ACSAC: ADAPTIVE CHUNK SIZE ACTOR-CRITIC WITH CAUSAL TRANSFORMER Q-NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon, sparse-reward tasks pose a fundamental challenge for reinforcement learning, since single-step TD learning suffers from bootstrapping error accumulation across successive Bellman updates. Actor-critic methods with action chunking address this by operating over temporally extended actions, which reduce the effective horizon, enable fast value backups, and support temporally consistent exploration. However, existing methods rely on a fixed chunk size and therefore cannot adaptively balance reactivity against temporal consistency. A large fixed chunk size reduces responsiveness to new observations, while a small one produces incoherent motions, forcing task-specific tuning of the chunk size. To address this limitation, we propose **A**daptive **C**hunk **S**ize **A**ctor-**C**ritic (**ACSAC**). ACSAC leverages a causal Transformer critic to evaluate expected returns for action chunks of different sizes. At each chunk boundary, it adaptively selects the chunk size that maximizes the expected return, supporting flexible, state-dependent chunk sizes. We prove that the ACSAC Bellman operator is a γ-contraction whose unique fixed point is the action-value function of the adaptive policy. Experiments on OGBench demonstrate that ACSAC achieves strong performance on long-horizon, sparse-reward manipulation and navigation tasks across both offline RL and offline-to-online RL settings.

1 INTRODUCTION

Offline reinforcement learning (RL) trains policies from previously collected datasets without environment interaction (Levine et al., 2020), and the offline-to-online setting further refines the offline-pretrained policy through additional online interaction (Nair et al., 2020). However, on long-horizon, sparse-reward tasks, single-step temporal difference (TD) learning suffers from bootstrapping error accumulation, since each update regresses toward the Q-network's own next-state estimate and small errors compound across many recursive Bellman updates (Park et al., 2025b). A common remedy is to use multi-step returns, which accelerate value propagation by shifting the regression target further into the future, but they introduce an off-policy bias because the intermediate rewards are collected under the behavior policy rather than the current policy (Li et al., 2025a;b). Action chunking has emerged as a recent and effective response to these limitations.

Action chunking trains the critic and the policy on action sequences rather than single actions, which enables multi-step value backups without incurring off-policy bias, since the critic conditions on the full sequence (Li et al., 2025b). Beyond these critic-side benefits, predicting full action sequences also produces temporally coherent behaviors that capture non-Markovian patterns in the data and improve exploration in sparse-reward settings (Zhao et al., 2023; Li et al., 2025b). However, existing action-chunking actor-critic methods rely on a fixed chunk size, manually tuned per task and shared across all states (Li et al., 2025b; 2026; Kim et al., 2026). Yet the optimal balance between reactivity and temporal consistency is itself state-dependent. Stable phases admit long open-loop chunks, whereas sensitive states demand frequent replanning (Liang et al., 2026). Over-committing to a long chunk in such states can drive the agent off the goal-reaching path (Fig. 1).

To address this limitation, we propose **A**daptive **C**hunk **S**ize **A**ctor-**C**ritic (**ACSAC**), which adaptively selects the chunk size to maximize the critic's value estimate. At each replanning state, ACSAC samples multiple candidate action chunks of length H from an expressive flow BC policy. A causal Transformer critic provides **calibrated and cross-horizon comparable** Q-value estimates

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.

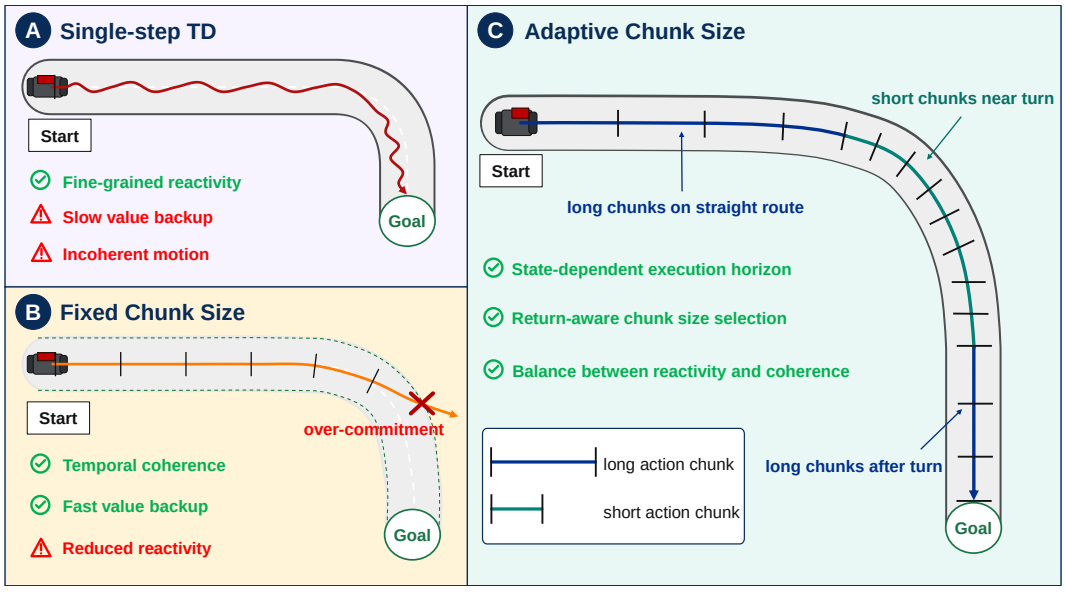


Figure 1: **Motivation for adaptive action chunk size.** **(A)** Single-step execution preserves fine-grained reactivity by replanning at every step, but suffers from slow value backups and produces incoherent motions. **(B)** A fixed chunk size improves motion coherence and accelerates value propagation, but its open-loop execution reduces reactivity within the chunk. In sensitive states such as turns, this over-commitment to a long chunk can drive the agent off the goal-reaching path, lowering the return. **(C)** ACSAC adaptively selects the execution horizon per state via return-aware chunk size selection, executing longer chunks on the straight route and shorter chunks near the turn, achieving a balance between reactivity and coherence.

for every *prefix* (i.e., the first h actions of a chunk, $h = 1, \dots, H$) of each candidate. ACSAC then applies rejection sampling jointly over the candidate index and the prefix length, executes the highest-value prefix, and replans at the next chunk boundary. ACSAC thus retains the multi-step value backups and coherent exploration of action chunking, while the chunk size becomes state-dependent rather than fixed, adaptively balancing reactivity and temporal consistency at each state.

Our contributions can be summarized as follows: 1) We propose **ACSAC**, an action-chunking actor-critic that adaptively selects the chunk size per state, using a causal Transformer critic and joint rejection sampling over candidate index and prefix length. 2) We prove that ACSAC's Bellman operator is a γ-contraction whose unique fixed point is the action-value function of the adaptive policy, and that the prefix-conditioned Q-values at this fixed point are cross-horizon comparable. 3) On long-horizon, sparse-reward manipulation and navigation tasks from OGBench (Park et al., 2025a), ACSAC achieves strong performance against single-step, multi-step, and fixed chunk size baselines in both offline and offline-to-online settings.

2 Related Work

Offline RL and offline-to-online RL. Offline RL aims to learn a policy from a fixed dataset without environment interaction (Levine et al., 2020). The main challenge is the distributional shift between the behavior policy and the learned policy, which can cause value overestimation and suboptimal performance (Fujimoto & Gu, 2021). Recently, expressive generative policies based on diffusion and flow matching (Ho et al., 2020; Lipman et al., 2023) have been widely adopted for their expressivity over Gaussian policies. Common policy extraction strategies for these generative policies include reparameterized gradients (Wang et al., 2023; Park et al., 2025c), weighted regression (Lu et al., 2023; Zhang et al., 2025), and rejection sampling (Chen et al., 2023; Hansen-Estruch et al., 2023). In the offline-to-online setting, the offline-pretrained policy is further fine-tuned with online interactions (Nair et al., 2020), with techniques such as balanced sampling (Lee et al., 2022), high update-to-data ratios (Ball et al., 2023), value calibration (Nakamoto et al., 2023), and more. Our method uses the same algorithm for both offline and online training, simply adding online transitions to the offline dataset and applying none of the above specialized techniques.

Action chunking. Action chunking originated in imitation learning, where a policy predicts and executes a sequence of actions in an open-loop manner, improving robustness and capturing non-Markovian behavior (Zhao et al., 2023; Chi et al., 2025). Recent RL methods have brought action chunking into actor-critic frameworks, where the critic evaluates whole action chunks and enables multi-step backup without off-policy bias. In the online setting, CQN-AS (Seo & Abbeel, 2025) learns a multi-level factorized critic on action chunks, and AC3 (Yang et al., 2026) builds on a DDPG-style framework to predict continuous action chunks. In the offline or offline-to-online setting, Q-chunking (Li et al., 2025b) runs RL at an action chunk level with a flow BC policy and rejection sampling. DQC (Li et al., 2026) decouples the policy chunk size from the critic chunk size, with the policy predicting a shorter action chunk while retaining the value learning benefits of the chunked critic. MAC (Park et al., 2026) combines an action-chunk dynamics model with rejection sampling from an expressive flow BC policy. DEAS (Kim et al., 2026) leverages action sequences for training critics with detached value learning and classification loss. CGQ (Song et al., 2026) regularizes a single-step critic toward a chunked critic. All of the above RL methods rely on a fixed chunk size across states. However, the optimal chunk size may vary by state, and any fixed choice forces a single trade-off between reactivity and temporal consistency. Our method brings adaptive, state-dependent chunk size selection to RL, driven by the critic's value estimate.

Transformer-based Q-networks. Transformers have become strong backbones for the Q-network in RL. Q-Transformer (Chebotar et al., 2023) converts Q-function estimation into a discrete token sequence modeling problem, with per-dimension discretization treating each action dimension as a separate time step for autoregressive Q-value prediction. TQL (Dong et al., 2026) scales Transformer Q-learning through per-layer control of attention entropy that prevents attention collapse. While Q-Transformer and TQL apply Transformers to single-action Q-networks, recent work instead trains Transformer critics on action chunks for direct multi-step value evaluation. TOP-ERL (Li et al., 2025a) introduces a causal Transformer critic in episodic RL, with the policy outputting full trajectories via movement primitives. T-SAC (Tian et al., 2026) adapts a similar causal Transformer critic to step-based RL, conditioning the critic on short trajectory segments while keeping the actor single-step. SEAR (Nagy et al., 2026) combines a causal Transformer critic with multi-horizon targets and random replanning during data collection. CO-RFT (Huang et al., 2025) applies a causal Transformer critic to fine-tune vision-language-action models with offline RL. None of these methods adapt the chunk size to state. Our method also adopts a causal Transformer critic, but uniquely exploits its prefix-conditioned outputs at all prefix lengths to support state-dependent chunk size selection during both training and inference.

3 PRELIMINARIES

Problem formulation. We consider an infinite-horizon Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^d$ is the continuous action space of dimension d, $T(s'|s,a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition dynamics distribution, $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in (0,1)$ is the discount factor. Here $\Delta(\mathcal{X})$ denotes the set of probability distributions over a space $\mathcal{X}$. We assume access to a prior offline dataset $\mathcal{D}$ consisting of transition rollouts $\{(s,a,s',r)\}$ collected from $\mathcal{M}$. The goal is to find a policy $\pi(a|s) : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return $J(\pi) := \mathbb{E}_{s_{t+1}\sim T(s_t,a_t),a_t\sim\pi(\cdot|s_t)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\right]$. In the *offline* setting, the policy is learned entirely from the fixed dataset $\mathcal{D}$ without environment interactions; in the *offline-to-online* setting, the offline-pretrained policy is further fine-tuned with online interactions.

Chunk-based reinforcement learning. Standard TD-based methods learn a state-action value function $Q_\phi(s,a)$ by regressing toward the single-step target $r_t + \gamma Q_{\bar{\phi}}(s_{t+1}, a_{t+1})$, where $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$. Each single-step backup propagates value only one step backward, slowing learning in long-horizon tasks. A common strategy is the multi-step return, which replaces the single-step target with $\sum_{\tau=0}^{n-1} \gamma^\tau r_{t+\tau} + \gamma^n Q_{\bar{\phi}}(s_{t+n}, a_{t+n})$ for some horizon $n \geq 1$ and allows for an n-fold speed-up in value propagation. However, the multi-step return introduces off-policy bias, since the discounted reward sum from the replay buffer may not reflect the expected rewards under the current policy when the intermediate actions $a_{t+1}, \ldots, a_{t+n-1}$ are chosen by a different policy (Li et al., 2025a;b).

Action chunking attains the value-propagation speedup of multi-step returns without their off-policy bias by extending RL to action sequences. An action chunk of length H starting at time t is

$a_{t:t+H} := (a_t, a_{t+1}, \ldots, a_{t+H-1}) \in \mathcal{A}^H$, and the corresponding H-step discounted reward is $r_t^H := \sum_{\tau=0}^{H-1} \gamma^\tau r_{t+\tau}$. The chunked critic $Q_\phi(s_t, a_{t:t+H})$ is trained with:

$$\mathcal{L}^{\text{chunk}}(\phi) = \mathbb{E}_{(s_t, a_{t:t+H}, r_t^H, s_{t+H})\sim\mathcal{D}} \left[\left(Q_\phi(s_t, a_{t:t+H}) - r_t^H - \gamma^H Q_{\bar{\phi}}(s_{t+H}, a_{t+H:t+2H}) \right)^2 \right], \quad (1)$$

where $a_{t+H:t+2H} \sim \pi(\cdot|s_{t+H})$. Crucially, unlike the multi-step return where the single-action critic $Q(s_t, a_t)$ is backed up with rewards generated by potentially off-policy actions, the chunked critic $Q(s_t, a_{t:t+H})$ conditions on the exact action sequence used to obtain the multi-step rewards r_t^H, eliminating the off-policy bias (Li et al., 2025b).

Flow policy. Flow matching (Lipman et al., 2023; Liu et al., 2023) is a generative modeling technique that trains a velocity field to transform a noise distribution into a target data distribution. Given a target distribution $p(x) \in \Delta(\mathbb{R}^d)$, flow matching fits a time-dependent velocity field $v_\theta(u, x) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ whose corresponding flow $\psi_\theta(u, x)$, defined by the ODE $\frac{d}{du}\psi_\theta(u, x) = v_\theta(u, \psi_\theta(u, x))$, transforms $\mathcal{N}(0, I_d)$ at $u = 0$ into $p(x)$ at $u = 1$, by minimizing $\mathcal{L}(\theta) = \mathbb{E}_{x^0, x^1, u}[\|v_\theta(u, x^u) - (x^1 - x^0)\|_2^2]$ with $x^0 \sim \mathcal{N}(0, I_d)$, $x^1 \sim p(x)$, and $u \sim \text{Unif}([0, 1])$, where $x^u := (1 - u)x^0 + ux^1$ is the linear interpolation between x^0 and x^1. To use flow matching for policy learning, we train a state-conditioned velocity field $v_\theta : [0, 1] \times \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}^d$ with the behavior cloning objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a\sim\mathcal{D},\, z\sim\mathcal{N}(0,I_d),\, u\sim\text{Unif}([0,1])} \left[\|v_\theta(u, s, a^u) - (a - z)\|_2^2 \right], \quad (2)$$

where $a^u := (1 - u)z + ua$ is the linear interpolation between the noise $z = a^0$ and the action $a = a^1$. We denote the ODE solution at $u = 1$ starting from noise z as $\pi_\theta(s, z) := \psi_\theta(1, s, z)$, which maps a noise vector $z \sim \mathcal{N}(0, I_d)$ to an action $a = \pi_\theta(s, z)$. Compared to Gaussian policies, flow policies can model complex, multi-modal action distributions, making them particularly suited for offline RL where datasets often contain diverse behavior patterns (Park et al., 2025c).

4 METHOD

ACSAC is an action-chunking actor-critic that adaptively selects the chunk size at each replanning state, rather than fixing one across all states as in prior methods. At each replanning state s_t, it samples N candidate chunks of length H from a flow BC policy π_θ, where H is the maximum chunk size. A causal Transformer critic Q_ϕ then evaluates every prefix (i.e., the first h actions, $h \in [H] := \{1, \ldots, H\}$) of each candidate. ACSAC's adaptive policy $\pi_\star$ executes the highest-value prefix, of length $h^\star$, and replans at $s_{t+h^\star}$ (Fig. 2). The critic is trained at every prefix length with a multi-step TD objective, in which $\pi_\star$ also provides the bootstrap action prefix. We describe the causal Transformer critic architecture in Sec. 4.1, the multi-step TD objective in Sec. 4.2, and the adaptive policy extraction by joint rejection sampling in Sec. 4.3.

4.1 CAUSAL TRANSFORMER CRITIC ARCHITECTURE

ACSAC's joint argmax (Sec. 4.3) requires the critic to evaluate action chunks of different lengths $a_{t:t+h}$ ($h \in [H]$) from a single state s_t and to produce values that are comparable across h. An MLP critic on action chunks consumes fixed-length inputs, so reusing it on shorter sub-prefixes can incur causal leakage (Tian et al., 2026). A causal Transformer instead ingests $(s_t, a_{t:t+H})$ as a token sequence and outputs the H prefix-conditioned values $\{Q_\phi(s_t, a_{t:t+h})\}_{h\in[H]}$ jointly, with a causal attention mask that restricts position i to attend only to positions $j \le i$.

This design has three consequences. First, $Q_\phi(s_t, a_{t:t+h})$ depends only on the prefix $a_{t:t+h}$ and not on future actions $a_{t+h:t+H}$, so each value is a valid estimate for executing exactly h actions and the MLP-style causal leakage is eliminated. Second, the shared backbone produces sequence-aware value estimates that capture the temporal structure within the chunk and enable fine-grained credit assignment across positions (Li et al., 2025a; Tian et al., 2026). Third, all H values share a backbone and are jointly trained against per-horizon targets (Sec. 4.2) to estimate expected total discounted return under the same adaptive policy $\pi_\star$. As formalized in Theorem 1, these returns lie on a common return scale, making the joint argmax in $\pi_\star$ meaningful across different prefix lengths.

Theorem 1 (Cross-horizon Q-value comparability; proof in Appendix E.2)**.** *Let $Q^\pi(s_t, a_{t:t+h})$ denote the expected discounted return from executing $a_{t:t+h}$ and then following policy π. For every*

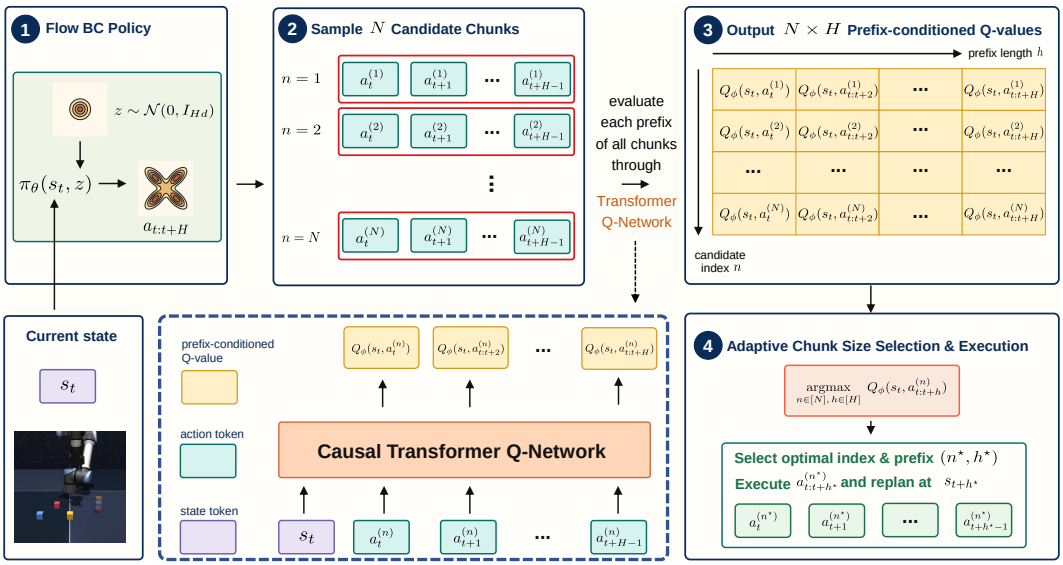


Figure 2: **Adaptive policy extraction in ACSAC.** At replanning state s_t, the flow BC policy maps N noise vectors $z^{(n)} \sim \mathcal{N}(0, I_{Hd})$ to length-H candidate chunks $a^{(n)}_{t:t+H} = \pi_\theta(s_t, z^{(n)})$. The causal Transformer Q-network Q_ϕ takes the state token and each candidate's action tokens as input and jointly outputs an $N \times H$ array of prefix-conditioned Q-values. ACSAC selects $(n^\star, h^\star)$ by jointly maximizing $Q_\phi(s_t, a^{(n)}_{t:t+h})$ over candidate index $n \in [N]$ and prefix length $h \in [H]$. It executes the corresponding prefix $a^{(n^\star)}_{t:t+h^\star}$ and replans at $s_{t+h^\star}$.

$h \in [H]$, with $V^\pi(s)$ denoting the expected discounted return of following π from state s, we have

$$Q^\pi(s_t, a_{t:t+h}) \;=\; \sum_{\tau=0}^{h-1} \gamma^\tau r_{t+\tau} \;+\; \gamma^h \, V^\pi(s_{t+h}). \tag{3}$$

Using the same π for every h makes these Q-values comparable across prefix lengths.

4.2 Multi-Step TD Objective

We train the causal Transformer critic with a multi-step TD loss at every prefix length. For each $h \in [H]$, define the h-step return target

$$G_h(s_t, a_{t:t+h}) := \sum_{\tau=0}^{h-1} \gamma^\tau r_{t+\tau} + \gamma^h \, Q_{\bar{\phi}}\big(s_{t+h},\, \pi_\star(s_{t+h})\big), \tag{4}$$

which sums the h rewards obtained by executing the action prefix $a_{t:t+h}$, and a bootstrap value at s_{t+h}. For the bootstrap value at s_{t+h}, $\pi_\star$ samples N length-H candidate chunks from π_θ and selects the highest-value prefix via joint rejection sampling over candidate index and prefix length (Sec. 4.3). The critic loss averages the squared error against G_h across $h \in [H]$ in expectation over chunked transitions from the offline dataset or replay buffer:

$$\mathcal{L}(\phi) = \mathbb{E}_{(s_{t:t+H+1},\, a_{t:t+H},\, r_{t:t+H}) \sim \mathcal{D}} \left[\frac{1}{H} \sum_{h=1}^{H} \big(Q_\phi(s_t, a_{t:t+h}) - G_h(s_t, a_{t:t+h}) \big)^2 \right]. \tag{5}$$

Each G_h propagates the bootstrap value at s_{t+h} back h steps to $Q_\phi(s_t, a_{t:t+h})$ in a single critic update. Averaging gradients across prefix lengths can further reduce gradient variance while preserving sparse reward signals, as formalized in Appendix E.5 (Tian et al., 2026).

This TD backup generalizes EMaQ's expected-max Q operator (Ghasemipour et al., 2021) from a max over N candidate actions to a joint max over NH candidate prefixes, while retaining its contraction and convergence properties (Theorem 2).

Theorem 2 (Bellman contraction and convergence; proof in Appendix E.3)**.** *Fix the flow BC policy π_θ, rejection sampling size N, and maximum chunk size H. The resulting expected Bellman backup $\mathcal{B}_\theta^{N,H}$ (Definition E.6) satisfies, for any bounded prefix-value functions Q_1, Q_2,*

$$\|\mathcal{B}_\theta^{N,H} Q_1 - \mathcal{B}_\theta^{N,H} Q_2\|_\infty \;\le\; \gamma \, \|Q_1 - Q_2\|_\infty. \tag{6}$$

Repeated application of $\mathcal{B}_\theta^{N,H}$ converges to its unique fixed point $Q_\theta^{N,H}$ from any bounded initial Q-function. Moreover, $Q_\theta^{N,H} = Q^{\pi_\star}$, where the adaptive policy $\pi_\star$ selects the candidate prefix with the highest value under $Q_\theta^{N,H}$.

These fixed-point Q-values lie on a common return scale across prefix lengths (Theorem 1), as required by the joint argmax over (n, h) (Sec. 4.3). We test the learned Q-values for calibration against realized returns and cross-horizon comparability in Sec. 5.3.

4.3 Adaptive Policy Extraction

We use a flow BC policy $\pi_\theta(s, z)$ to generate candidate action chunks of length H from the offline dataset $\mathcal{D}$. π_θ is parameterized by a state-conditioned velocity field $v_\theta : [0,1] \times \mathcal{S} \times \mathbb{R}^{Hd} \to \mathbb{R}^{Hd}$ trained with the flow-matching loss

$$\mathcal{L}(\theta) = \mathbb{E}_{z\sim\mathcal{N}(0,I_{Hd}),\,(s_t,a_{t:t+H})\sim\mathcal{D},\,u\sim\mathrm{Unif}([0,1])}\left[\|v_\theta(u, s_t, a^u_{t:t+H}) - (a_{t:t+H} - z)\|_2^2\right], \tag{7}$$

where $a^u_{t:t+H} := (1-u)z + u a_{t:t+H}$ is the linear interpolation between the noise z and the action chunk $a_{t:t+H}$. Fitting full-length action chunks lets π_θ capture non-Markovian behavior patterns in the offline data, supporting temporally coherent exploration in long-horizon tasks (Li et al., 2025b).

Given π_θ and the trained critic Q_ϕ, we extract ACSAC's policy $\pi_\star$ via rejection sampling, which implicitly enforces a behavior constraint. The KL divergence of the selected full-length chunk distribution from the flow BC policy distribution π_θ has a closed-form bound (Li et al., 2025b). Prefix truncation does not further increase this KL divergence when the same rule is applied to both distributions (Appendix E.4). Rejection sampling is also generally more robust to hyperparameters than alternatives that require tuning a behavior regularization coefficient (Park et al., 2026).

Prior action-chunking methods (Li et al., 2025b; 2026; Park et al., 2026) fix the chunk size h and apply rejection sampling: $n^\star = \arg\max_{n\in[N]} Q_\phi(s_t, a^{(n)}_{t:t+h})$, $\pi(s_t) = a^{(n^\star)}_{t:t+h}$. ACSAC extends this single-axis rejection sampling to a joint search over NH candidates indexed by (n, h) (Fig. 2):

$$(n^\star, h^\star) = \arg\max_{n\in[N],\, h\in[H]} Q_\phi\left(s_t, a^{(n)}_{t:t+h}\right), \qquad \pi_\star(s_t) = a^{(n^\star)}_{t:t+h^\star}, \tag{8}$$

where $\{a^{(n)}_{t:t+H}\}_{n\in[N]}$ are obtained by drawing N noise vectors $z^{(n)} \sim \mathcal{N}(0, I_{Hd})$ and mapping them through the flow BC policy: $a^{(n)}_{t:t+H} = \pi_\theta(s_t, z^{(n)})$. The Transformer critic provides Q-values for all NH pairs, and $(n^\star, h^\star)$ is selected by a single argmax.

Intuitively, the selected execution horizon $h^\star$ reflects the current phase of the policy rollout at state s_t. When the Q-values of longer prefixes decline relative to shorter ones, the agent should replan after fewer steps to maintain reactivity. Conversely, when Q-values stay high or increase with longer prefixes, the state admits a coherent long-horizon plan and the agent benefits from a longer prefix to maintain temporal consistency. The selected execution horizon $h^\star \in [H]$ thus varies per state, with shorter $h^\star$ at sensitive states and longer $h^\star$ during stable phases.

We provide pseudocode in Alg. 1 and further implementation details in Appendix B.

5 Experiments

We evaluate ACSAC on manipulation and navigation tasks from OGBench (Park et al., 2025a). Our experiments are designed to answer three questions. **Q1.** Does ACSAC improve performance in both offline RL and offline-to-online RL over prior methods? **Q2.** Are ACSAC's prefix-conditioned Q-values calibrated and cross-horizon comparable, and does its adaptive chunk size selection follow task phases? **Q3.** How do the maximum chunk size H and the rejection sampling size N affect performance, and does ACSAC's advantage come from the causal Transformer critic alone?

Algorithm 1 Adaptive Chunk Size Actor-Critic (ACSAC)

Require: Dataset $\mathcal{D}$, flow BC policy π_θ, causal Transformer critic Q_ϕ, maximum chunk size H, rejection sampling size N

// Offline training loop
while not converged **do**
  Sample $\{(s_{t:t+H+1}, a_{t:t+H}, r_{t:t+H})\} \sim \mathcal{D}$.
  Update π_θ with the flow-matching loss (Eq. (7)).
  Update Q_ϕ with the multi-step TD loss (Eq. (5)).
// Adaptive policy extraction with rejection sampling
function $\pi_\star(s_t)$
  $z^{(n)} \sim \mathcal{N}(0, I_{Hd})$, $a^{(n)}_{t:t+H} = \pi_\theta(s_t, z^{(n)})$, $n \in [N]$.
  $(n^\star, h^\star) \leftarrow \arg\max_{n\in[N],\ h\in[H]} Q_\phi(s_t, a^{(n)}_{t:t+h})$
  return $a^{(n^\star)}_{t:t+h^\star}$

// Online fine-tuning with adaptive replanning
Initialize $\mathcal{D}$ with offline data.
for every environment step t **do**
  if the previous selected chunk has been fully executed **then**
    $a^\star_{t:t+h^\star} \leftarrow \pi_\star(s_t)$
  Act with $a^\star_t$ and receive s_{t+1}, r_t.
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a^\star_t, s_{t+1}, r_t)\}$.
  Sample $\{(s_{t:t+H+1}, a_{t:t+H}, r_{t:t+H})\} \sim \mathcal{D}$.
  Update π_θ with the flow-matching loss (Eq. (7)).
  Update Q_ϕ with the multi-step TD loss (Eq. (5)).

5.1 Experimental Setup

Environments and datasets. We evaluate ACSAC on eight long-horizon, sparse-reward domains from OGBench (five tasks each, 40 tasks in total), with domain abbreviations following QAM (Li & Levine, 2026). The five manipulation domains are `scene-sparse` (`scene`), `puzzle-3x3-sparse` (`p33`), `cube-double` (`c2`), `cube-triple` (`c3`), and `cube-quadruple` (`c4`). The three navigation domains are `antmaze-giant` (`ag`), `antsoccer-arena` (`soccer`), and `humanoidmaze-medium` (`hm`). We use the standard `play` datasets for manipulation and `navigate` datasets for navigation (Park et al., 2025a). For `cube-quadruple`, we use the 100M-size dataset (Li et al., 2025b). All methods use 4 random seeds per task in the main experiments. Additional domain metadata is provided in Appendix C.1.

Baselines. We compare ACSAC against single-step, multi-step, and fixed chunk size methods. The single-step baselines are `IQL` (Kostrikov et al., 2022), `ReBRAC` (Tarasov et al., 2023), `FQL` (Park et al., 2025c), and `BFN` (Li et al., 2025b). The multi-step baselines are `FQL-n` and `BFN-n` (Li et al., 2025b), which use multi-step returns without learning both a chunked critic and a chunked policy. Finally, we compare against fixed chunk size methods, including `QC`, `QC-FQL` (Li et al., 2025b), and `DEAS` (Kim et al., 2026). Implementation details for all baselines are summarized in Appendix C.2.

Table 1: **Summary table for OGBench offline-to-online RL results.** Each cell reports mean success rates (%) over 4 seeds at the end of 1M offline training steps → 1M additional online steps, with 95% confidence intervals above each mean. Manipulation and navigation are averaged separately. The best nonzero offline and online results in each row are highlighted in color.

	Single-step				Multi-step		Fixed Chunk Size			Adaptive
	`IQL`	`ReBRAC`	`FQL`	`BFN`	`FQL-n`	`BFN-n`	`QC-FQL`	`QC`	`DEAS`	**`ACSAC`**
Manipulation (25 tasks)										
`p33`	0 → 20	62 → **100**	99 → **100**	97 → **100**	99 → **100**	58 → 90	64 → **100**	**100** → **100**	95 → **100**	**100** → **100**
`scene`	0 → 39	15 → 99	52 → 95	87 → 99	21 → 70	55 → 97	86 → 99	82 → 99	92 → 95	**100** → **100**
`c2`	0 → 0	3 → 30	32 → 76	67 → 79	9 → 77	6 → 65	42 → **100**	66 → 98	56 → 86	**85** → **100**
`c3`	0 → 0	0 → 0	2 → 18	6 → 23	1 → 1	0 → 0	3 → 53	5 → 64	4 → 69	**17** → **98**
`c4`	0 → 0	1 → 20	0 → 3	2 → 12	7 → 36	0 → 0	2 → **77**	3 → 73	0 → 9	**9** → 73
`Overall`	0 → 12	16 → 50	37 → 58	52 → 63	28 → 57	24 → 50	39 → 86	51 → 86	50 → 72	**62** → **94**
Navigation (15 tasks)										
`ag`	2 → 2	**47** → 62	6 → 74	0 → 20	22 → 72	21 → 68	0 → 56	0 → 70	1 → 19	0 → **87**
`soccer`	9 → 4	0 → 0	34 → 72	42 → 68	7 → 9	14 → 22	8 → 67	32 → 82	4 → 71	**52** → **93**
`hm`	32 → 16	22 → 1	34 → 59	54 → 84	34 → 54	24 → 38	15 → 26	34 → 73	48 → 62	**66** → **92**
`Overall`	15 → 7	23 → 21	25 → 68	32 → 57	21 → 45	20 → 43	8 → 49	22 → 75	18 → 51	**39** → **91**

5.2 Main Results

We report the offline and offline-to-online RL results in Tab. 1. ACSAC achieves the highest average success rates on both manipulation and navigation in both settings.

Offline RL. ACSAC reaches 62% average success on manipulation and 39% on navigation, compared with 52% and 32% for `BFN`, the strongest baseline in both task families. The gains are particularly clear on `cube-double`, where ACSAC achieves 85% success versus 67% for `BFN`, and `humanoidmaze-medium`, where it reaches 66% versus 54% for `BFN`.

Offline-to-online RL. After 1M additional environment steps, ACSAC reaches 94% average success on manipulation and 91% on navigation, compared with 86% and 75% for `QC`. The largest manipulation gain occurs on `cube-triple`, where ACSAC achieves 98% success versus 64% for `QC` and 69% for `DEAS`. On navigation, ACSAC achieves the highest mean success rate on all three domains, extending its advantage to tasks requiring reactive control. ACSAC remains competitive on `cube-quadruple`, matching `QC` at 73% online success, compared with 77% for `QC-FQL`. Complete per-task results and learning curves are provided in Appendix D.2, with additional ACSAC results over 8 random seeds in Appendix D.3.

5.3 Qualitative and Quantitative Analyses

Prefix-Q calibration and cross-horizon comparability. We test whether ACSAC's prefix-conditioned Q-values are calibrated against realized returns and comparable across horizons. On `cube-triple-task4`, we collect 50 rollouts each for the adaptive policy and five fixed-h variants ($h = 1, \ldots, 5$), using the checkpoint after 1M steps of online fine-tuning. Each fixed-h variant executes the highest-value length-h prefix from N candidate chunks and then follows $\pi_\star$. At each replanning state, we record the predicted Q-value $\hat{Q}$ of the selected prefix and the discounted Monte Carlo return $\hat{G}$ realized from that state. For each policy, we sort these pairs by $\hat{G}$, divide them into ten equal-count bins, and plot the mean $\hat{Q}$ against the mean $\hat{G}$ in each bin. As shown in Fig. 3, the six curves broadly follow the diagonal $\hat{Q} = \hat{G}$ and largely coincide, indicating approximate calibration and a common return scale across horizons.

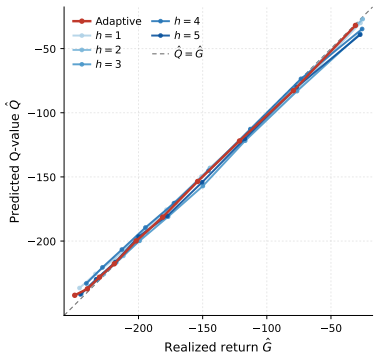


Figure 3: **Prefix-Q calibration.** Points show mean $\hat{Q}$ and mean $\hat{G}$ within each bin. The dashed line denotes $\hat{Q} = \hat{G}$.

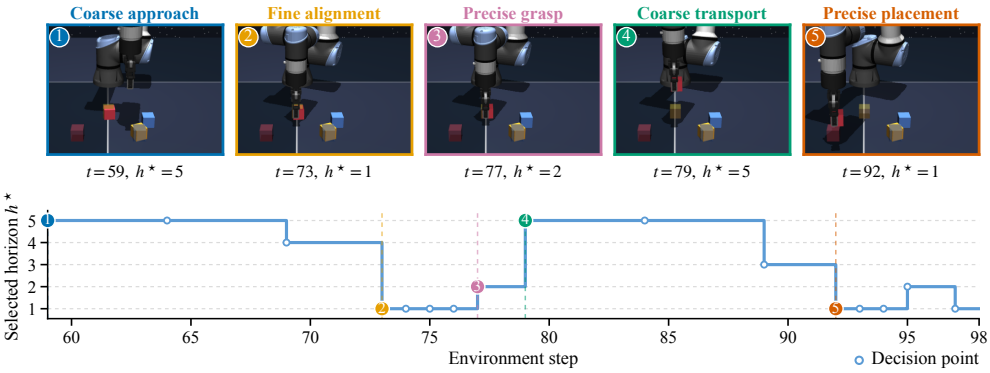


Figure 4: **Qualitative rollout of ACSAC.** Selected execution horizon $h^\star$ during a pick-and-place segment on `cube-triple-task4`. The numbered frames illustrate the corresponding task phases, with labels t and $h^\star$ referring to the marked replanning decisions below.

State-dependent chunk size selection. Fig. 4 visualizes the selected execution horizon $h^\star$ during a pick-and-place segment of an evaluation rollout on `cube-triple-task4`. At the highlighted decisions, ACSAC selects a long chunk for the coarse approach ($h^\star = 5$) and shorter chunks for fine alignment ($h^\star = 1$) and precise grasping ($h^\star = 2$). It then returns to a longer horizon for coarse transport ($h^\star = 5$) and shortens it again as precise placement begins ($h^\star = 1$). This pattern illustrates how maximizing the learned prefix-conditioned Q-values adjusts the replanning frequency across task phases, combining longer open-loop execution during coarse motion with frequent feedback during precise manipulation. Appendix D.5 presents further results with a larger maximum chunk size of $H = 10$.

5.4 Ablation Study

We conduct ablation studies on the `cube-double` and `cube-triple` domains, using 4 random seeds per task.

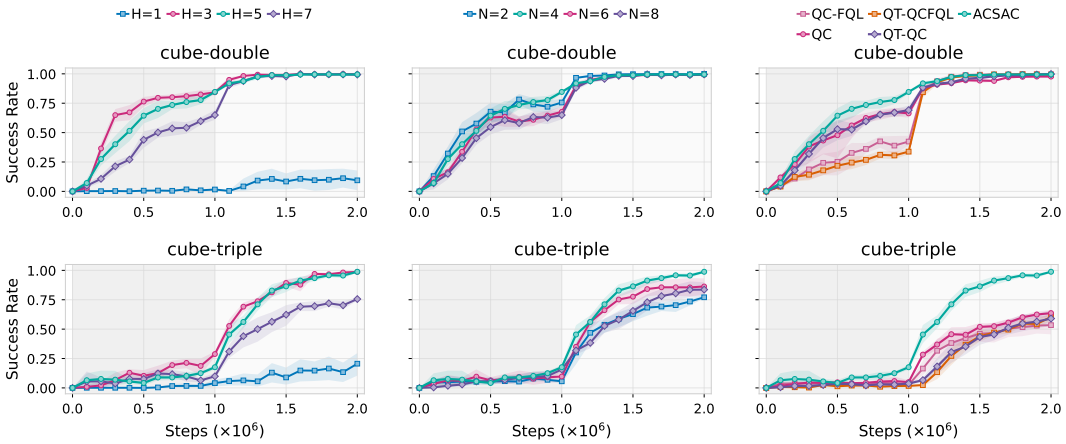


Figure 5: **Ablation studies (4 seeds).** *(a):* maximum chunk size H sweep ($N = 4$). *(b):* rejection sampling size N sweep ($H = 5$). *(c):* QT-QC and QT-QCFQL replace the MLP critics in QC and QC-FQL with ACSAC's causal Transformer critic. Curves aggregate five tasks per domain over 4 random seeds, with 95% confidence intervals. The first 1M steps are offline and the next 1M steps are online.

In Fig. 5 (a,b), we examine the effects of the maximum chunk size $H \in \{1, 3, 5, 7\}$ and rejection sampling size $N \in \{2, 4, 6, 8\}$. H=1 performs poorly on both domains, highlighting the importance of multi-step value propagation for these sparse-reward tasks. H=3 and H=5 achieve similarly strong final online performance, whereas H=7 reduces performance on `cube-triple`. We suspect that longer chunks make the behavior policy and critic harder to learn, as they require modeling longer action sequences and estimating more prefix values with the same network capacity (Li et al., 2025b; Kim et al., 2026). N=4 achieves the best online performance on `cube-triple`, suggesting a favorable balance between candidate coverage and maximization bias. We use H=5 and N=4 in most domains, as this configuration performs well while providing a broad range of execution horizons with a modest sampling budget (Appendix C.2).

In Fig. 5 (c), we replace the MLP critics in QC and QC-FQL with ACSAC's causal Transformer critic, yielding QT-QC and QT-QCFQL while keeping the remaining components unchanged. On `cube-triple`, neither replacement consistently improves upon its corresponding baseline, and both remain substantially below ACSAC. These results show that ACSAC's gains extend beyond the critic backbone. Appendix D.4 further isolates the benefit of adaptive chunk size selection with the trained model held fixed.

6 Conclusion

We propose ACSAC, which learns calibrated and cross-horizon comparable prefix-conditioned Q-values, supporting adaptive replanning without manually fixing the execution horizon. A causal Transformer critic trained with a multi-step TD objective produces these values, and a flow BC policy paired with rejection sampling over the joint (n, h) axes yields the executed chunk. We prove that the ACSAC Bellman operator is a γ-contraction whose unique fixed point is the action-value function of the adaptive policy. Experiments on OGBench demonstrate that ACSAC achieves strong performance on long-horizon, sparse-reward manipulation and navigation tasks across both offline RL and offline-to-online RL settings. A promising future direction is to extend ACSAC's state-dependent chunk size selection to vision-language-action models for real-world robotic deployment.

AI USE STATEMENT

We used large language models to refine and polish our writing. The authors bear full responsibility for the content and presentation of this paper.

REPRODUCIBILITY STATEMENT

We include additional implementation details for hyperparameters, datasets, and evaluation protocols in Appendices B and C. The assumptions and proofs of all theoretical results are provided in Appendix E. Our source code will be made public upon publication.

REFERENCES

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.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, Keerthana Gopalakrishnan, Julian Ibarz, Ofir Nachum, Sumedh Anand Sontakke, Grecia Salazar, Huong T. Tran, Jodilyn Peralta, Clayton Tan, Deeksha Manjunath, Jaspiar Singh, Brianna Zitkovich, Tomas Jackson, Kanishka Rao, Chelsea Finn, and Sergey Levine. Q-Transformer: Scalable offline reinforcement learning via autoregressive Q-functions. In *Conference on Robot Learning*, pp. 3909–3928. PMLR, 2023.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=42zs3qa2kpy.

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. doi: 10.1177/02783649241273668.

Perry Dong, Kuo-Han Hung, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. TQL: Scaling Q-functions with transformers by preventing attention collapse. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=lwHSE1xYHH.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20132–20145, 2021.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. EMaQ: Expected-max q-learning operator for simple yet effective offline and online RL. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

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.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Dongchi Huang, Zhirui Fang, Tianle Zhang, Yihang Li, Lin Zhao, and Chunhe Xia. CO-RFT: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning. *arXiv preprint arXiv:2508.02219*, 2025.

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*, 2026. URL https://openreview.net/forum?id=bVTaAXeBmE.

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

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*, 2025a. URL https://openreview.net/forum?id=N4NhVN30ph.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vd4eNAdtO6.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=XUks1Y96NR.

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

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*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pp. 22825–22855. PMLR, 2023.

C. F. Maximilian 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.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 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*, 2026. URL https://openreview.net/forum?id=WXGb9unEHo.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=M992mjgKzI.

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. URL https://openreview.net/forum?id=hguaupzLCU.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=KVf2SFL1pi.

Younggyo Seo and Pieter Abbeel. Coarse-to-fine q-network with action sequence for data-efficient reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VoFXUNc9Zh.

Gwanwoo Song, Kwanyoung Park, and Youngwoon Lee. Chunk-guided Q-learning. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=yojuz84NLb.

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. doi: 10.1016/S0004-3702(99)00052-1.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 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*, 2026. URL https://openreview.net/forum?id=rb5eTktqbc.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18692–18700, 2026. doi: 10.1609/aaai.v40i22.38937.

Yang Yue, Rui Lu, Bingyi Kang, Shiji Song, and Gao Huang. Understanding, predicting and better resolving Q-value divergence in offline-RL. In *Advances in Neural Information Processing Systems*, volume 36, pp. 60247–60277, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/bd6bb13e78da078d8adcabbe6d9ca737-Abstract-Conference.html.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HA0oLUvuGI.

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 LIMITATIONS

We highlight three limitations of ACSAC and corresponding directions for future work. First, our online fine-tuning follows a simple protocol that adds online transitions to the offline dataset while retaining the same training objective. How adaptive chunk selection interacts with specialized replay strategies, such as balanced sampling (Lee et al., 2022), remains unexplored. Second, while Proposition E.18 establishes contraction and fixed-point results for stochastic transitions, a more comprehensive theoretical analysis of learning in stochastic environments is left for future work. Third, extending ACSAC to large vision-language-action (VLA) models remains an open direction. Since ACSAC selects among candidates from a behavior policy without separately parameterizing an RL policy (Li et al., 2025b), a possible extension is to use a frozen pretrained VLA to generate candidate chunks and train a prefix-conditioned critic for adaptive chunk selection.

B IMPLEMENTATION DETAILS

In this section, we describe the full implementation details of ACSAC.

Online fine-tuning. For offline-to-online RL, we simply add online transitions to the dataset, without distinguishing them from the offline transitions. We continue to train ACSAC with the same objective as in offline training, following Park et al. (2025c) and Li et al. (2025b).

Flow matching. Following Li et al. (2025b), ACSAC trains a flow BC policy on the dataset, parameterized by a state-conditioned velocity field v_θ trained with the flow-matching loss in Eq. (7). At inference, an action chunk is generated by Euler integration of v_θ over F flow steps from $z \sim \mathcal{N}(0, I_{Hd})$. We use a step count of 10 for the Euler method across all tasks.

Causal Transformer. ACSAC ingests $(s_t, a_t, a_{t+1}, \ldots, a_{t+H-1})$ as a token sequence and outputs H prefix values, $[Q_\phi(s_t, a_t), Q_\phi(s_t, a_{t:t+2}), \ldots, Q_\phi(s_t, a_{t:t+H})]$, in one forward pass. A causal attention mask ensures that the h-th prefix value depends only on s_t and $a_{t:t+h}$. Each block consists of multi-head self-attention followed by a two-layer feed-forward network with a hidden dimension of 512. Pre-LayerNorm is applied before every attention and feed-forward sublayer. The Transformer critic uses 2 attention layers, 8 attention heads with per-head dimension 16, and embedding width 128. We adopt this shallow network as a single configuration for all tasks rather than a task-specific setting.

Value learning. Following standard practice, ACSAC trains two Q-networks to improve stability. ACSAC takes the minimum of the two Q-values on manipulation tasks and the mean on navigation tasks, for both the bootstrap target and policy extraction. The discount factor γ is 0.99 on manipulation tasks and 0.995 on navigation tasks. ACSAC's bootstrap target uses the current online critic with the gradient stopped. SEEM (Yue et al., 2023) shows that LayerNorm bounds the critic's NTK, which allows stable training when the online and target networks share parameters.

Training and evaluation. Following the offline-to-online protocol of Park et al. (2025c); Li et al. (2025b), we train ACSAC with 1M offline gradient steps and fine-tune it for 1M online environment steps. We evaluate the agent every 100K steps using 50 episodes, and report the offline and online success rates at 1M and 2M steps, respectively.

Results from prior works. For manipulation tasks, results of `IQL`, `ReBRAC`, `FQL`, `BFN`, `FQL-n`, `BFN-n`, `QC`, and `QC-FQL` are taken from Li et al. (2025b)[1]; note that we use the same setting as this prior work, so the results are fully compatible. For `DEAS` and the baselines on navigation tasks, we individually tune the method-specific hyperparameters for each domain and evaluate them under our protocol.

[1] Released plot data: `https://github.com/ColinQiyangLi/qc/tree/main/plot_data`

C Experimental Details

C.1 Environments, Tasks, and Datasets

We consider five manipulation domains and three navigation domains from OGBench (Park et al., 2025a), using their state-based single-task variants (40 tasks in total). Each OGBench environment provides five evaluation goals, each of which defines a different task (`-singletask-task1` to `-singletask-task5`) (Park et al., 2025c).

Navigation tasks give a reward of -1 when the task is incomplete and 0 when it is completed (Park et al., 2025c). For `scene-sparse` and `puzzle-3x3-sparse`, we sparsify the rewards to $\{-1, 0\}$, following Li et al. (2025b). For `cube-double/triple/quadruple`, we use the default semi-sparse reward: the negative of the number of remaining subtasks (Park et al., 2025c).

We use the standard `navigate` datasets for navigation and `play` datasets for manipulation (Park et al., 2025a). The `cube-triple` and `cube-quadruple` datasets contain 3M and 100M transitions, respectively, following Li et al. (2025b). Tab. 2 lists the dataset size, episode length, and action dimension for each domain.

Table 2: **Domain metadata.**

Domain	Dataset Size	Episode Length	Action Dimension (d)
`scene-sparse-*`	1M	750	5
`puzzle-3x3-sparse-*`	1M	500	5
`cube-double-*`	1M	500	5
`cube-triple-*`	3M	1000	5
`cube-quadruple-100M-*`	100M	1000	5
`antmaze-giant-*`	1M	1000	8
`antsoccer-arena-*`	1M	1000	8
`humanoidmaze-medium-*`	2M	2000	21

C.2 Methods and Hyperparameters

We use the official implementation of FQL (Park et al., 2025c) for IQL and ReBRAC, and that of QC (Li et al., 2025b) for FQL, FQL-n, BFN, BFN-n, QC-FQL, and QC. We reproduce DEAS with its official implementation (Kim et al., 2026), adapted to offline-to-online RL. The hyperparameters for navigation tasks are based on Park et al. (2025c); Song et al. (2026); Kim et al. (2026).

We report the common hyperparameters in Tab. 3 and the domain-specific hyperparameters in Tab. 4. We use the same hyperparameter configuration for all tasks in each domain.

In Tab. 4, α denotes the AWR inverse temperature for IQL and the BC coefficient for the other methods; τ is IQL's expectile coefficient. For ReBRAC on navigation tasks, α is the actor BC coefficient, with the critic BC coefficient fixed at 0.01. H denotes ACSAC's maximum chunk size or the baselines' fixed chunk size; n is the multi-step return horizon, and N is the rejection sampling size.

Table 3: **Common hyperparameters.**

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	3×10^{-4}
Batch size	256
Discount factor γ	0.99 (manipulation), 0.995 (navigation)
Critic ensemble size K	2
UTD ratio	1
Number of flow steps F	10
Number of offline training steps	10^6
Number of online environment steps	10^6
Network width	512
Network depth	4 hidden layers
Nonlinearity	GELU

Table 4: **Domain-specific hyperparameters.**

Domain	IQL (α, τ)	ReBRAC α	FQL α	BFN N	FQL-n (α, n)	BFN-n (n, N)	QC-FQL (α, H)	QC (H, N)	DEAS (α, H)	ACSAC (H, N)
scene-sparse	$(10, 0.9)$	0.1	300	4	$(100, 5)$	$(5, 4)$	$(300, 5)$	$(5, 32)$	$(3, 4)$	$(5, 4)$
puzzle-3x3-sparse	$(10, 0.9)$	0.1	100	4	$(100, 5)$	$(5, 4)$	$(300, 5)$	$(5, 64)$	$(3, 8)$	$(5, 4)$
cube-double	$(0.3, 0.9)$	0.1	300	4	$(100, 5)$	$(5, 4)$	$(300, 5)$	$(5, 32)$	$(300, 4)$	$(5, 4)$
cube-triple	$(0.3, 0.9)$	0.1	300	4	$(100, 5)$	$(5, 4)$	$(100, 5)$	$(5, 32)$	$(1, 4)$	$(5, 4)$
cube-quadruple	$(0.3, 0.9)$	0.1	300	4	$(100, 5)$	$(5, 4)$	$(100, 5)$	$(5, 32)$	$(1, 4)$	$(5, 8)$
antmaze-giant	$(10, 0.9)$	0.01	10	4	$(30, 5)$	$(5, 4)$	$(30, 5)$	$(5, 32)$	$(1, 4)$	$(5, 4)$
antsoccer-arena	$(1, 0.9)$	0.01	30	4	$(30, 5)$	$(5, 4)$	$(30, 5)$	$(5, 32)$	$(0.3, 4)$	$(5, 4)$
humanoidmaze-medium	$(10, 0.9)$	0.01	100	4	$(100, 5)$	$(5, 4)$	$(100, 5)$	$(5, 32)$	$(1, 4)$	$(3, 4)$

D ADDITIONAL RESULTS

D.1 COMPUTATIONAL COSTS

We run all experiments on NVIDIA RTX 3090 GPUs. Tab. 5 compares parameter counts and per-step runtimes on `cube-triple`. ACSAC has the smallest parameter count among the compared methods, using a single shallow Transformer critic to evaluate all prefix lengths. Compared with `QC`, it uses 41% fewer parameters and requires about 10% less time per online step, while achieving a higher success rate (98% versus 64%). Its offline runtime is comparable to `QC`, which shares the same rejection sampling backbone.

One-step flow distillation. ACSAC + Distill additionally distills the flow BC policy into a one-step policy (Park et al., 2025c; 2026), and samples the N candidate chunks from it for both the bootstrap targets in Eq. (4) and the joint argmax in Eq. (8). Each chunk then requires a single network call instead of $F{=}10$ flow steps. This reduces the offline and online per-step runtimes by 29% and 24%, respectively. With online runtime comparable to `FQL`, ACSAC + Distill achieves 80% success, outperforming both `FQL` (18%) and `QC` (64%).

Table 5: **Training speed and parameter count on `cube-triple`.** Offline measures one agent training step. Online measures one agent training step plus one environment step. Parameter counts are measured on `cube-triple` (46-dim observation, 5-dim action). Online performance is the mean success rate (%) across the five `cube-triple` tasks after 1M online steps following 1M offline updates. For all methods, performance is averaged over 4 training seeds, with 95% confidence intervals shown above each mean.

Method	Online Performance (%)	Training Time (ms/step)		Parameter Count
		Offline	Online	
`FQL`	$18^{[16,20]}$	2.84	10.26	4 911 630
`BFN`	$23^{[21,28]}$	3.19	11.82	4 094 473
`QC`	$64^{[61,66]}$	11.50	15.37	4 155 933
`QC-FQL`	$53^{[46,61]}$	4.06	11.55	4 993 590
`DEAS`	$69^{[67,71]}$	3.00	5.42	5 231 137
`ACSAC`	$98^{[97,99]}$	12.50	13.76	2 455 065
`ACSAC + Distill`	$80^{[78,83]}$	8.92	10.49	3 292 722

D.2 FULL RESULTS

Tab. 6 reports per-task offline-to-online RL results aligned with the domain-level summary in Tab. 1. The table covers five manipulation domains and three navigation domains. Fig. 6 provides summary plots by domain, followed by per-task curves in Fig. 7.

Table 6: **Complete OGBench offline-to-online RL results by task.** Each cell reports mean success rates (%) over 4 seeds at the end of 1M offline training steps → 1M additional online steps, with 95% confidence intervals above each mean. Manipulation and navigation are averaged separately. The best nonzero offline and online results in each row are highlighted in color.

		IQL	ReBRAC	FQL	BFN	FQL-n	BFN-n	QC-FQL	QC	DEAS	ACSAC
Manipulation											
puzzle-3x3-sparse	task1	[0,2] [97,100] 1 → 99	[97,100] [100,100] 98 → **100**	[100,100] [100,100] 100 → **100**	[98,100] [100,100] 99 → **100**	[100,100] [100,100] 100 → **100**	[82,89] [100,100] 85 → **100**	[100,100] [100,100] 100 → **100**	[100,100] [100,100] 100 → **100**	[100,100] [100,100] 100 → **100**	[100,100] [100,100] 100 → **100**
	task2	[0,0] [0,0] 0 → 0	[49,80] [100,100] 66 → **100**	[100,100] [100,100] 100 → **100**	[91,100] [100,100] 97 → **100**	[98,100] [100,100] 100 → **100**	[67,80] [100,100] 73 → **100**	[66,100] [100,100] 89 → **100**	[100,100] [100,100] 100 → **100**	[85,100] [100,100] 98 → **100**	[100,100] [100,100] 100 → **100**
	task3	[0,0] [0,0] 0 → 0	[12,44] [100,100] 25 → **100**	[92,100] [100,100] 97 → **100**	[89,98] [100,100] 95 → **100**	[100,100] [100,100] 100 → **100**	[47,59] [79,88] 48 → 82	[0,99] [100,100] 50 → **100**	[100,100] [100,100] 100 → **100**	[82,94] [100,100] 88 → **100**	[100,100] [100,100] 100 → **100**
	task4	[0,0] [0,0] 0 → 0	[42,68] [100,100] 55 → **100**	[94,100] [100,100] 98 → **100**	[95,100] [100,100] 97 → **100**	[97,100] [100,100] 99 → **100**	[33,51] [59,76] 43 → 66	[26,100] [100,100] 75 → **100**	[100,100] [100,100] 100 → **100**	[94,97] [100,100] 95 → **100**	[100,100] [100,100] 100 → **100**
	task5	[0,0] [0,0] 0 → 0	[48,84] [100,100] 68 → **100**	[100,100] [100,100] 100 → **100**	[94,97] [100,100] 95 → **100**	[97,100] [100,100] 98 → **100**	[39,53] [88,100] 46 → 99	[1,9] [100,100] 5 → **100**	[100,100] [100,100] 100 → **100**	[90,98] [100,100] 94 → **100**	[100,100] [100,100] 100 → **100**
	avg.	[0,0] [19,20] 0 → 20	[58,67] [100,100] 62 → **100**	[98,100] [100,100] 99 → **100**	[95,98] [100,100] 97 → **100**	[99,100] [100,100] 99 → **100**	[57,59] [89,92] 58 → 90	[50,70] [100,100] 64 → **100**	[100,100] [100,100] 100 → **100**	[93,97] [100,100] 95 → **100**	[100,100] [100,100] 100 → **100**
scene-sparse	task1	[0,0] [96,99] 0 → 97	[2,18] [100,100] 9 → **100**	[49,80] [100,100] 65 → **100**	[98,100] [100,100] 99 → **100**	[14,38] [100,100] 24 → **100**	[54,79] [100,100] 68 → **100**	[100,100] [100,100] 100 → **100**	[98,100] [100,100] 99 → **100**	[98,100] [100,100] 98 → **100**	[100,100] [100,100] 100 → **100**
	task2	[0,0] [94,100] 0 → 97	[48,80] [100,100] 66 → **100**	[38,61] [98,100] 52 → **100**	[98,100] [98,100] 100 → 99	[2,5] [33,51] 3 → 42	[11,22] [98,100] 16 → 99	[89,96] [100,100] 93 → **100**	[98,100] [100,100] 99 → **100**	[97,100] [100,100] 98 → **100**	[100,100] [100,100] 100 → **100**
	task3	[0,0] [0,0] 0 → 0	[0,0] [98,100] 0 → 99	[58,61] [100,100] 60 → **100**	[88,99] [100,100] 93 → **100**	[2,5] [9,40] 4 → 25	[24,31] [95,99] 28 → 97	[90,99] [100,100] 97 → **100**	[80,96] [100,100] 89 → **100**	[74,89] [97,100] 82 → 96	[100,100] [100,100] 100 → **100**
	task4	[0,0] [0,0] 0 → 0	[0,0] [98,100] 0 → 99	[53,84] [88,100] 69 → 96	[87,93] [96,99] 90 → 98	[49,65] [86,99] 56 → 94	[91,94] [85,100] 93 → 94	[91,98] [95,100] 95 → 97	[87,96] [98,100] 92 → 99	[98,100] [84,96] 100 → **100**	[100,100] [100,100] 100 → **100**
	task5	[0,0] [0,0] 0 → 0	[0,0] [95,98] 0 → 97	[15,25] [73,88] 19 → 81	[30,54] [98,100] 45 → **99**	[14,22] [79,98] 17 → 91	[63,76] [97,100] 68 → **99**	[37,52] [98,100] 44 → **99**	[29,64] [91,100] 43 → 96	[79,89] [95,100] 84 → 93	[96,100] [97,100] **98** → **99**
	avg.	[0,0] [39,39] 0 → 39	[11,19] [99,99] 15 → 99	[43,61] [93,98] 52 → 95	[84,88] [98,100] 87 → 99	[17,26] [64,76] 21 → 70	[53,58] [93,99] 55 → 97	[83,89] [99,100] 86 → 99	[79,85] [98,100] 82 → 99	[91,94] [93,98] 92 → 95	[99,100] [99,100] **100** → **100**
cube-double	task1	[0,2] [0,0] 0 → 0	[7,16] [98,100] 11 → 99	[49,80] [97,100] 64 → 98	[70,77] [94,100] 73 → 98	[17,39] [97,100] 27 → 99	[12,14] [98,100] 13 → 99	[70,87] [100,100] 80 → **100**	[81,84] [98,100] 83 → 99	[89,96] [96,100] 92 → 98	[98,100] [98,100] **98** → 99
	task2	[0,0] [0,0] 0 → 0	[0,2] [4,28] 1 → 16	[37,53] [90,94] 43 → 92	[74,81] [85,92] 77 → 88	[4,7] [94,97] 6 → 96	[1,10] [56,75] 5 → 65	[32,58] [100,100] 45 → **100**	[62,82] [100,100] 72 → **100**	[60,82] [88,96] 71 → 92	[88,93] [100,100] **90** → **100**
	task3	[0,0] [0,0] 0 → 0	[0,2] [14,54] 1 → 34	[23,32] [94,99] 28 → 96	[71,83] [84,91] 77 → 88	[0,2] [88,96] 0 → 91	[3,7] [66,84] 5 → 73	[30,40] [98,100] 35 → 99	[65,87] [98,100] 76 → 99	[41,68] [94,98] 54 → 96	[90,95] [100,100] **92** → **100**
	task4	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[7,9] [0,2] 5 → 1	[18,28] [28,45] 23 → 37	[1,4] [0,2] 2 → 2	[0,2] [0,0] 0 → 0	[7,16] [100,100] 11 → **100**	[25,33] [85,99] 30 → 92	[7,14] [3,19] 8 → 46	[47,54] [98,100] **50** → **100**
	task5	[0,0] [0,0] 0 → 0	[0,2] [0,2] 1 → 1	[15,29] [90,94] 22 → 92	[72,83] [94,98] 78 → 96	[4,16] [93,99] 10 → 96	[3,11] [65,88] 7 → 77	[33,49] [98,100] 41 → **100**	[73,82] [97,100] 77 → 99	[46,63] [94,98] 54 → 96	[89,94] [97,100] **91** → 99
	avg.	[0,0] [0,0] 0 → 0	[2,4] [28,32] 3 → 30	[29,36] [75,76] 32 → 76	[65,69] [78,81] 67 → 79	[5,13] [76,77] 9 → 77	[5,7] [62,69] 6 → 65	[39,47] [100,100] 42 → **100**	[63,70] [96,99] 66 → 98	[52,60] [84,87] 56 → 86	[84,86] [99,100] **85** → **100**
cube-triple	task1	[0,0] [0,2] 0 → 1	[0,2] [0,2] 1 → 0	[2,14] [79,96] 7 → 87	[16,38] [95,98] 28 → 97	[2,10] [0,0] 5 → 3	[0,3] [0,3] 1 → 1	[7,18] [100,100] 13 → **100**	[19,30] [100,100] 23 → **100**	[17,29] [100,100] 22 → **100**	[52,68] [100,100] **60** → **100**
	task2	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 1 → 8	[0,2] [0,26] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,2] [66,93] 0 → 89	[0,0] [52,84] 0 → 89	[0,0] [47,72] 0 → 68	[0,5] [99,100] **2** → **100**
	task3	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,2] [0,12] 0 → 5	[2,2] [5,18] 2 → 12	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [20,70] 0 → 52	[0,3] [70,78] 1 → 73	[0,2] [64,72] 1 → 68	[12,17] [97,99] **14** → **98**
	task4	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,2] 0 → 1	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [10,52] 0 → 27	[0,2] [40,65] 0 → 54	[0,0] [9,24] 0 → 16	[0,9] [98,100] **4** → **99**
	task5	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 94	[1,4] [91,99] **2** → **95**
	avg.	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,3] [16,20] 2 → 18	[4,8] [21,28] 6 → 23	[0,2] [0,1] 1 → 1	[0,1] [0,1] 0 → 0	[2,4] [46,61] 3 → 53	[4,6] [61,66] 5 → 64	[3,6] [67,71] 4 → 69	[15,19] [97,99] **17** → **98**
cube-quadruple	task1	[0,0] [0,0] 0 → 0	[2,6] [97,100] 4 → 98	[0,2] [2,31] 1 → 14	[3,23] [52,64] 13 → 57	[15,26] [95,99] 20 → 97	[0,2] [0,0] 0 → 0	[4,14] [96,100] 9 → 98	[8,13] [98,100] 14 → **100**	[0,4] [7,42] 2 → 45	[16,54] [98,100] **37** → 99
	task2	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 3	[0,0] [0,8] 0 → 3	[3,15] [39,59] **8** → 49	[0,0] [0,0] 0 → 0	[0,0] [96,99] 0 → 97	[0,0] [98,100] 0 → **98**	[0,0] [0,0] 0 → 0	[0,8] [94,100] 3 → **98**
	task3	[0,0] [0,0] 0 → 0	[0,0] [0,2] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,2] 0 → 0	[3,15] [2,6] **9** → 4	[0,0] [0,0] 0 → 6	[0,0] [91,96] 0 → **93**	[0,3] [74,82] 2 → 78	[0,0] [0,2] 0 → 0	[0,8] [90,96] 3 → 73
	task4	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [18,46] 0 → 32	[0,0] [0,0] 0 → 0	[0,0] [91,97] 0 → 95	[0,0] [82,95] 0 → 89	[0,0] [0,0] 0 → 0	[0,0] [96,96] 0 → **96**
	task5	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0	[0,0] [0,0] 0 → 0
	avg.	[0,0] [0,0] 0 → 0	[0,1] [20,20] 1 → 20	[0,0] [0,0] 0 → 3	[0,4] [2,13] 2 → 12	[6,9] [30,37] 7 → 36	[0,0] [0,0] 0 → 0	[1,2] [75,77] 2 → **77**	[2,4] [72,75] 3 → 73	[0,1] [5,16] 0 → 9	[5,12] [72,73] **9** → 73
overall (25 tasks)	avg.	[0,0] [12,12] 0 → 12	[15,18] [49,50] 16 → 50	[36,38] [57,60] 37 → 58	[51,53] [62,63] 52 → 63	[26,29] [56,58] 28 → 57	[24,25] [48,52] 24 → 50	[35,43] [64,68] 39 → 86	[50,52] [86,87] 51 → 86	[49,51] [70,74] 50 → 72	[61,63] [94,95] **62** → **94**
Navigation											
antmaze-giant	task1	[0,0] [0,0] 0 → 0	[19,20] [73,69] **16** → 56	[4,14] [95,100] 11 → **98**	[0,0] [0,0] 0 → 0	[0,0] [91,92] 3 → 86	[2,8] [98,70] 5 → 64	[0,0] [8,45] 0 → 28	[0,0] [0,63] 0 → 32	[0,0] [0,0] 0 → 0	[0,0] [97,94] 0 → 90
	task2	[0,0] [0,2] 0 → 1	[56,67] [46,84] **60** → 67	[0,33] [98,100] 12 → **99**	[0,0] [14,36] 0 → 26	[21,40] [87,96] 30 → 92	[42,52] [91,97] 46 → 94	[0,0] [66,94] 0 → 82	[0,0] [76,82] 0 → 80	[0,0] [2,8] 0 → 4	[0,0] [83,90] 0 → 86
	task3	[0,0] [0,0] 0 → 0	[0,8] [0,0] **4** → 0	[0,2] [0,0] 1 → 0	[0,0] [0,0] 0 → 0	[0,2] [1,5] 2 → 3	[0,2] [2,11] 0 → 6	[0,0] [2,18] 0 → 10	[0,0] [70,85] 0 → 75	[0,0] [0,0] 0 → 0	[0,0] [75,84] 0 → **80**
	task4	[0,0] [0,0] 0 → 0	[62,80] [84,94] **70** → **89**	[0,2] [24,39] 0 → 74	[0,0] [0,0] 0 → 0	[19,31] [79,89] 23 → 79	[4,14] [75,88] 11 → 81	[0,0] [45,61] 0 → 66	[0,0] [42,54] 0 → 74	[0,0] [0,2] 0 → 2	[0,0] [80,86] 0 → **83**
	task5	[11,15] [2,12] 12 → 8	[76,90] [98,100] **84** → 99	[2,15] [97,100] 8 → 99	[0,8] [60,86] 2 → 73	[46,59] [100,100] 52 → **100**	[25,52] [92,96] 41 → 94	[0,0] [89,96] 0 → 92	[0,0] [90,94] 0 → 92	[5,6] [85,92] 6 → 88	[0,0] [94,100] 0 → 97
	avg.	[2,3] [1,3] 2 → 2	[44,50] [58,65] **47** → 62	[3,11] [64,79] 6 → 74	[0,2] [16,23] 0 → 20	[19,23] [70,74] 22 → 72	[17,24] [66,70] 21 → 68	[0,0] [49,62] 0 → 56	[0,0] [62,79] 0 → 70	[1,1] [14,20] 1 → 19	[0,0] [86,89] 0 → **87**
antsoccer-arena	task1	[15,19] [6,12] 17 → 8	[0,2] [0,0] 0 → 0	[31,48] [78,90] 40 → 84	[44,62] [78,90] 52 → 84	[7,19] [11,51] 13 → 31	[21,34] [47,64] 27 → 56	[15,26] [67,74] 20 → 70	[55,60] [88,96] 57 → 91	[5,9] [93,96] 7 → 94	[85,88] [96,98] **87** → **96**
	task2	[10,24] [2,8] 17 → 6	[0,2] [0,0] 0 → 0	[15,32] [76,90] 24 → **92**	[42,46] [72,82] 44 → 76	[2,12] [2,6] 8 → 4	[6,14] [1,22] 10 → 12	[2,7] [77,86] 5 → 76	[55,57] [91,95] 55 → **93**	[0,4] [83,86] 2 → 84	[70,73] [90,94] **72** → 92
	task3	[4,9] [0,2] 6 → 1	[0,0] [0,0] 0 → 0	[45,48] [89,90] 47 → 92	[44,52] [83,92] **48** → 88	[4,17] [3,8] 11 → 6	[13,16] [9,20] 15 → 16	[3,7] [58,73] 5 → 66	[17,22] [79,89] 20 → 82	[1,4] [65,82] 3 → 74	[35,42] [94,99] 42 → **96**
	task4	[0,4] [0,2] 2 → 1	[0,0] [0,0] 0 → 0	[21,40] [84,92] 32 → **89**	[37,47] [75,80] **42** → 78	[0,3] [0,5] 2 → 2	[9,18] [16,30] 14 → 24	[2,5] [54,65] 4 → 60	[10,22] [60,68] 17 → 64	[2,8] [52,60] 5 → 56	[38,42] [79,88] 40 → 83
	task5	[1,3] [0,3] 3 → 1	[0,0] [0,0] **28** → 2	[11,45] [0,4] 24 → 15	[14,33] [0,2] 0 → 0	[0,2] [0,2] 0 → 2	[0,5] [1,4] 2 → 2	[2,14] [6,28] 8 → 62	[10,18] [74,80] 14 → 80	[0,0] [0,6] 2 → 46	[13,26] [92,96] 19 → **96**
	avg.	[7,11] [2,5] 9 → 4	[0,0] [0,0] 0 → 0	[22,30] [70,74] 34 → 72	[39,45] [64,74] 42 → 68	[5,9] [5,13] 7 → 9	[12,16] [19,25] 14 → 22	[6,10] [64,69] 8 → 67	[31,34] [80,84] 32 → 82	[3,5] [68,74] 4 → 71	[50,54] [92,94] **52** → **93**
humanoidmaze-medium	task1	[16,30] [1,12] 26 → 6	[10,22] [0,0] 16 → 0	[1,14] [10,40] 8 → 26	[34,59] [96,100] 37 → **98**	[8,10] [9,13] 8 → 11	[13,18] [27,45] 16 → 36	[0,0] [0,37] 0 → 18	[4,22] [64,67] 13 → 61	[26,38] [38,51] 32 → 44	[61,68] [94,97] **64** → 95
	task2	[32,50] [9,18] 41 → 15	[7,50] [0,0] 26 → 0	[42,52] [88,95] 47 → 91	[55,93] [98,100] 79 → **99**	[40,57] [77,90] 48 → 84	[16,26] [36,50] 21 → 44	[7,14] [0,48] 11 → 24	[33,44] [67,70] 40 → 72	[45,64] [55,70] 55 → 62	[85,92] [90,96] **88** → 94
	task3	[24,28] [19,24] 26 → 21	[10,46] [0,0] 26 → 0	[34,51] [77,80] 42 → 73	[35,75] [27,100] 56 → 66	[56,68] [79,93] 62 → 86	[43,53] [39,52] 48 → 46	[16,51] [0,3] 37 → 2	[38,50] [80,90] 44 → 93	[73,79] [88,96] 76 → 92	[78,80] [91,100] **84** → **96**
	task4	[0,2] [0,0] 0 → 0	[0,0] [0,0] 3 → 0	[2,8] [8,11] 5 → 8	[0,0] [12,59] 0 → 59	[7,17] [7,12] **12** → 10	[2,6] [10,14] 4 → 12	[0,0] [0,0] 0 → 0	[0,0] [34,54] 0 → 45	[7,13] [15,26] 11 → 25	[68,85] [0,0] 0 → **77**
	task5	[64,73] [26,44] 68 → 37	[11,66] [0,0] 38 → 3	[58,75] [95,98] 68 → 96	[96,100] [99,100] **98** → **100**	[34,44] [75,86] 39 → 81	[24,39] [46,60] 32 → 53	[22,36] [80,88] 29 → 84	[64,70] [91,96] 70 → 94	[58,75] [82,88] 65 → 85	[93,98] [97,100] 96 → 98
	avg.	[29,35] [13,18] 32 → 16	[14,30] [0,1] 22 → 1	[51,57] [74,64] 54 → 59	[46,61] [73,85] 54 → 84	[31,37] [52,57] 34 → 54	[22,24] [35,42] 24 → 38	[11,19] [30,32] 15 → 26	[31,36] [72,72] 34 → 73	[45,51] [59,65] 48 → 62	[65,68] [90,94] **66** → **92**
overall (15 tasks)	avg.	[13,16] [6,8] 15 → 7	[20,26] [20,22] 23 → 21	[22,27] [63,71] 25 → 68	[30,35] [53,62] 32 → 57	[19,22] [43,47] 21 → 45	[19,21] [41,44] 20 → 43	[6,9] [46,53] 8 → 49	[21,23] [72,78] 22 → 75	[16,19] [49,52] 18 → 51	[38,40] [90,92] **39** → **91**

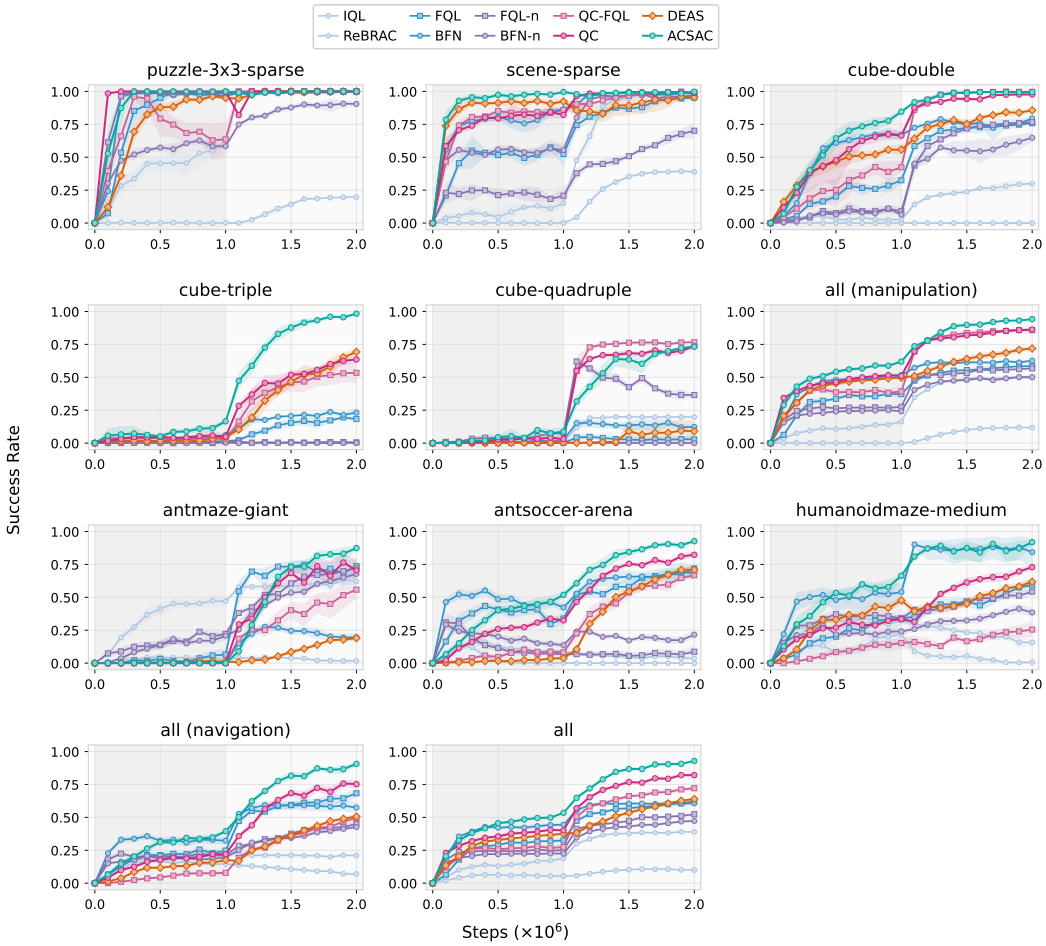


Figure 6: **OGBench offline-to-online RL results by domain.** Each domain panel aggregates five tasks, and the `all` panels average the 25 manipulation tasks, the 15 navigation tasks, and all 40 tasks. The first 1M steps correspond to offline training and the next 1M steps correspond to online fine-tuning. Results are averaged over 4 seeds with 95% confidence intervals.

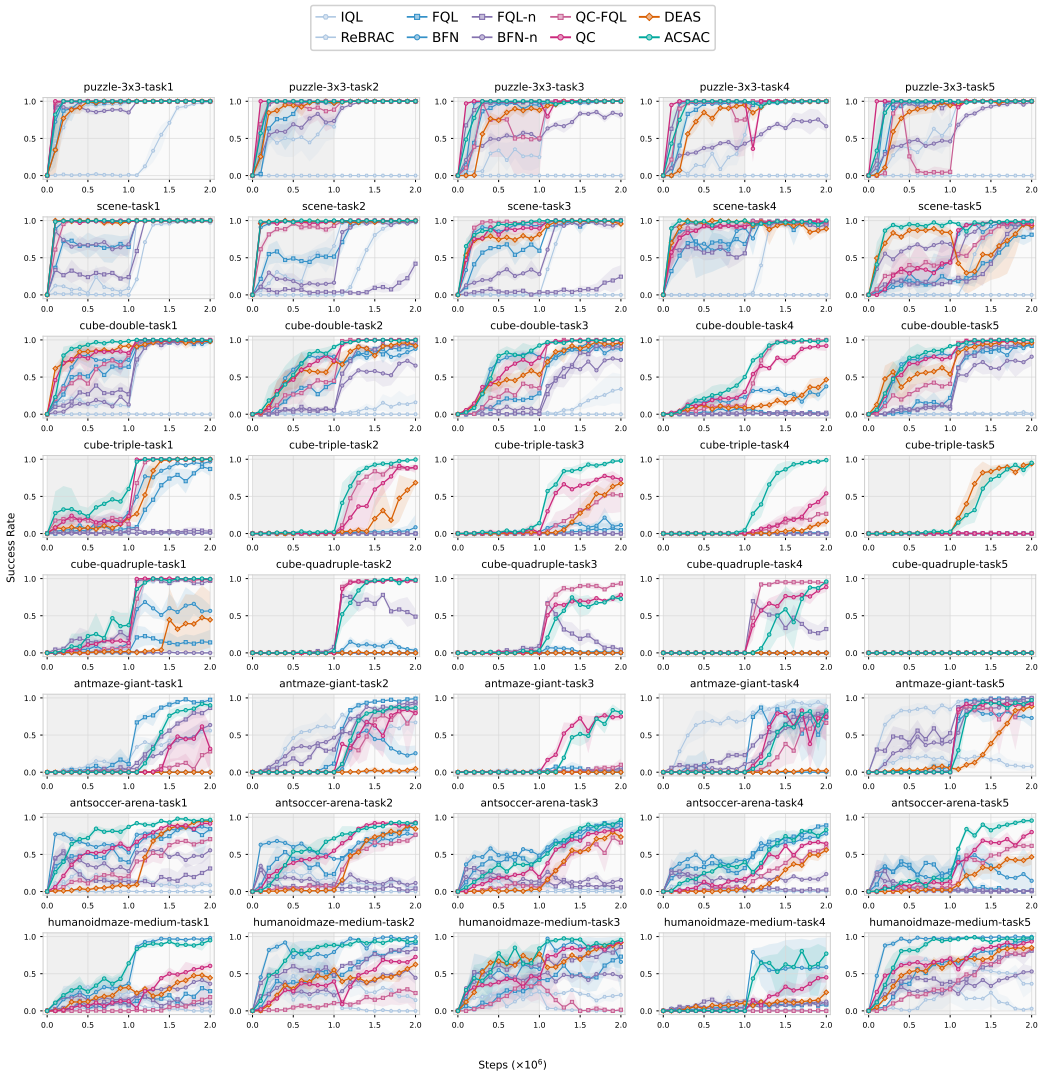


Figure 7: **Complete OGBench offline-to-online RL results by task.** The first 1M steps correspond to offline training and the next 1M steps correspond to online fine-tuning. Results are averaged over 4 seeds with 95% confidence intervals.

D.3 Additional Evaluations with More Seeds

Tab. 7 presents additional evaluations of ACSAC, increasing the number of random seeds per task from 4 to 8. We report mean success rates and 95% confidence intervals at the end of offline training and online fine-tuning.

Table 7: **OGBench offline-to-online RL results with 4 and 8 random seeds.** Entries report mean success rates (%) at the end of offline training → online fine-tuning, with 95% confidence intervals above each mean. Manipulation and navigation averages cover 25 and 15 tasks, respectively.

Domain	`ACSAC` (4-seed)	`ACSAC` (8-seed)
`puzzle-3x3-sparse`	$^{[100,100]}_{100}$ → $^{[100,100]}_{100}$	$^{[100,100]}_{100}$ → $^{[100,100]}_{100}$
`scene-sparse`	$^{[99,100]}_{100}$ → $^{[99,100]}_{100}$	$^{[99,100]}_{100}$ → $^{[99,100]}_{100}$
`cube-double`	$^{[84,86]}_{85}$ → $^{[99,100]}_{100}$	$^{[84,85]}_{85}$ → $^{[99,100]}_{100}$
`cube-triple`	$^{[15,19]}_{17}$ → $^{[97,99]}_{98}$	$^{[16,19]}_{18}$ → $^{[98,99]}_{99}$
`cube-quadruple`	$^{[5,12]}_{9}$ → $^{[72,75]}_{73}$	$^{[5,10]}_{8}$ → $^{[72,74]}_{73}$
`antmaze-giant`	$^{[0,0]}_{0}$ → $^{[86,89]}_{87}$	$^{[0,0]}_{0}$ → $^{[86,89]}_{87}$
`antsoccer-arena`	$^{[50,54]}_{52}$ → $^{[92,94]}_{93}$	$^{[49,54]}_{52}$ → $^{[92,94]}_{93}$
`humanoidmaze-medium`	$^{[65,68]}_{66}$ → $^{[90,94]}_{92}$	$^{[64,67]}_{65}$ → $^{[90,93]}_{92}$
Manipulation Average	$^{[61,63]}_{62}$ → $^{[94,95]}_{94}$	$^{[61,62]}_{62}$ → $^{[94,95]}_{94}$
Navigation Average	$^{[38,40]}_{39}$ → $^{[90,92]}_{91}$	$^{[38,40]}_{39}$ → $^{[90,91]}_{91}$

D.4 Benefit of State-Dependent Chunk Size Selection

We examine state-dependent chunk size selection on `cube-triple` by changing only the chunk size rule after online fine-tuning, keeping the same flow BC policy and critic for each task and seed. Fixed-h uses a constant $h \in \{1, \dots, 5\}$, while uniform selection samples h uniformly from this set at each replanning state. Both select the highest-value length-h prefix among $N = 4$ candidate chunks of length $H = 5$. In Tab. 8, adaptive selection achieves 98% success, compared with 32–69% for fixed chunk sizes and 43% for uniform selection, highlighting the benefit of selecting the chunk size according to the current state.

Table 8: **Effect of chunk size selection on `cube-triple`.** All variants use the same trained ACSAC checkpoints. Success rates (%) are averaged over 5 tasks, 4 seeds, and 50 episodes per task and seed.

Chunk size	Adaptive	$h = 1$	$h = 2$	$h = 3$	$h = 4$	$h = 5$	Uniform
Success (%)	**98**	32	37	54	69	63	43

D.5 Adaptive Execution with a Larger Maximum Chunk Size

ACSAC also achieves strong performance with a larger maximum chunk size of $H = 10$ on `cube-triple-task4`, a challenging multi-object manipulation task. The online checkpoint used here achieves a 98% success rate over 50 evaluation episodes, after 1M offline updates and 1M online environment steps.

In Fig. 8, ACSAC selects long chunks ($h^\star = 10$) for coarse approach and transport, and short chunks ($h^\star = 1$–2) for fine alignment, grasping, and the final placement adjustment. This pattern illustrates how ACSAC exploits a larger maximum horizon for sustained coarse motion while retaining frequent replanning for precise manipulation.

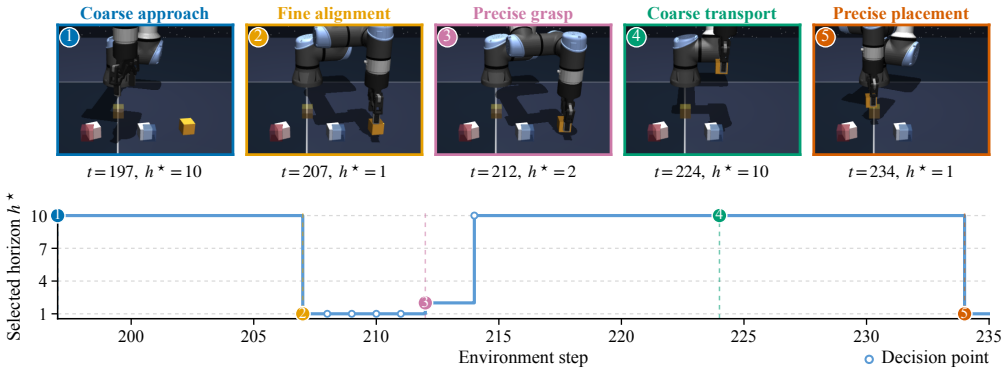


Figure 8: **Qualitative rollout of ACSAC with $H = 10$.** Selected execution horizon $h^\star$ during a pick-and-place segment on `cube-triple-task4`. Numbered observations (top) correspond to the marked replanning decisions (bottom).

E THEORETICAL FOUNDATIONS

We analyze ACSAC as behavior-constrained variable-duration Bellman learning. A length-H chunk drawn from the flow BC policy is treated as a proposal path. At each replanning state, ACSAC may execute any prefix of this path and replan afterwards. We establish four results. First, prefix-conditioned Q-values of different lengths are well-defined under prefix consistency (Proposition E.3) and share the unit of total discounted return under a common continuation policy (Theorem 1). Second, the critic loss in Eq. (5) approximates a γ-contractive Bellman backup whose unique fixed point is the action-value function of the corresponding adaptive policy $\pi_\star$ (Theorem 2). Third, prefix truncation cannot increase the action-level mismatch between the flow BC policy and the behavior chunk distribution at the corresponding prefix length (Proposition E.12). Fourth, averaging per-horizon squared losses tightens a common gradient-variance bound under the stated covariance conditions, while preserving the sparse reward signal in each per-horizon target (Lemma E.15).

We work at the level of exact expectations, with deterministic transitions except in Appendix E.6. The Bellman analysis holds the flow BC policy, N, and H fixed; it does not establish convergence of joint neural critic and flow BC policy training. Appendix E.6 gives a semi-Markov interpretation of the executed prefixes and extends the Bellman results to stochastic transitions (Proposition E.18).

E.1 SETUP AND NOTATION

Assumption E.1 (Deterministic MDP for analysis)**.** For the proofs we consider the deterministic version of the MDP in Sec. 3, with $s_{t+1} = f(s_t, a_t)$ and bounded rewards $|r(s,a)| \le R_{\max}$, $\gamma \in (0,1)$. We use the sup-norm $\|Q\|_\infty := \sup_{s,a_{t:t+h}} |Q(s, a_{t:t+h})|$ on bounded Q-functions, and write the proofs for finite or discretized state-action spaces. The same contraction arguments extend to bounded continuous spaces when the displayed maxima exist; otherwise, maxima can be replaced by suprema. The OGBench tasks evaluated in Sec. 5 are deterministic.

Variable-length action chunks. We use the body's chunk notation $a_{t:t+h} := (a_t, a_{t+1}, \dots, a_{t+h-1}) \in \mathcal{A}^h$ throughout this appendix, with $h \in [H] := \{1, \dots, H\}$ and $[N] := \{1, \dots, N\}$. We write

$$\mathcal{A}^{\le H} := \bigcup_{h=1}^{H} \mathcal{A}^h \tag{9}$$

for the set of executable prefixes, with the convention that an element carries its own length h. Under Assumption E.1, executing $a_{t:t+h}$ from s_t produces the unique trajectory $s_{t+1} = f(s_t, a_t), \dots, s_{t+h} = f(s_{t+h-1}, a_{t+h-1})$, with open-loop chunk return

$$r_t^{(h)} := \sum_{\tau=0}^{h-1} \gamma^\tau \, r(s_{t+\tau}, a_{t+\tau}), \tag{10}$$

which is exactly the cumulative-reward term in the h-step target G_h in Eq. (4).

Behavior and proposal distributions. Let $\pi_\beta^H(\cdot \mid s_t) \in \Delta(\mathcal{A}^H)$ denote the true behavior chunk distribution under the data-collection policy of $\mathcal{D}$, and let $\pi_\theta^H(\cdot \mid s_t) \in \Delta(\mathcal{A}^H)$ denote the full-length law of the flow BC policy of Sec. 4.3. For $h \in [H]$, $\pi_\theta^h(\cdot \mid s_t)$ denotes the distribution of the first h actions when a full chunk is sampled from $\pi_\theta^H(\cdot \mid s_t)$,

$$\pi_\theta^h(a_{t:t+h} \mid s_t) := \int_{\mathcal{A}^{H-h}} \pi_\theta^H(a_{t:t+H} \mid s_t) \, da_{t+h:t+H}, \tag{11}$$

and π_β^h is defined analogously from π_β^H. The integral becomes a sum in the discrete case.

Equal-value candidates. When several candidates attain the same maximum value, we choose one using a fixed state-independent rule, such as the first candidate in the stored order. The contraction and fixed-point conclusions below do not depend on how ties are resolved; the rule makes the policy extraction in Definition E.8 below single-valued and is the rule assumed in Remark E.14.

E.2 Prefix Values Are Well-Defined and Cross-Horizon Comparable

For ACSAC's joint argmax over (n, h) to be meaningful, two conditions are needed. The network output at the h-th position must represent a well-defined Q-value of the length-h prefix, and the resulting Q-values across prefix lengths must lie on a single comparable scale.

Definition E.2 (Prefix consistency)**.** A network $\hat{Q} : \mathcal{S} \times \mathcal{A}^H \to \mathbb{R}^H$ is *prefix-consistent* if its h-th output takes the same value on any two length-H chunks that share the same first h actions. Equivalently, $\hat{Q}^{(h)}(s, a_{t:t+H})$ depends only on $(s, a_{t:t+h})$ and not on the suffix $a_{t+h:t+H}$.

Proposition E.3 (Causal-masked Transformer is sufficient)**.** *A decoder-only Transformer with causal self-attention and one output head per position satisfies prefix consistency.*

Proof. Index the input tokens as $(s, a_t, \ldots, a_{t+H-1})$ with action a_{t+h-1} at position h. The causal mask restricts the hidden representation at position h to attend only to positions $0, \ldots, h$, and induction over layers preserves this restriction. The h-th output head reads only this hidden representation, so its value depends only on $(s, a_{t:t+h})$. □

Remark E.4 (Sufficient, not necessary). A generic MLP that takes the entire chunk as input does not enforce prefix consistency by architecture. It could in principle learn the suffix-invariance from data, but the property is not guaranteed. ACSAC relies on the causal Transformer for this property by construction, not on data-driven invariance.

Variable-horizon Bellman semantics. In Theorem 1, V^π is the state-value function of the continuation policy π starting at a replanning state. The policy may itself select variable-length action chunks. Prefix consistency ensures that each network output depends only on the prefix whose value it estimates.

Proof of Theorem 1. Split the discounted return of executing $a_{t:t+h}$ and then following π into the first h rewards and the remaining return. Under Assumption E.1, the prefix fixes $r_t^{(h)}$ and s_{t+h}, and the expected remaining return is $V^\pi(s_{t+h})$, discounted by γ^h. This gives Eq. (3); every h thus measures the total discounted return from the same s_t under the same continuation policy.

For a full chunk and any $h_1 \le h_2 \in [H]$, writing the additional reward over the segment $[h_1, h_2)$ as

$$r_{t+h_1}^{(h_2-h_1)} := \sum_{j=0}^{h_2-h_1-1} \gamma^j r(s_{t+h_1+j}, a_{t+h_1+j}), \tag{12}$$

we obtain the difference identity

$$\begin{aligned} &Q^\pi(s_t, a_{t:t+h_2}) - Q^\pi(s_t, a_{t:t+h_1}) \\ &\quad = \gamma^{h_1}\Big(r_{t+h_1}^{(h_2-h_1)} + \gamma^{h_2-h_1} V^\pi(s_{t+h_2}) - V^\pi(s_{t+h_1})\Big). \end{aligned} \tag{13}$$

Substitute $h = h_1, h_2$ into Eq. (3) and split $r_t^{(h_2)} - r_t^{(h_1)} = \gamma^{h_1} r_{t+h_1}^{(h_2-h_1)}$ to obtain Eq. (13). □

Remark E.5 (Replanning-horizon interpretation). A longer prefix is preferred whenever the extra open-loop segment plus delayed continuation exceeds the value of replanning at s_{t+h_1}. ACSAC's joint argmax over (n, h) in Eq. (8) therefore compares replanning horizons on a common return scale.

E.3 Expected-Prefix-Max Bellman Backup

We now show that the per-horizon target inside the critic loss in Eq. (5) is an unbiased Monte Carlo sample of an expected-prefix-max Bellman backup $\mathcal{B}_\theta^{N,H}$ that is a γ-contraction in sup-norm. Its unique fixed point is the action-value function of the corresponding idealized adaptive-prefix policy $\pi_\star$. The structure mirrors EMaQ (Ghasemipour et al., 2021), which samples N behavior actions and backs up their max; ACSAC samples N behavior chunks and backs up the max over all NH executable prefixes.

Definition E.6 (ACSAC expected-prefix-max Bellman backup)**.** For bounded $Q : \mathcal{S} \times \mathcal{A}^{\leq H} \to \mathbb{R}$, the *ACSAC Bellman backup* is

$$(\mathcal{B}_\theta^{N,H} Q)(s_t, a_{t:t+h}) \;:=\; r_t^{(h)} \\ +\; \gamma^h\, \mathbb{E}_{\tilde{a}_{t+h:t+h+H}^{(1:N)} \sim \pi_\theta^H(\cdot|s_{t+h})} \Big[\max_{n\in[N],\, k\in[H]} Q\Big(s_{t+h},\, \tilde{a}_{t+h:t+h+k}^{(n)}\Big) \Big]. \tag{14}$$

The tilde distinguishes the N i.i.d. proposal chunks at the bootstrap state s_{t+h} from the chunk $a_{t:t+H}$ on the regression side, and the maximum ranges over the NH candidate prefixes formed from these proposals.

Lemma E.7 (Max non-expansiveness over a common index set)**.** *For any finite index set $\mathcal{I}$ and any real-valued $g_1, g_2 : \mathcal{I} \to \mathbb{R}$,*

$$\big|\max_{i\in\mathcal{I}} g_1(i) - \max_{i\in\mathcal{I}} g_2(i)\big| \;\leq\; \max_{i\in\mathcal{I}} |g_1(i) - g_2(i)|. \tag{15}$$

Proof. Let $i_1 \in \arg\max_i g_1(i)$ and $i_2 \in \arg\max_i g_2(i)$. Then $\max_i g_1(i) - \max_i g_2(i) \leq g_1(i_1) - g_2(i_1) \leq |g_1(i_1) - g_2(i_1)| \leq \max_i |g_1(i) - g_2(i)|$, and the symmetric direction yields the absolute value. This is a deterministic, set-theoretic inequality, so it holds even when the indexed values $g_1(i), g_2(i)$ are correlated across i, which is what allows EMaQ's contraction argument to transport to ACSAC's NH correlated prefix candidates (Ghasemipour et al., 2021). □

Proof of the γ-contraction in Theorem 2. Fix $(s_t, a_{t:t+h})$ and let $s' = s_{t+h}$. The shared $r_t^{(h)}$ cancels, so $|(\mathcal{B}_\theta^{N,H} Q_1)(s_t, a_{t:t+h}) - (\mathcal{B}_\theta^{N,H} Q_2)(s_t, a_{t:t+h})| = \gamma^h \big|\mathbb{E}[\max_{n,k} Q_1(s', \cdot)] - \mathbb{E}[\max_{n,k} Q_2(s', \cdot)]\big|$. By Jensen's inequality and Lemma E.7 applied pointwise to each realized proposal sample, $\big|\mathbb{E}[\max Q_1] - \mathbb{E}[\max Q_2]\big| \leq \mathbb{E}\big[\max_{n,k} |Q_1(s', \cdot) - Q_2(s', \cdot)|\big] \leq \|Q_1 - Q_2\|_\infty$. Hence $|(\mathcal{B}_\theta^{N,H} Q_1) - (\mathcal{B}_\theta^{N,H} Q_2)| \leq \gamma^h \|Q_1 - Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty$. By the Banach fixed-point theorem, $\mathcal{B}_\theta^{N,H}$ has a unique fixed point $Q_\theta^{N,H}$, and repeated application of $\mathcal{B}_\theta^{N,H}$ converges to $Q_\theta^{N,H}$ from any bounded initial Q-function. □

Definition E.8 (Critic-induced extraction policy)**.** For any bounded Q, the extraction policy $\pi_\star^Q$ acts at state s_t as follows. Draw $a_{t:t+H}^{(1)}, \ldots, a_{t:t+H}^{(N)} \overset{\text{i.i.d.}}{\sim} \pi_\theta^H(\cdot \mid s_t)$, set

$$(n^\star, h^\star) \;=\; \arg\max_{n\in[N],\, h\in[H]} Q\Big(s_t,\, a_{t:t+h}^{(n)}\Big), \tag{16}$$

and execute the prefix $a_{t:t+h^\star}^{(n^\star)}$, treating it as one temporally extended action. We write $\pi_\star := \pi_\star^{Q_\theta^{N,H}}$ for the idealized adaptive-prefix policy induced by the operator fixed point; the deployed implementation in Sec. 4.3 applies the same extraction rule to the learned critic Q_ϕ.

Proof of the fixed-point identity in Theorem 2. Eq. (16) gives, for any draw of the N proposals, $Q_\theta^{N,H}(s_t, a_{t:t+h^\star}^{(n^\star)}) = \max_{n\in[N],\, k\in[H]} Q_\theta^{N,H}(s_t, a_{t:t+k}^{(n)})$. Taking the expectation over the proposal draws, the inner expected max in Eq. (14) equals $\mathbb{E}_{\pi_\star}[Q_\theta^{N,H}(s_t, \cdot)]$, so the fixed-point equation $Q_\theta^{N,H} = \mathcal{B}_\theta^{N,H} Q_\theta^{N,H}$ reads

$$Q_\theta^{N,H}(s_t, a_{t:t+h}) \;=\; r_t^{(h)} + \gamma^h\, \mathbb{E}_{a'\sim\pi_\star(\cdot|s_{t+h})}\big[Q_\theta^{N,H}(s_{t+h}, a')\big].$$

This is the variable-horizon Bellman equation for $\pi_\star$ and uniquely identifies its action-value function by the same contraction argument as Theorem 2, with the fixed policy $\pi_\star$ in place of the joint max. We may therefore write the fixed point as $Q^{\pi_\star}$ in the rest of the appendix. Together with Theorem 1, this shows that ACSAC's fixed-point prefix values share the same adaptive continuation $V^{\pi_\star}$. □

Proposition E.9 (Critic targets are Monte Carlo Bellman samples)**.** *Fix any bounded $Q : \mathcal{S} \times \mathcal{A}^{\leq H} \to \mathbb{R}$ and define the per-horizon target*

$$\hat{G}_h(Q) \;:=\; r_t^{(h)} \;+\; \gamma^h \max_{n\in[N],\, k\in[H]} Q\Big(s_{t+h},\, \tilde{a}_{t+h:t+h+k}^{(n)}\Big), \tag{17}$$

with $\tilde{a}^{(1)}_{t+h:t+h+H}, \ldots, \tilde{a}^{(N)}_{t+h:t+h+H} \overset{\text{i.i.d.}}{\sim} \pi^H_\theta(\cdot \mid s_{t+h})$. $\hat{G}_h(Q)$ *is the explicit-max sample form of the body's bootstrap target* G_h *in Eq.* (4)*, with* $\pi_\star$ *replaced by the joint* $\arg\max$ *and the bootstrap critic treated as a free argument; the body's* $G_h(s_t, a_{t:t+h})$ *corresponds to* $\hat{G}_h(Q_{\bar{\phi}})$ *at this conditioning point. Under Assumption E.1,*

$$\mathbb{E}\Big[\hat{G}_h(Q) \,\Big|\, s_t, a_{t:t+h}\Big] \;=\; (\mathcal{B}^{N,H}_\theta Q)(s_t, a_{t:t+h}), \tag{18}$$

and the conditional squared-error loss decomposes as

$$\begin{aligned}&\mathbb{E}\Big[(Q_\phi(s_t, a_{t:t+h}) - \hat{G}_h(Q))^2 \,\Big|\, s_t, a_{t:t+h}\Big] \\ &\quad= \big(Q_\phi(s_t, a_{t:t+h}) - (\mathcal{B}^{N,H}_\theta Q)(s_t, a_{t:t+h})\big)^2 + \mathrm{Var}\Big[\hat{G}_h(Q) \,\Big|\, s_t, a_{t:t+h}\Big].\end{aligned} \tag{19}$$

Hence the empirical critic loss in Eq. (5) *is a Monte Carlo regression toward the Bellman backup* $\mathcal{B}^{N,H}_\theta Q$*, not an exact squared Bellman residual for any single sampled target. In the tabular exact-expectation regime, fitted iteration with* $\mathcal{B}^{N,H}_\theta$ *converges to* $Q^{N,H}_\theta$.

Proof. Eq. (18) follows from Definition E.6, since under Assumption E.1 the rewards $\{r_{t+\tau}\}_{\tau<h}$ summed in $r^{(h)}_t$ are deterministic given $(s_t, a_{t:t+h})$, and the only randomness in $\hat{G}_h(Q)$ comes from the proposal sampling that defines $\mathcal{B}^{N,H}_\theta Q$. The decomposition in Eq. (19) is the standard bias-variance identity for the squared error of a deterministic prediction $Q_\phi(s_t, a_{t:t+h})$ against a stochastic target $\hat{G}_h(Q)$ with mean $(\mathcal{B}^{N,H}_\theta Q)(s_t, a_{t:t+h})$. □

Remark E.10 (Bounded fixed-point values). With rewards in $[-R_{\max}, R_{\max}]$, the contraction in Theorem 2 maps the interval $[-R_{\max}/(1-\gamma),\, R_{\max}/(1-\gamma)]$ to itself pointwise, so $Q^{N,H}_\theta$ is bounded by the reward scale uniformly in $(s_t, a_{t:t+h})$. This is an operator-level statement; the iterates of a learned neural critic may transiently exceed this interval without explicit clipping.

E.4 Prefix Truncation Does Not Increase Behavior Mismatch

ACSAC's bootstrap target queries the critic at an adaptively-chosen prefix of a behavior-generated chunk. A natural concern is whether shorter prefixes drift further from the behavior support than the full chunk does. We rule this out at the level of total variation distance, using only the elementary fact that marginalization is non-expansive in TV.

Lemma E.11 (Marginalization is TV non-expansive)**.** *For any two distributions* μ, ν *on* $\mathcal{A}^H$ *and any* $h \in [H]$*, let* μ_h, ν_h *denote their length-*h *prefix marginals as in Eq.* (11)*. Then*

$$D_{\mathrm{TV}}(\mu_h, \nu_h) \;\le\; D_{\mathrm{TV}}(\mu, \nu). \tag{20}$$

Proof. Using the convention $D_{\mathrm{TV}}(p, q) = \frac{1}{2}\sum_x |p(x) - q(x)|$ and the triangle inequality, $\sum_{a_{t:t+h}} \big|\sum_{a_{t+h:t+H}} (\mu - \nu)(a_{t:t+H})\big| \le \sum_{a_{t:t+H}} |\mu(a_{t:t+H}) - \nu(a_{t:t+H})|$. Dividing by 2 gives Eq. (20). The continuous case is analogous, with sums replaced by integrals. □

Proposition E.12 (Prefix truncation and behavior mismatch)**.** *Let* π^H_θ *and* π^H_β *be the flow BC and behavior chunk distributions, and let* π^h_θ *and* π^h_β *be their length-*h *prefix marginals. For every* $h \in [H]$*,*

$$D_{\mathrm{TV}}\big(\pi^h_\theta(\cdot \mid s_t), \pi^h_\beta(\cdot \mid s_t)\big) \le D_{\mathrm{TV}}\big(\pi^H_\theta(\cdot \mid s_t), \pi^H_\beta(\cdot \mid s_t)\big). \tag{21}$$

Proof of Proposition E.12. Apply Lemma E.11 with $\mu = \pi^H_\theta(\cdot \mid s_t)$ and $\nu = \pi^H_\beta(\cdot \mid s_t)$. □

Remark E.13 (Implication for the TD target). Proposition E.12 concerns the proposal distribution before value-based selection. Shortening a full behavior-generated chunk to any prefix does not increase its action-level mismatch from the corresponding behavior prefix marginal. The set of prefixes used by the bootstrap target in Eq. (5) is therefore obtained from behavior-generated chunks without any length-dependent shift caused by truncation itself.

Remark E.14 (Selection adds a separate distribution shift). Value-based rejection sampling does change the selected action distribution. QC analyzes the behavior constraint induced by rejection sampling from a fixed proposal distribution (Li et al., 2025b). Fix s_t and a bounded Q. For a full-length chunk $a = a_{t:t+H}$, let $h_Q(a) := \min \arg\max_{h\in[H]} Q(s_t, a_{t:t+h})$ and $T_Q(a) := a_{t:t+h_Q(a)} \in \mathcal{A}^{\le H}$, so that T_Q keeps the highest-value prefix of a and resolves equal values toward the shorter prefix. Since $\max_{n\in[N],\, h\in[H]} Q(s_t, a^{(n)}_{t:t+h}) = \max_{n\in[N]} Q(s_t, T_Q(a^{(n)}))$, the joint argmax in Eq. (16) under the tie rule for equal-value candidates in Appendix E.1 is equivalent to a two-step selection. It first picks the candidate with the highest score $Q(s_t, T_Q(a^{(n)}))$, equal scores going to the smaller index, and then applies T_Q to the selected chunk. Let $\pi^H_{N\text{-best}}(\cdot \mid s_t)$ denote the distribution of the chunk selected in this way, and write $(T_Q)_{\#}\mu$ for the distribution of $T_Q(a)$ when $a \sim \mu$. The extraction policy of Definition E.8 then satisfies $\pi^Q_\star = (T_Q)_{\#}\pi^H_{N\text{-best}}$, and the data-processing inequality together with QC's closed-form bound (Li et al., 2025b) gives

$$D_{\mathrm{KL}}\big(\pi^Q_\star(\cdot \mid s_t) \,\big\|\, (T_Q)_{\#}\pi^H_\theta(\cdot \mid s_t)\big) \;\le\; D_{\mathrm{KL}}\big(\pi^H_{N\text{-best}}(\cdot \mid s_t) \,\big\|\, \pi^H_\theta(\cdot \mid s_t)\big) \;\le\; \log N - \frac{N-1}{N}. \tag{22}$$

Ties are broken by candidate index and not by the actions themselves, so the selected chunk, given its score, is distributed like a single proposal chunk with that score. The last inequality in Eq. (22) therefore also holds when scores can coincide. The upper bound in Eq. (22) depends only on the number N of sampled chunks, not on the NH scored prefixes, and is uniform over H. The reference distribution $(T_Q)_{\#}\pi^H_\theta$ in Eq. (22) is the full-length law of the flow BC policy truncated by the same rule. The bound therefore compares $\pi^Q_\star$ with this truncated proposal law, not with the fixed-length prefix marginal π^h_θ of Eq. (11) at the selected length. By contrast, Proposition E.12 concerns prefix truncation before value-based selection. The mismatch between replay trajectories and open-loop execution is a separate issue, discussed briefly at the end of Appendix E.6.

E.5 Per-Horizon Loss Averaging Stabilizes Updates

The multi-step targets G_h in Eq. (5) accumulate rewards over h steps, and the variance of this cumulative sum tends to grow with h. ACSAC averages all H per-horizon squared losses at the gradient level, in line with T-SAC's gradient-level averaging (Tian et al., 2026) and TOP-ERL's multi-horizon Transformer supervision (Li et al., 2025a). A standard variance-of-mean argument explains this design choice.

Cumulative-reward variance. For the cumulative-reward part of G_h, expansion gives

$$\mathrm{Var}\Big[r^{(h)}_t\Big] \;=\; \sum_{i=0}^{h-1}\sum_{j=0}^{h-1} \gamma^{i+j}\,\mathrm{Cov}(r_{t+i},\, r_{t+j}), \tag{23}$$

which typically increases with h, especially when γ is close to 1 and per-step rewards are positively correlated.

Per-horizon gradient. Let $\delta_h(\phi) := Q_\phi(s_t, a_{t:t+h}) - G_h$ denote the per-horizon Bellman residual, and let $g_h := 2\delta_h(\phi)\,\nabla_\phi Q_\phi(s_t, a_{t:t+h})$ denote its per-horizon loss gradient. The averaged loss $\frac{1}{H}\sum_h \delta_h^2$ in Eq. (5) produces the gradient $\bar g := \frac{1}{H}\sum_{h=1}^H g_h$, and let $\tilde g_h := g_h - \mathbb{E}[g_h]$ denote its centered version.

Lemma E.15 (Variance reduction by averaging)**.** *Let $H \ge 2$ and $\sigma^2 > 0$. Suppose the centered per-horizon gradients satisfy $\mathbb{E}[\|\tilde g_h\|^2] \le \sigma^2$ for every h and $\mathbb{E}[\langle \tilde g_h, \tilde g_{h'}\rangle] \le \rho\,\sigma^2$ for some $\rho \in [-1/(H-1), 1]$ and every $h \ne h'$. Then the centered average $\bar{\tilde g} := \frac{1}{H}\sum_h \tilde g_h$ satisfies*

$$\mathrm{Var}(\bar g) \;:=\; \mathbb{E}\Big[\big\|\bar{\tilde g}\big\|^2\Big] \;\le\; \sigma^2\left[\rho + \frac{1-\rho}{H}\right]. \tag{24}$$

Whenever $\rho < 1$, this is strictly less than σ^2, the common upper bound on the per-horizon gradient variances.

Proof. Expand $\|\bar{\tilde g}\|^2 = H^{-2}\sum_{h,h'}\langle \tilde g_h, \tilde g_{h'}\rangle$ and take expectations. The diagonal terms contribute at most $H^{-2}\cdot H\sigma^2 = \sigma^2/H$, and the off-diagonal terms contribute at most $H^{-2}\cdot H(H-1)\rho\sigma^2$, which sum to the right-hand side of Eq. (24). □

Remark E.16 (Imperfect correlation in ACSAC). This is the setting targeted by T-SAC's gradient-level averaging argument (Tian et al., 2026). The per-horizon gradients g_h in ACSAC share the same critic parameters ϕ and may be correlated. Their residuals δ_h use different cumulative-reward sums and different bootstrap states s_{t+h}, so their gradients need not be perfectly aligned. When the covariance condition holds with $\rho < 1$, Lemma E.15 gives a tighter common variance bound; it does not imply lower variance than every individual single-horizon gradient. Target-level averaging, in contrast, would collapse the distinct per-horizon signals.

Remark E.17 (Sparse reward signals are preserved). ACSAC averages per-horizon *losses*, not the targets themselves. A non-zero reward $r_{t+\tau}$ enters every G_h with $h > \tau$ as a discounted summand, so the sparse signal propagates into all $H - \tau$ residuals simultaneously. This contrasts with target-level averaging, which would replace the per-horizon targets by a single averaged scalar and could dilute sparse rewards (Tian et al., 2026).

E.6 Semi-Markov Interpretation and Stochastic Extension

Semi-Markov interpretation. At each replanning state s_{t_m}, ACSAC selects a prefix of length h_m and commits to its actions until the next replanning state at $t_{m+1} = t_m + h_m$. The selected prefix is one temporally extended action of duration h_m, with open-loop chunk return $r_{t_m}^{(h_m)}$ and continuation discount γ^{h_m}. The decisions at replanning states thus form a semi-Markov decision process (SMDP) over $\mathcal{S}$ whose actions are the executable prefixes (Sutton et al., 1999). DEAS likewise treats a fixed-length action sequence as a single decision unit (Kim et al., 2026). Regrouping the per-step rewards at replanning states leaves the discounted return unchanged, since $\gamma^{t_m}\gamma^j = \gamma^{t_m+j}$. Hence V^π and Q^π in Theorem 1 measure the original per-step discounted return, evaluated from replanning states.

Stochastic transitions. We now drop the deterministic dynamics in Assumption E.1 and keep its other conventions, so that $s_{t+\tau+1} \sim T(\cdot \mid s_{t+\tau}, a_{t+\tau})$ for $\tau = 0, \dots, h-1$ with T the transition dynamics of Sec. 3. Executing $a_{t:t+h}$ from s_t now produces a random state sequence $s_{t+1:t+h+1} = (s_{t+1}, \dots, s_{t+h})$, so the chunk return $r_t^{(h)}$ in Eq. (10) and the bootstrap state s_{t+h} depend on the sampled transitions. The backup of Definition E.6 takes an outer expectation over this sequence,

$$\begin{aligned}(\mathcal{B}_\theta^{N,H} Q)(s_t, a_{t:t+h}) \;:=\; & \mathbb{E}_{s_{t+1:t+h+1} \mid s_t,\, a_{t:t+h}}\Big[\, r_t^{(h)} \\ & +\; \gamma^h\, \mathbb{E}_{\tilde{a}_{t+h:t+h+H}^{(1:N)} \sim \pi_\theta^H(\cdot \mid s_{t+h})}\Big[\, \max_{n\in[N],\, k\in[H]} Q\Big(s_{t+h},\, \tilde{a}_{t+h:t+h+k}^{(n)}\Big)\Big]\Big].\end{aligned} \tag{25}$$

The outer expectation fixes s_t and the whole prefix $a_{t:t+h}$ first and then samples the h transitions, and the inner expectation samples the N proposal chunks at the realized s_{t+h} exactly as in Definition E.6. Under Assumption E.1 the outer expectation is trivial and Eq. (25) reduces to Eq. (14), so we keep the symbols $\mathcal{B}_\theta^{N,H}$ and $Q_\theta^{N,H}$.

Proposition E.18 (Stochastic transitions)**.** *Let transitions be stochastic as above, and fix the flow BC policy π_θ, rejection sampling size N, and maximum chunk size H. (i) Let π be any continuation policy whose distribution over executed prefixes at a replanning state depends only on that state, such as the extraction policy $\pi_\star^Q$ of Definition E.8 for any bounded Q. Then Eq.* (3) *holds with its right-hand side replaced by the expectation over the executed transitions,*

$$Q^\pi(s_t, a_{t:t+h}) \;=\; \mathbb{E}_{s_{t+1:t+h+1} \mid s_t,\, a_{t:t+h}}\Big[\, r_t^{(h)} + \gamma^h\, V^\pi(s_{t+h})\Big]. \tag{26}$$

(ii) The backup $\mathcal{B}_\theta^{N,H}$ in Eq. (25) *is a γ-contraction in sup-norm on bounded $Q : \mathcal{S} \times \mathcal{A}^{\le H} \to \mathbb{R}$. Its unique fixed point $Q_\theta^{N,H}$ equals $Q^{\pi_\star}$, where $\pi_\star = \pi_\star^{Q_\theta^{N,H}}$ is the extraction policy of Definition E.8 induced by this fixed point.*

Proof. For (i), split the return of executing $a_{t:t+h}$ and then following π into the first h rewards and the remaining return. Conditional on the realized state sequence $s_{t+1:t+h+1}$, the first h rewards are fixed. The transitions are Markov and π chooses at s_{t+h} from that state alone, so the expected remaining return is $V^\pi(s_{t+h})$. Taking the expectation over the executed transitions gives Eq. (26).

For the contraction in (ii), fix $(s_t, a_{t:t+h})$. The reward term cancels in $\mathcal{B}_\theta^{N,H} Q_1 - \mathcal{B}_\theta^{N,H} Q_2$. Jensen's inequality moves the absolute value inside both expectations, and Lemma E.7 applied to

each realized $(s_{t+h}, \tilde{a}^{(1:N)}_{t+h:t+h+H})$ bounds the integrand by $\|Q_1 - Q_2\|_\infty$. Hence $|(\mathcal{B}^{N,H}_\theta Q_1 - \mathcal{B}^{N,H}_\theta Q_2)(s_t, a_{t:t+h})| \le \gamma^h \|Q_1 - Q_2\|_\infty \le \gamma \|Q_1 - Q_2\|_\infty$, and the Banach fixed-point theorem applies as in Theorem 2.

For the fixed-point identity in (ii), the proposals in Eq. (25) are drawn at the realized s_{t+h} from the same law as the fresh proposals of $\pi_\star$. As in the proof of Theorem 2, the inner expected maximum of $Q^{N,H}_\theta$ therefore equals $\mathbb{E}_{a' \sim \pi_\star(\cdot|s_{t+h})}[Q^{N,H}_\theta(s_{t+h}, a')]$ at every realized s_{t+h}. Substituting this into $Q^{N,H}_\theta = \mathcal{B}^{N,H}_\theta Q^{N,H}_\theta$ shows that $Q^{N,H}_\theta$ satisfies the Bellman evaluation equation of the fixed policy $\pi_\star$. This equation has a unique bounded solution, since its right-hand side is a γ-contraction by the argument above with $\pi_\star$ held fixed in place of the joint maximum. By (i) and $V^{\pi_\star}(s) = \mathbb{E}_{a' \sim \pi_\star(\cdot|s)}[Q^{\pi_\star}(s, a')]$, the function $Q^{\pi_\star}$ satisfies the same equation. Uniqueness gives $Q^{N,H}_\theta = Q^{\pi_\star}$. □

With $H = 1$, Eq. (25) is exactly EMaQ's expected-max Q operator over single actions, including its expectation over the next state (Ghasemipour et al., 2021). Proposition E.18 concerns the backup operator, not its estimation from data. Whether the sampled targets $\hat{G}_h$ of Proposition E.9 estimate this backup under stochastic transitions depends on how the data were collected. A logged trajectory, conditioned on its recorded actions, need not have the same law as executing those actions open loop from the same state. DQC bounds the resulting gap between the nominal value estimated from such data and the actual value of open-loop execution in terms of the open-loop consistency of the dataset (Li et al., 2026).