OFFLINE RL WITH HIERARCHICAL ACTION CHUNKING

Ahad Jawaid
The University of Texas at Dallas
ahad.jawaid@utdallas.edu

ABSTRACT

Offline goal-conditioned reinforcement learning (RL) holds the promise of learning general-purpose policies from static datasets. However, effectively scaling these methods to long-horizon tasks remains a significant challenge due to the "curse of horizon," where value estimation errors can compound through long chains of bootstrapped Bellman backups. Existing hierarchical approaches mitigate this by decomposing tasks into subgoals, yet they often rely on low-level controllers that suffer from myopic execution and biased value estimates. In this work, we propose **Hierarchical Implicit Q-Chunking (HiQC)**, a offline RL algorithm that combines high-level latent planning with low-level action chunking. By conditioning the low-level critic on temporally extended action sequences, HiQC enables unbiased k-step value backups, effectively compressing the horizon at both the planning and execution levels. We theoretically demonstrate that this dual decomposition results in a tighter bound on value error under a bounded per-backup error model compared to standard hierarchy or flat chunking alone. Empirically, HiQC outperforms strong baselines on the OGBench suite, particularly in challenging long-horizon navigation tasks such as `humanoid-giant`, while maintaining robust performance on high-dimensional manipulation tasks.

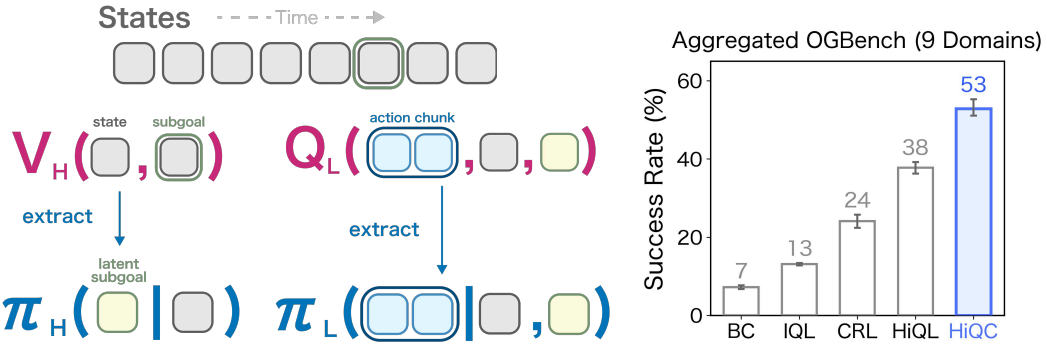

Figure 1: **Hierarchical Implicit Q-Chunking (HiQC).** HiQC addresses the curse of horizon in offline RL through a two-level temporal decomposition. The **high-level planner** (left) operates in a latent space, predicting subgoals z over a coarse horizon c to bridge distant states. The **low-level controller** (right) executes action chunks $\boldsymbol{a}$ of length k to reach these subgoals. By conditioning the low-level critic on full action chunks, HiQC enables unbiased k-step value backups, significantly reducing the recursion depth required for value estimation while ensuring consistent low-level execution.

1 INTRODUCTION

Offline reinforcement learning (RL) provides a framework for deriving optimal policies from static datasets, circumventing the need for expensive online interaction (Levine et al., 2020). An obstacle to the success of offline RL in complex environments is the *curse of horizon* (Liu et al., 2018; Park et al., 2025b). Popular offline RL algorithms rely on temporal difference (TD) learning (Kostrikov et al., 2022a), which minimizes the error between a value estimate and a bootstrapped Bellman target (Sutton, 1988; Sutton & Barto, 2018). In the offline setting, extrapolation errors on out-of-distribution

actions introduce bias into this target (Kumar et al., 2020). As the task horizon increases, these biases can compound through long chains of recursive Bellman backups, rendering value estimates unreliable for distant goals (Park et al., 2025b).

The bias accumulation can be mitigated by employing n-step returns, which skips n steps of rewards before bootstrapping. However, as n grows, this approach suffers from high variance, particularly in stochastic environments or when the data distribution deviates from the policy (Brandfonbrener et al., 2021). Hierarchical reinforcement learning (HRL) offers an alternative mechanism for horizon reduction by decomposing tasks into sequences of high-level subgoals (Nachum et al., 2018; Park et al., 2023). Yet, standard hierarchies often rely on low-level policies that execute single actions to reach these subgoals. Consequently, the low-level policy remains susceptible to the curse of horizon over the duration of the subgoal, while the high-level policy depends on a potentially unstable low-level controller (Park et al., 2025b).

Independently, *action chunking* has emerged as a technique in imitation learning to assist in generating temporally coherent actions where policies predict sequences of future actions rather than single steps (Zhao et al., 2023). Recent work has shown that within TD learning, action chunking allows the critic to perform unbiased n-step backups over the length of the chunk, effectively compressing the execution horizon (Li et al., 2025b). While action chunking is useful for low-level policies, it only reduces the horizon by a constant factor, which may be insufficient for long-horizon tasks.

In this work, we introduce **Hierarchical Implicit Q-Chunking (HiQC)**, an offline RL algorithm designed to mitigate error accumulation through a dual horizon reduction on both the planning horizon and the execution horizon. In HiQC, the low-level policy operates directly on action chunks as the action space. This allows the low-level critic to utilize unbiased k-step returns. Simultaneously, the high-level policy decomposes the long-horizon task into a sequence of subgoals with a significantly shorter horizon. By enforcing this chunk-based low-level policy, HiQC compresses the effective horizon at both the planning and execution levels, significantly reducing the recursion depth required for value estimation.

Our main contributions are as follows:

- We introduce **Hierarchical Implicit Q-Chunking (HiQC)**, an offline RL algorithm that utilizes low-level action chunks as the basis for a latent hierarchical planner. This design enables stable long-horizon learning by ensuring unbiased value propagation at the low level and compressed planning at the high level.
- We provide a theoretical analysis demonstrating that HiQC achieves a tighter bound on value estimation error compared to standard hierarchical or flat chunking methods under a bounded per-backup error model. By decomposing the task horizon T into subgoals and action chunks of size k, we show the resulting bound scales with $\mathcal{O}(\sqrt{T/k})$ rather than $\mathcal{O}(T)$ or $\mathcal{O}(\sqrt{T})$.
- We empirically validate HiQC on the OGBench suite (Park et al., 2025a), showing that it outperforms strong baselines on long-horizon navigation tasks, such as `humanoid-giant` and `pointmaze-giant`, while maintaining competitive performance on high-dimensional manipulation tasks.

2 Related Work

Offline Reinforcement Learning. Offline RL algorithms aim to learn optimal policies from static datasets without online interaction (Levine et al., 2020). A challenge in this setting is the distributional shift between the learned policy and the behavioral policy, which leads to value overestimation for out-of-distribution actions (Kumar et al., 2020). To address this, prior methods have employed constraints on the policy (Fujimoto et al., 2019), conservative value estimation (Kumar et al., 2020), or in-sample advantage weighted regression such as Implicit Q-Learning (IQL) (Kostrikov et al., 2022b). While these methods have shown success, they often struggle in long-horizon tasks due to the accumulation of temporal difference (TD) errors over many time steps, a phenomenon characterized as the curse of horizon (Park et al., 2025b). Our work builds upon the value learning stability of IQL but addresses the horizon bottleneck through hierarchical and low-level action abstractions.

Hierarchical Reinforcement Learning. Hierarchical RL (HRL) decomposes long-horizon problems into tractable sub-problems, typically formalized under the options framework (Sutton et al., 1999).

In the context of goal-conditioned RL, this often manifests as a high-level policy generating subgoals for a low-level controller (Nachum et al., 2018). While early methods operated in the raw observation space, recent approaches have demonstrated the efficacy of planning over latent representations. Notably, Hierarchical Implicit Q-Learning (HIQL) (Park et al., 2023) learns a two-level hierarchy from a single value function, where the high-level policy predicts latent subgoals and the low-level policy reaches them. Although HIQL effectively reduces the decision horizon for the high-level policy, its low-level controller operates at a single-step granularity, leaving it susceptible to local horizon effects and high-frequency actuation noise. HiQC extends this paradigm by integrating action chunking into the low-level controller, further compressing the effective horizon.

Action Chunking in Learning. Action chunking, the prediction of action sequences rather than single atomic actions, has gained prominence in imitation learning to model temporal correlations and mitigate compounding errors (Zhao et al., 2023; Intelligence et al., 2025; Park et al., 2024). In the realm of reinforcement learning, Q-Chunking (Li et al., 2025b) adapted this concept to TD learning, demonstrating that using action chunks on the critic allows for unbiased n-step backups. This effectively accelerates value propagation and encourages temporally coherent exploration. Variations such as Decoupled Q-Chunking (Li et al., 2025a) have further refined this by separating the critic's chunk length from the policy's execution horizon. HiQC unifies these streams of research: we employ the latent goal-reaching structure of HIQL to handle global task decomposition into subgoals, while leveraging the temporal abstraction of Q-Chunking at the local level to ensure robust low-level execution and efficient value learning.

3 Preliminaries

Problem Setting. We consider the offline goal-conditioned reinforcement learning setting modeled as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p(s'|s,a)$ is the transition dynamics, and $\gamma \in [0,1)$ is the discount factor. We assume the goal space $\mathcal{G}$ coincides with the state space $\mathcal{S}$. The reward function $r(s,g)$ typically indicates binary success (e.g., $\mathbb{I}[s=g]$) or sparse distance. We are provided with a static dataset $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^{N}$ consisting of trajectories $\tau = (s_0, a_0, s_1, \ldots, s_T)$. The objective is to learn a goal-conditioned policy $\pi(a|s,g)$ that maximizes the expected cumulative discounted reward from $\mathcal{D}$ without further environment interaction.

Implicit Q-Learning (IQL). IQL (Kostrikov et al., 2022b) is an offline RL algorithm that avoids querying values for out-of-distribution actions—a common source of value overestimation—by treating the Bellman maximization as an expectile regression problem. IQL learns a value function $V_\psi(s,g)$ and a Q-function $Q_\theta(s,a,g)$ by minimizing the following objectives:

$$L_V(\psi) = \mathbb{E}_{s,a,g\sim\mathcal{D}}\left[L_2^\tau\left(Q_{\hat{\theta}}(s,a,g) - V_\psi(s,g)\right)\right], \tag{1}$$

$$L_Q(\theta) = \mathbb{E}_{s,a,s',g\sim\mathcal{D}}\left[\left(r(s,g) + \gamma V_\psi(s',g) - Q_\theta(s,a,g)\right)^2\right], \tag{2}$$

where $L_2^\tau(u) = |\tau - \mathbb{I}(u<0)|u^2$ is the expectile loss with parameter $\tau \in (0.5, 1)$. The value function V_ψ approximates the expectile of the Q-values, which implicitly estimates the value of the best actions supported by the data. The policy is then extracted via advantage-weighted regression (AWR) (Peng et al., 2019).

Q-Learning with Action Chunking. Action chunking extends the standard MDP formulation by treating sequences of actions as atomic units. Let $\boldsymbol{a}_{t:t+k} = (a_t, a_{t+1}, \ldots, a_{t+k-1}) \in \mathcal{A}^k$ denote an action chunk of length k. In this setting, the policy $\pi(\boldsymbol{a}_{t:t+k}|s_t)$ predicts a sequence of k actions, which are executed open-loop.

Standard approaches using n-step returns in offline RL often introduce bias because the intermediate actions in the dataset may not match the learned policy (Li et al., 2025b). Q-Chunking addresses this by training a critic $Q(s_t, \boldsymbol{a}_{t:t+k})$ conditioned on the entire action sequence. Because the critic evaluates the specific sequence present in the buffer, the k-step Bellman backup becomes unbiased with respect to that sequence (see Figure 2):

$$Q(s_t, \boldsymbol{a}_{t:t+k}) \leftarrow r(s_t) + \gamma^k V(s_{t+k}). \tag{3}$$

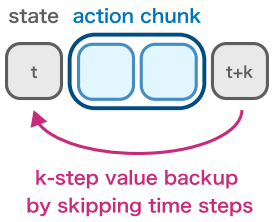

Figure 2: **k-step Value Backup.** Action chunks enable unbiased value propagation by skipping intermediate steps.

This formulation allows for efficient value propagation over the horizon k while avoiding the off-policy bias inherent in standard multi-step returns (Li et al., 2025b).

4 Hierarchical Implicit Q-Chunking

We propose **Hierarchical Implicit Q-Chunking (HiQC)**, a hierarchical offline RL algorithm designed to mitigate the curse of horizon by combining latent high-level planning with temporally extended low-level execution. HiQC decomposes the learning problem into two levels: a high-level (H) policy that plans a sequence of subgoals in a learned latent space over a horizon c, and a low-level (L) policy that executes action chunks of length k to reach these subgoals.

4.1 High-Level Learning: Latent Planning

Following the architecture of HIQL (Park et al., 2023), we train a high-level value function $V_H(s, g)$ parametrized as $V_H(s, g) \approx w(s, \phi(g))$, where $\phi : \mathcal{S} \to \mathcal{Z}$ maps states to a latent subgoal space $\mathcal{Z}$. This parameterization simultaneously learns the value function and the representation ϕ required for planning.

The high-level value function estimates the discounted return of reaching the global goal g from state s with a temporal abstraction of c steps (the subgoal horizon). We define the goal-conditioned reward function as $r(s, g) = \mathbb{I}(s = g) - 1$, where the agent receives 0 upon reaching the goal and -1 otherwise. We train V_H using Implicit V-Learning (IVL) (Kostrikov et al., 2022b), minimizing the expectile loss over c-step returns:

$$\mathcal{L}_{V_H}(\psi) = \mathbb{E}_{\tau\sim\mathcal{D},t}\left[L_2^\tau\left(r(s_t, g) + \gamma^c V_H(s_{t+c}, g) - V_H(s_t, g)\right)\right]. \tag{4}$$

The high-level policy $\pi_H(z|s, g)$ is trained to predict the latent subgoal $z = \phi(s_{t+c})$ that leads to high-value states. We employ advantage-weighted regression (AWR) (Peng et al., 2019) to maximize the likelihood of subgoals z that have high advantage:

$$\mathcal{L}_{\pi_H}(\vartheta) = \mathbb{E}_{\tau\sim\mathcal{D},t}\left[\exp\left(\frac{V_H(s_{t+c}, g) - V_H(s_t, g)}{\alpha}\right)\log \pi_H(\phi(s_{t+c})|s_t, g)\right]. \tag{5}$$

4.2 Low-Level Learning: Q-Chunking

The low-level controller operates on action chunks $\boldsymbol{a} \in \mathcal{A}^k$, where $\boldsymbol{a} = (a_t, \ldots, a_{t+k-1})$ represents a sequence of k atomic actions. The goal of the low-level policy is to execute a chunk that reaches the latent subgoal z specified by the high level. Note that the chunk size k is an execution hyperparameter distinct from the planning horizon c. We define the low level subgoal-conditioned reward function as $r(s, z) = \mathbb{I}(s = z) - 1$, where the agent receives 0 upon reaching the goal and -1 otherwise.

Chunked Critic and Value. To enable unbiased multi-step backups, we apply Q-Chunking (Li et al., 2025b) to the low-level critic. We define a chunk-conditioned Q-function $Q_L(s, \boldsymbol{a}, z)$ and a state-value function $V_L(s, z)$. We train V_L and Q_L using IQL adapted for action chunks. The value function V_L approximates the expectile of the chunk Q-values, while Q_L regresses to a k-step target derived from the dataset transitions:

$$\mathcal{L}_{V_L}(\eta) = \mathbb{E}_{s,\boldsymbol{a},z\sim\mathcal{D}}\left[L_2^\tau\left(Q_L(s, \boldsymbol{a}, z) - V_L(s, z)\right)\right], \tag{6}$$

$$\mathcal{L}_{Q_L}(\theta) = \mathbb{E}_{s,\boldsymbol{a},s_{t+k},z\sim\mathcal{D}}\left[\left(r(s, z) + \gamma^k V_L(s_{t+k}, z) - Q_L(s, \boldsymbol{a}, z)\right)^2\right]. \tag{7}$$

Conditioning Q_L on the full action chunk $\boldsymbol{a}$ removes the off-policy bias typically associated with n-step returns, allowing the low level to propagate values efficiently over the chunk horizon k without divergence.

Flow-Based Action Chunking Policy. To model the complex, high-dimensional distribution of action sequences, we parameterize the low-level policy $\pi_L(\boldsymbol{a}|s, z)$ using Conditional Flow Matching (CFM) (Lipman et al., 2023; Zhang et al., 2025). The policy is defined by a vector field $v_\omega(t, x, s, z)$ that transforms noise $x_0 \sim \mathcal{N}(0, I)$ into an action chunk $\boldsymbol{a}$. We extract the optimal policy via advantage-weighted flow matching:

$$\mathcal{L}_{\pi_L}(\omega) = \mathbb{E}_{t,x_t,s,\boldsymbol{a},z\sim\mathcal{D}}\left[\exp\left(\frac{Q_L(s, \boldsymbol{a}, z) - V_L(s, z)}{\beta}\right)\left\|v_\omega(t, x_t, s, z) - u_t(x|\boldsymbol{a})\right\|^2\right], \tag{8}$$

where $u_t(x|\boldsymbol{a})$ is the target vector field and β is a temperature parameter.

Algorithm 1 Hierarchical Implicit Q-Chunking (HiQC)

Input: Dataset $\mathcal{D}$, subgoal horizon c, chunk size k.
Initialize: $V_H, \pi_H, V_L, Q_L, \pi_L$.
while training **do**
  Sample batch of trajectories $\tau \sim \mathcal{D}$.
  // High-Level Update (Planning Horizon c)
  Sample time steps t. Set current state s_t, goal g.
  Compute latent target $z = \phi(s_{t+c})$.
  Update V_H via Implicit V-Learning using Eq. 4.
  Update π_H via AWR using Eq. 5.
  // Low-Level Update (Execution Horizon k)
  Extract action chunk $\boldsymbol{a} = a_{t:t+k}$.
  Update V_L via Chunked IQL using Eq. 6.
  Update Q_L via k-step regression using Eq. 7.
  Update π_L via Advantage-Weighted Flow Matching using Eq. 8.
Inference:
Given current state s and global goal g:
  1. Sample latent subgoal $z \sim \pi_H(\cdot|s, g)$.
  2. Generate action chunk $\boldsymbol{a} \sim \pi_L(\cdot|s, z)$.
  3. Execute $\boldsymbol{a}$ open-loop for k steps.
  4. Repeat (High-level planning may recur at frequency $1/c$ or $1/k$).

5 Theoretical Analysis

We analyze HiQC using a *bootstrap-chain* error model: value learning propagates information backward by repeatedly applying bootstrapped targets, and each bootstrap step can introduce bounded error. The key quantity is therefore the *number of bootstraps* required to propagate information over the horizon.

5.1 Bootstrap-Chain Error Model

Let T be the task horizon in *atomic* environment steps. Consider any value-learning scheme that propagates values backward through a sequence of bootstrapped updates. We define:

Bootstrap depth. Let D denote the number of bootstraps (Bellman-style backups) that must be chained to propagate terminal signal back to the start state under a given temporal abstraction.

Assumption 1 (Bounded one-backup error) For a given value recursion, each bootstrap step introduces at most ϵ value error in magnitude. For hierarchical methods we use ϵ_H for the high-level recursion and ϵ_L for the low-level recursion.

Under Assumption 1, the value error grows at most linearly with the number of chained bootstraps.

Lemma 1 (Error grows with bootstrap depth) If a value estimate is obtained by chaining D bootstraps and each bootstrap contributes at most ϵ error (Assumption 1), then the resulting value error is bounded by

$$\mathcal{E} \;\leq\; D\,\epsilon. \tag{9}$$

Lemma 1 is proved by a one-line unrolling argument in Appendix B. We now compute D for TD learning, flat chunking, standard hierarchy, and HiQC.

5.2 Baselines

For readability we assume T is divisible by k and c, and c is divisible by k.[1]

[1] If not divisible, the same bounds hold up to rounding by replacing ratios with $\lceil \cdot \rceil$. This does not change the scaling and is omitted for clarity.

Standard TD learning (IQL). TD bootstraps every atomic step, so the chain length is $D = T$. By Lemma 1,

$$\mathcal{E}_{\text{TD}}(T) \;\le\; T\,\epsilon. \tag{10}$$

Flat action chunking (QC). With chunk size k, the critic bootstraps every k steps, reducing the number of bootstraps to $D = T/k$. By Lemma 1,

$$\mathcal{E}_{\text{QC}}(T,k) \;\le\; \frac{T}{k}\,\epsilon. \tag{11}$$

Standard hierarchy (HIQL-style). With subgoal spacing c, the high level bootstraps every c steps, so its chain length is $D_H = T/c$ and contributes $(T/c)\epsilon_H$. The low level must bridge c atomic steps using one-step bootstraps, so $D_L = c$ and contributes $c\,\epsilon_L$. Summing the two gives

$$\mathcal{E}_{\text{HIQL}}(T,c) \;\le\; \frac{T}{c}\,\epsilon_H \;+\; c\,\epsilon_L. \tag{12}$$

5.3 HiQC

HiQC uses the same high-level recursion as HIQL (bootstrapping every c steps), but its low-level critic bootstraps over *chunks* of length k. To bridge a subgoal interval of length c, the low level therefore chains $D_L = c/k$ chunk-bootstraps (rather than c one-step bootstraps). Applying Lemma 1 at each level yields:

Theorem 1 (HiQC bootstrap-chain bound) Under Assumption 1, HiQC satisfies

$$\mathcal{E}_{\text{HiQC}}(T,c,k) \;\le\; \frac{T}{c}\,\epsilon_H \;+\; \frac{c}{k}\,\epsilon_L. \tag{13}$$

Moreover, the right-hand side is minimized at

$$c^\star \;=\; \sqrt{\frac{Tk\,\epsilon_H}{\epsilon_L}}, \tag{14}$$

with minimum value

$$\min_c \mathcal{E}_{\text{HiQC}} \;\le\; 2\sqrt{\epsilon_H\epsilon_L}\,\sqrt{\frac{T}{k}} \;=\; \mathcal{O}\left(\sqrt{\frac{T}{k}}\right). \tag{15}$$

The proof is given in Appendix B. Intuitively, chunking reduces the number of low-level bootstraps needed per subgoal interval from c to c/k, which improves the optimized scaling from $\mathcal{O}(\sqrt{T})$ to $\mathcal{O}(\sqrt{T/k})$ in this bootstrap-depth model.

6 Experiments

In this section, we empirically evaluate Hierarchical Implicit Q-Chunking (HiQC) to determine if the combination of high-level latent planning and low-level action chunking effectively mitigates the curse of horizon. We focus our analysis on three key questions: (1) Does HiQC outperform standard hierarchical and flat baselines on long-horizon navigation tasks? (2) How does it compare to state-of-the-art offline GCRL methods on high-dimensional manipulation tasks? (3) What is the impact of key design choices, such as the flow-based policy parameterization and chunk size?

6.1 Experimental Setup

Environments. We evaluate our method on the OGBench suite (Park et al., 2025a), a benchmark designed to stress-test offline GCRL algorithms. We select 9 tasks covering two distinct domains: *locomotion* (`antmaze`, `humanoidmaze`, `pointmaze`) and *manipulation* (`cube`, `scene`, `puzzle`). We strictly do not use the oracle state representation in any of our evaluations. The reward is sparse as defined in our preliminaries. Furthermore, for the low-level policy, the reward function retains the

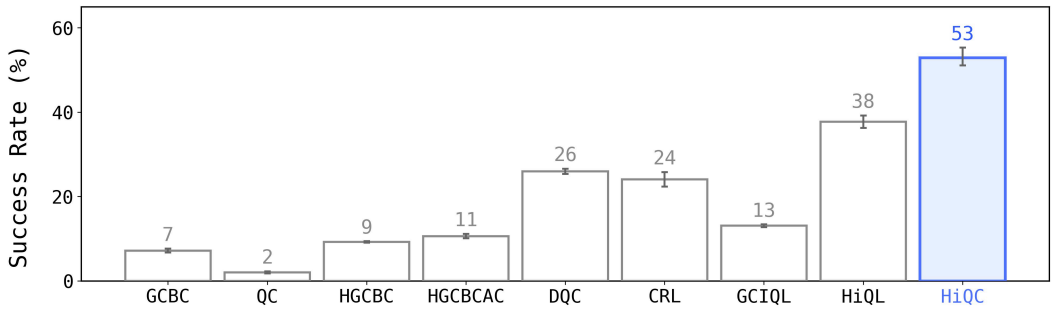

Figure 3: **Aggregated Performance on OGBench.** We report the mean success rate across 9 diverse domains.

Table 1: **Success rates on OGBench tasks.** We report the mean success rate 95% confidence interval over 4 seeds. The best performing method for each task is highlighted in **bold**.

Task	BC	QC	HBC	HBCAC	DQC	CRL	GCIQL	HiQL	HiQC
antmaze-large	27 [24,30]	0 [0,0]	49 [48,51]	54 [52,56]	35 [33,38]	81 [78,84]	32 [25,37]	88 [86,89]	**93 [92,94]**
antmaze-giant	0 [0,0]	0 [0,0]	25 [22,27]	30 [27,33]	1 [0,1]	14 [11,18]	0 [0,1]	**68 [67,69]**	**68 [67,69]**
humanoid-large	1 [1,2]	0 [0,0]	4 [3,5]	6 [5,7]	1 [1,1]	17 [12,23]	1 [1,2]	34 [23,44]	**42 [39,44]**
humanoid-giant	0 [0,0]	0 [0,0]	1 [0,1]	0 [0,0]	0 [0,0]	4 [2,5]	0 [0,0]	10 [7,15]	**33 [29,39]**
pointmaze-large	29 [25,34]	15 [13,17]	0 [0,0]	0 [0,1]	62 [60,65]	36 [32,41]	29 [24,33]	47 [39,55]	**87 [81,92]**
pointmaze-giant	1 [0,1]	0 [0,0]	0 [0,0]	0 [0,0]	35 [29,40]	33 [27,39]	1 [0,3]	44 [35,53]	**80 [76,85]**
cube-triple	1 [1,2]	0 [0,0]	0 [0,0]	0 [0,0]	**8 [7,10]**	4 [3,4]	1 [1,2]	3 [2,4]	**9 [4,15]**
puzzle-4x6	0 [0,0]	0 [0,0]	0 [0,0]	0 [0,0]	**16 [14,17]**	5 [4,6]	5 [4,6]	4 [4,5]	1 [1,2]
scene-play	5 [4,5]	3 [2,3]	4 [4,5]	5 [5,6]	**76 [75,77]**	22 [21,23]	47 [44,49]	41 [38,44]	65 [61,68]
Overall	7 [6,8]	2 [2,2]	9 [8,10]	11 [10,11]	26 [23,28]	23 [21,26]	13 [11,15]	42 [37,47]	**53 [50,56]**

same sparse structure but evaluates reaching the generated subgoal rather than the overarching task goal. To rigorously test horizon capabilities, we include the "giant" variants of navigation tasks (e.g., `humanoidmaze-giant`), which require reasoning over thousands of time steps and have been shown to break prior methods (Park et al., 2025b).

Baselines. We compare HiQC against a diverse set of baselines representing three paradigms:

- **Flat Baselines:** Goal-Conditioned Behavioral Cloning (GCBC), Contrastive RL (CRL) (Eysenbach et al., 2022), and Goal-Conditioned Implicit Q-Learning (GCIQL) (Kostrikov et al., 2022b).
- **Action Chunking Baselines:** Q-Chunking (QC) (Li et al., 2025b) and Decoupled Q-Chunking (DQC) (Li et al., 2025a). These methods utilize temporal abstraction but lack explicit high-level planning.
- **Hierarchical Baselines:** Hierarchical GCBC (HGCBC) (Park et al., 2025b), HGCBC with Action Chunking (HGCBCAC), and Hierarchical Implicit Q-Learning (HIQL) (Park et al., 2023). We implement HGCBCAC by extending HGCBC to predict action chunks at the low level.

Implementation Details. We trained all methods for 1 million gradient steps with a batch size of 1024. Experiments were conducted on NVIDIA RTX 4090 GPUs, with each run taking approximately 1.5 hours. We report results averaged over 4 random seeds. Due to a limited compute budget, we did not perform extensive hyperparameter tuning for HiQC. Instead, we largely adopted the default hyperparameters from HIQL (Park et al., 2023) for the high-level hierarchy and QC (Li et al., 2025b) for the low-level chunking components. All policies and value functions were parameterized as Multi-Layer Perceptrons (MLPs). Full architectural and hyperparameter details are provided in Supplementary Material A.

6.2 Comparative Analysis on OGBench

We present the main comparative results in Table 1 and summarize the aggregate performance in Figure 3.

Long-Horizon Navigation. HiQC demonstrates its most significant advantage in long-horizon locomotion tasks. In `pointmaze-giant` and `humanoid-giant`, which require traversing extremely long paths, flat methods (GCBC, QC, GCIQL) fail almost completely, illustrating the severity of the curse of horizon for flat value functions (Park et al., 2025b). While standard HIQL achieves non-trivial performance on `humanoid-giant` (10%), HiQC significantly outperforms it (33%). This result supports our theoretical analysis (Theorem 1): by utilizing unbiased k-step backups in the low-level critic, HiQC compresses the effective horizon, reducing the number of recursive Bellman backups required for value propagation, which typically destabilizes hierarchical execution in giant mazes. On `antmaze-large`, HiQC (93%) surpasses both HIQL (88%) and CRL (81%).

High-Dimensional Manipulation. On manipulation tasks such as `cube-triple` and `scene-play`, which require precise sequential object interactions, HiQC remains competitive. Notably, HiQC outperforms standard QC and HIQL on `cube-triple` (9% vs 0% and 3%). However, we observe that Decoupled Q-Chunking (DQC) performs exceptionally well on `scene-play` (76%) and `puzzle-4x6` (16%). We hypothesize that DQC achieves this superior performance because its policy predicts smaller chunks than the high-level critic's horizon. This allows it to handle open-loop execution better in highly reactive manipulation environments. Nevertheless, HiQC outperforms standard QC and purely hierarchical methods across the board, demonstrating robust generalization.

6.3 Ablation Studies

Impact of Flow Matching Policies. To verify the importance of our policy parameterization, we compare HiQC using a standard Gaussian policy against our default Conditional Flow Matching (CFM) policy. As shown in Figure 4, the flow-based policy consistently achieves higher success on 9 environments from OGBench. This aligns with prior findings in imitation learning (Zhao et al., 2023) and Q-learning (Li et al., 2025b), suggesting that flow matching better captures the multimodal distributions inherent in trajectory data, preventing the "averaging" artifacts of Gaussian policies that can disrupt temporally extended actions.

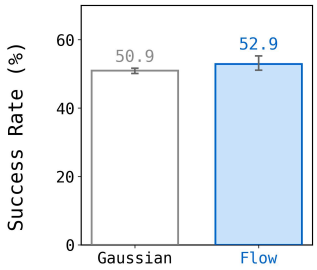

Figure 4: **Flow vs. Gaussian Policy Ablation.**

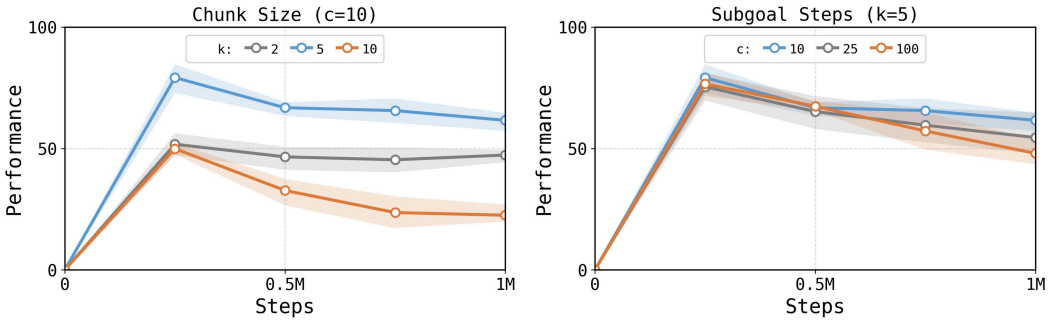

Figure 5: **Sensitivity Analysis.** Performance as a function of action chunk size k (left) and high-level subgoal interval c (right) evaluated on the *scene-play* environment with 4 seeds.

Sensitivity to Hierarchy and Chunk Size. We analyze the sensitivity of HiQC to the action chunk size k and the high-level planning horizon c in Figure 5.

- **Chunk Size (k):** Performance improves as k increases from 1 to 5, confirming the benefit of horizon compression. However, excessively large chunks ($k > 10$) degrade performance,

likely because open-loop execution becomes susceptible to compounding errors in the dynamics, a limitation noted in prior chunking work (Li et al., 2025b).

- **Subgoal Horizon (c):** We observe a "sweet spot" for the high-level planning horizon. If c is too small, the high-level problem retains the complexity of the original task; if c is too large, the low-level policy struggles to reach the distant subgoal. HiQC generally benefits from a larger c than standard HIQL, as the chunked low-level policy can reliably cover more ground per decision step.

7 Discussion

We presented Hierarchical Implicit Q-Chunking (HiQC), a hierarchical offline RL algorithm that addresses the curse of horizon (Park et al., 2025b) through a dual temporal decomposition: a high-level latent planner that decomposes long-horizon tasks into subgoals spaced by c steps, and a low-level controller that executes action chunks of size k to reach each subgoal. By conditioning the low-level critic on full action chunks, HiQC enables unbiased k-step value backups (Li et al., 2025b), compressing the effective horizon at both the planning and execution levels. Our theoretical analysis (Theorem 1) shows that, under a bounded per-backup error model, this decomposition yields an error bound scaling of $\mathcal{O}(\sqrt{T/k})$, improving over standard two-level hierarchies (Park et al., 2023) by a factor of $1/\sqrt{k}$ within this model. Empirically, HiQC outperforms baselines on long-horizon navigation tasks in OGBench (Park et al., 2025a), with particularly large gains on `humanoid-giant` and `pointmaze-giant`, while remaining competitive on high-dimensional manipulation tasks.

Several limitations remain. HiQC uses a fixed chunk size k across all states, and as shown in our ablation studies (Figure 5), excessively large chunks degrade performance due to compounding open-loop errors (Li et al., 2025b); developing adaptive, state-conditioned chunk boundaries is an open problem shared with prior chunking methods (Li et al., 2025a). Furthermore, our theoretical analysis relies on the assumption of bounded per-backup errors, which may not hold strictly when combined with deep neural network function approximation. Additionally, the high-level value function is unbiased only under deterministic dynamics (Park et al., 2023), and as noted by Park et al. (2025b), two-level hierarchies only mitigate rather than fundamentally solve error accumulation in TD learning. Extending HiQC to stochastic settings and deeper hierarchies remains an important direction for future work.

References

David Brandfonbrener, William F Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 2021.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as a reinforcement learning algorithm. *arXiv preprint arXiv:2206.07568*, 2022.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2019.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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, 2025. URL https://arxiv.org/abs/2504.16054.

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

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

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.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled q-chunking. *arXiv preprint arXiv:2512.10926*, 2025a.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, 2025b.

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

Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, 2018.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

J Hyeon Park, Wonhyuk Choi, Sunpyo Hong, Hoseong Seo, Joonmo Ahn, Changsu Ha, Heungwoo Han, and Junghyun Kwon. Hierarchical action chunking transformer: Learning temporal multi-modality from demonstrations with fast imitation behavior. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024.

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

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

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. In *Advances in Neural Information Processing Systems*, 2025b.

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

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International Conference on Learning Representations*, 2025.

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

A Implementation Details and Hyperparameters

We provide the full implementation details and hyperparameters for HiQC and all baselines. All algorithms were implemented in JAX. We build on top of the codebases released by Park et al. (2023) and Park et al. (2025a).

A.1 Shared Hyperparameters

Table 2 describes the common hyperparameters shared across all methods. We follow the network architecture and optimization settings used in HIQL Park et al. (2023) and OGBench Park et al. (2025a).

Table 2: **Shared hyperparameters across all methods.**

Parameter	Value
Optimizer	Adam
Learning rate	3×10^{-4}
Batch size	1024
Discount factor (γ)	0.99 (HIQL, GCIQL, CRL, HiQC), 0.999 (QC, DQC)
Target network update rate (τ)	0.005
Number of training steps	10^6
Network hidden dimensions	$(512, 512, 512)$
Nonlinearity	GELU
Layer normalization	True (first two layers)
Value goal sampling $(w^{\mathrm{v}}_{\mathrm{cur}}, w^{\mathrm{v}}_{\mathrm{traj}}, w^{\mathrm{v}}_{\mathrm{rand}})$	$(0.2, 0.5, 0.3)$
Value geometric sampling	True
Actor goal sampling $(w^{\mathrm{p}}_{\mathrm{cur}}, w^{\mathrm{p}}_{\mathrm{traj}}, w^{\mathrm{p}}_{\mathrm{rand}})$	$(0.0, 1.0, 0.0)$

For methods utilizing flow matching policies (HiQC, DQC, QC), the policy is parameterized by a conditional vector field $v_\omega(t, x, s, z)$ and integrated using the Euler method with 10 flow steps during inference. For methods using Gaussian policies (GCBC, HGCBC, HGCBCAC, HIQL, GCIQL, CRL), we use a constant standard deviation parameterization for the actors, following Park et al. (2023).

A.2 HiQC Hyperparameters

HiQC combines a high-level latent planner from HIQL Park et al. (2023) with a low-level chunked critic from Q-Chunking Li et al. (2025b). Due to a limited compute budget, we did not perform extensive hyperparameter tuning. Instead, we adopted the default hyperparameters from HIQL for the high-level hierarchy and from QC for the low-level chunking components. Table 3 reports the HiQC-specific shared hyperparameters, and Table 4 reports the task-specific hyperparameters.

Table 3: **HiQC-specific hyperparameters (shared across all tasks).**

Parameter	Value
Latent goal dimension ($\lvert\mathcal{Z}\rvert$)	10
High-level AWR temperature (α_H)	3.0
Low-level AWR temperature (α_L)	3.0
Low-level policy type	Conditional Flow Matching
Flow steps	10

The high-level value function is trained via Implicit V-Learning Kostrikov et al. (2022b) using c-step returns (Eq. 4), while the low-level critic uses an explicit Q-function with k-step Bellman backups (Eq. 7). We found that the explicit Q-function was important for the low-level critic to enable standard TD error minimization over action chunks. For `scene-*`, we used a reduced learning rate of 10^{-6}, target update rate of 10^{-5}, and low-level AWR temperature $\alpha_L = 1.0$ for training stability.

Table 4: **HiQC task-specific hyperparameters.** We adjust the subgoal horizon c, chunk size k, and expectile τ based on the domain. Locomotion domains use a lower expectile (0.5) while manipulation domains use higher expectiles (0.9–0.93).

Domain	Subgoal Steps (c)	Chunk Size (k)	Expectile (τ)
`pointmaze-*`	25	5	0.5
`antmaze-*`	25	2	0.5
`humanoidmaze-*`	100	5	0.5
`cube-*`	10	5	0.93
`scene-*`	10	5	0.9
`puzzle-*`	10	5	0.9

A.3 Baseline Hyperparameters

Baseline hyperparameters were selected to match the configurations reported in their respective original papers or the OGBench benchmark Park et al. (2025a). For HIQL and GCIQL, we follow the hyperparameters from Park et al. (2023) and Park et al. (2025a), respectively. For DQC and QC, we follow Li et al. (2025a) and Li et al. (2025b). Table 5 reports the task-specific hyperparameters for all baselines.

Table 5: **Task-specific hyperparameters for baselines.** For each method, we report only the parameters that deviate from the shared defaults in Table 2. All other parameters use the shared defaults.

	HIQL		GCIQL		CRL
Domain	c	α	τ	α	α
`pointmaze-*`	25	3.0	0.9	0.003	0.03
`antmaze-*`	25	3.0	0.9	0.3	0.1
`humanoidmaze-*`	100	3.0	0.9	0.1	0.1

Table 6: **Task-specific hyperparameters for action chunking baselines.** DQC and QC use $\gamma = 0.999$ and flow matching policies. HGCBC and HGCBCAC use Gaussian policies.

	DQC				QC		HGCBCAC	
Domain	h	h_a	κ_b	κ_d	h=h_a	κ_b	c	k
`pointmaze-*`	25	5	0.9	0.5	5	0.9	25	5
`antmaze-*`	25	5	0.9	0.5	5	0.9	25	2
`humanoidmaze-*`	25	1	0.5	0.8	5	0.5	100	5
`cube-triple`	25	5	0.93	0.8	5	0.93	10	5
`scene-*`	25	5	0.9	0.5	5	0.9	10	5
`puzzle-4x6`	25	5	0.7	0.5	5	0.7	10	5

HGCBC / HGCBCAC. These hierarchical behavioral cloning baselines use the same high-level architecture as HiQC (latent subgoal prediction with AWR) but lack the value-based planning component. HGCBC predicts single actions at the low level, while HGCBCAC extends HGCBC to predict action chunks. Both use Gaussian policy parameterizations with the same subgoal horizons as HiQC (Table 4). For HGCBCAC, the chunk size is $k = 2$ for `antmaze-*` and $k = 5$ for all other domains.

GCBC. Goal-Conditioned Behavioral Cloning uses a Gaussian policy with the same network architecture as the other methods. No value function is trained.

B Theoretical Proofs and Derivations

This appendix proves Lemma 1 and Theorem 1 using only unrolling and the AM–GM inequality.

B.1 Proof of Lemma 1

Proof. Consider a sequence of D bootstraps indexed by $j = 0, 1, \ldots, D-1$ (these indices can represent atomic steps, chunk steps, or high-level steps depending on the method). Let V_j be the exact value along this recursion and $\widehat{V}_j$ the learned value. Assume each bootstrap step satisfies

$$\widehat{V}_j \;=\; (\text{exact bootstrap target using } \widehat{V}_{j+1}) \;+\; \delta_j, \qquad |\delta_j| \le \epsilon. \tag{16}$$

Define the error $e_j \triangleq \widehat{V}_j - V_j$. Subtracting the exact recursion from the approximate recursion gives

$$e_j \;=\; e_{j+1} + \delta_j, \tag{17}$$

where $e_D = 0$ at the terminal end of the chain (no further bootstrap). Unrolling from $j = 0$ yields

$$e_0 \;=\; \sum_{j=0}^{D-1} \delta_j \quad\Longrightarrow\quad |e_0| \;\le\; \sum_{j=0}^{D-1} |\delta_j| \;\le\; D\,\epsilon. \tag{18}$$

This proves equation 9. □

B.2 Proof of Theorem 1

Proof. We compute bootstrap depths at each level.

High level. The high-level recursion advances c atomic steps per bootstrap across total horizon T, so the bootstrap depth is $D_H = T/c$. By Lemma 1, the high-level contribution is at most $(T/c)\epsilon_H$.

Low level. Within a single high-level interval of length c, the low-level chunk recursion advances k atomic steps per bootstrap, so the bootstrap depth is $D_L = c/k$. By Lemma 1, the low-level contribution is at most $(c/k)\epsilon_L$.

Summing gives equation 13. To optimize over c, consider

$$f(c) \triangleq \frac{T}{c}\epsilon_H + \frac{c}{k}\epsilon_L, \qquad c > 0. \tag{19}$$

By AM–GM inequality, for any $c > 0$,

$$\frac{T}{c}\epsilon_H + \frac{c}{k}\epsilon_L \;\ge\; 2\sqrt{\left(\frac{T}{c}\epsilon_H\right)\left(\frac{c}{k}\epsilon_L\right)} \;=\; 2\sqrt{\epsilon_H\epsilon_L}\sqrt{\frac{T}{k}}. \tag{20}$$

Equality holds when the two terms are equal:

$$\frac{T}{c}\epsilon_H = \frac{c}{k}\epsilon_L \quad\Longrightarrow\quad c^\star = \sqrt{\frac{Tk\,\epsilon_H}{\epsilon_L}}. \tag{21}$$

This proves equation 14 and equation 15. □

C Per-Goal Evaluation Results

To provide a more granular understanding of HiQC's performance and address variations in goal difficulty, we report the per-goal success rates across the full suite of evaluated OGBench environments. Each environment defines up to 5 individual tasks (goals) whose difficulty varies substantially. In Table 7, we present the mean success rates and corresponding 95% confidence intervals evaluated over 4 random seeds.

Table 7: **HiQC Per-Goal Success Rates.** We report the mean and 95% confidence intervals across 4 seeds for each task in the evaluated OGBench environments. Note that for manipulation environments (e.g., `scene-play`), the columns correspond sequentially to specific sub-tasks defined by the environment (e.g., `task1_open`, `task2_unlock_and_lock`).

Environment	Task 1	Task 2	Task 3	Task 4	Task 5	Overall
antmaze-large	92 [91,93]	88 [86,89]	96 [96,97]	94 [93,96]	94 [93,95]	93 [92,93]
antmaze-giant	36 [30,43]	84 [83,85]	64 [60,66]	65 [62,68]	91 [90,92]	68 [67,69]
humanoid-large	45 [42,48]	14 [11,16]	65 [61,69]	48 [45,52]	37 [36,38]	42 [40,43]
humanoid-giant	28 [23,33]	32 [23,40]	26 [24,28]	39 [35,44]	39 [36,41]	33 [29,37]
pointmaze-large	100 [99,100]	85 [77,93]	83 [75,90]	68 [50,82]	99 [98,100]	87 [83,91]
pointmaze-giant	89 [83,95]	79 [72,85]	64 [58,70]	95 [93,98]	72 [59,85]	80 [76,83]
cube-triple	30 [21,40]	4 [1,8]	8 [2,16]	0 [0,0]	1 [0,1]	9 [5,13]
puzzle-4x6	5 [4,6]	1 [0,1]	0 [0,0]	0 [0,0]	0 [0,0]	1 [1,2]
scene-play	74 [61,84]	58 [54,62]	67 [61,71]	77 [70,83]	49 [40,57]	65 [62,67]