pm 3}$
OGBench cube-quadruple-play-10M-task1	$2_{\pm 1} \to 95_{\pm 3}$	$0_{\pm 0} \to 13_{\pm 20}$	$0_{\pm 0} \to 36_{\pm 29}$	$0_{\pm 0} \to \mathbf{97}_{\pm 2}$
OGBench cube-quadruple-play-10M-task2	$0_{\pm 0} \to 55_{\pm 14}$	$0_{\pm 0} \to 1_{\pm 2}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to \mathbf{50}_{\pm 21}$
OGBench cube-quadruple-play-10M-task3	$0_{\pm 0} \to 7_{\pm 4}$	$0_{\pm 0} \to 31_{\pm 20}$	$0_{\pm 0} \to 27_{\pm 16}$	$0_{\pm 0} \to \mathbf{65}_{\pm 13}$
OGBench cube-quadruple-play-10M-task4	$0_{\pm 0} \to 6_{\pm 5}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 14_{\pm 17}$	$0_{\pm 0} \to \mathbf{45}_{\pm 20}$
OGBench cube-quadruple-play-10M-task5	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 0_{\pm 0}$
OGBench cube-triple-play-task1	$14_{\pm 6} \to \mathbf{100}_{\pm 0}$	$4_{\pm 4} \to 7_{\pm 8}$	$2_{\pm 2} \to 92_{\pm 12}$	$3_{\pm 3} \to \mathbf{100}_{\pm 0}$
OGBench cube-triple-play-task2	$0_{\pm 0} \to \mathbf{75}_{\pm 17}$	$0_{\pm 0} \to 15_{\pm 19}$	$0_{\pm 0} \to 44_{\pm 29}$	$0_{\pm 0} \to 60_{\pm 14}$
OGBench cube-triple-play-task3	$1_{\pm 1} \to 52_{\pm 11}$	$1_{\pm 1} \to 80_{\pm 19}$	$0_{\pm 0} \to 81_{\pm 13}$	$0_{\pm 0} \to \mathbf{94}_{\pm 4}$
OGBench cube-triple-play-task4	$0_{\pm 0} \to 27_{\pm 10}$	$0_{\pm 0} \to 20_{\pm 25}$	$0_{\pm 0} \to 50_{\pm 18}$	$0_{\pm 0} \to \mathbf{81}_{\pm 7}$
OGBench cube-triple-play-task5	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 0_{\pm 0}$
OGBench humanoidmaze-medium-navigate-task1	$0_{\pm 0} \to 30_{\pm 12}$	$49_{\pm 5} \to 31_{\pm 2}$	$38_{\pm 5} \to \mathbf{88}_{\pm 6}$	$37_{\pm 9} \to \mathbf{89}_{\pm 3}$
OGBench humanoidmaze-medium-navigate-task2	$8_{\pm 2} \to 54_{\pm 5}$	$52_{\pm 4} \to 78_{\pm 7}$	$63_{\pm 3} \to \mathbf{94}_{\pm 3}$	$57_{\pm 4} \to \mathbf{99}_{\pm 1}$
OGBench humanoidmaze-medium-navigate-task3	$23_{\pm 10} \to 9_{\pm 16}$	$42_{\pm 7} \to 50_{\pm 5}$	$51_{\pm 5} \to \mathbf{89}_{\pm 4}$	$58_{\pm 4} \to \mathbf{95}_{\pm 2}$
OGBench humanoidmaze-medium-navigate-task4	$0_{\pm 0} \to 1_{\pm 1}$	$6_{\pm 3} \to 5_{\pm 2}$	$3_{\pm 4} \to 22_{\pm 4}$	$3_{\pm 2} \to \mathbf{30}_{\pm 4}$
OGBench humanoidmaze-medium-navigate-task5	$29_{\pm 6} \to 85_{\pm 6}$	$67_{\pm 8} \to 82_{\pm 7}$	$70_{\pm 4} \to \mathbf{96}_{\pm 1}$	$71_{\pm 5} \to \mathbf{99}_{\pm 1}$
OGBench puzzle-4x4-play-task1	$35_{\pm 7} \to \mathbf{99}_{\pm 1}$	$63_{\pm 15} \to \mathbf{100}_{\pm 0}$	$49_{\pm 6} \to \mathbf{100}_{\pm 0}$	$52_{\pm 11} \to \mathbf{100}_{\pm 0}$
OGBench puzzle-4x4-play-task2	$0_{\pm 0} \to 20_{\pm 26}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 20_{\pm 26}$	$0_{\pm 0} \to \mathbf{70}_{\pm 30}$
OGBench puzzle-4x4-play-task3	$14_{\pm 7} \to 80_{\pm 26}$	$28_{\pm 11} \to 50_{\pm 33}$	$9_{\pm 4} \to \mathbf{100}_{\pm 0}$	$13_{\pm 7} \to \mathbf{100}_{\pm 0}$
OGBench puzzle-4x4-play-task4	$0_{\pm 0} \to 10_{\pm 20}$	$1_{\pm 1} \to 30_{\pm 30}$	$0_{\pm 1} \to 90_{\pm 20}$	$0_{\pm 0} \to \mathbf{100}_{\pm 0}$
OGBench puzzle-4x4-play-task5	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 0_{\pm 0}$	$0_{\pm 0} \to 20_{\pm 26}$	$0_{\pm 0} \to \mathbf{80}_{\pm 26}$