LEARNING CHUNK VALUES FROM PARTIAL EXECUTION IN OFFLINE-TO-ONLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Action chunking, in which a policy generates sequences of actions rather than individual actions, can improve the sample efficiency of reinforcement learning through temporally coherent exploration and multi-step value learning. Adaptive action chunking selects how many actions to execute based on the current state, allowing the agent to replan using new observations. Though only part of a full chunk is executed, the policy must learn to generate a complete action chunk. Learning the value of that chunk from partial execution can provide feedback for improving the entire sequence. In this paper, we propose Partial-Execution Q-Chunking (PEQC), an offline-to-online reinforcement learning method that learns the values of complete action chunks from partial execution. PEQC retains generated chunks and constructs additional learning targets for the full-chunk value used in policy optimization by combining observed rewards with the estimated value of the original unexecuted actions. On OGBench, PEQC improves average evaluation success during online fine-tuning on several manipulation and navigation tasks relative to the same agent without this additional supervision. The observed gains suggest that full-chunk values can be effectively learned from partial execution while retaining the flexibility to replan before completing the generated action sequence.

1 INTRODUCTION

Offline-to-online reinforcement learning combines offline pretraining with online fine-tuning to learn from both existing data and new environment interactions (Levine et al., 2020; Nair et al., 2021; Ball et al., 2023). Action chunking provides a way to exploit the temporal structure of these data. An action chunking policy generates a sequence of H actions, and a chunk-conditioned critic estimates the return from executing that sequence. Q-chunking uses behavior regularization over action sequences to support temporally coherent exploration and trains the critic with multi-step temporal-difference targets (Li et al., 2025b).

Long action chunks also reduce the frequency of policy decisions. Under open-loop execution, the agent cannot adjust its actions to new observations until the chunk ends. Receding-horizon execution allows the agent to execute a shorter prefix and then replan (Nagy et al., 2026). Adaptive action chunking further selects the execution length according to the current state and estimated action values (Shin et al., 2026a; Chen et al., 2026; Gireesh et al., 2026). The policy can therefore generate H actions while executing only the first $k \leq H$ actions before sampling a new chunk.

Partial execution changes which action sequences are available for critic learning. When the policy is optimized using the full-chunk Q-value, its objective depends on all H generated actions. However, replanning replaces the unexecuted suffix with actions from a new chunk. A sequence sampled from replay may consequently contain actions from several chunks rather than the original policy output. Standard temporal-difference (TD) learning uses the observed rewards to update the Q-value of the executed sequence, but does not construct a target for the original full chunk from that partial execution. Figure 1 illustrates the problem setup.

In this work, we introduce Partial-Execution Q-Chunking (PEQC) to retain full-chunk policy optimization while learning from adaptive execution. We train a causal Transformer critic (Li et al.,

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.

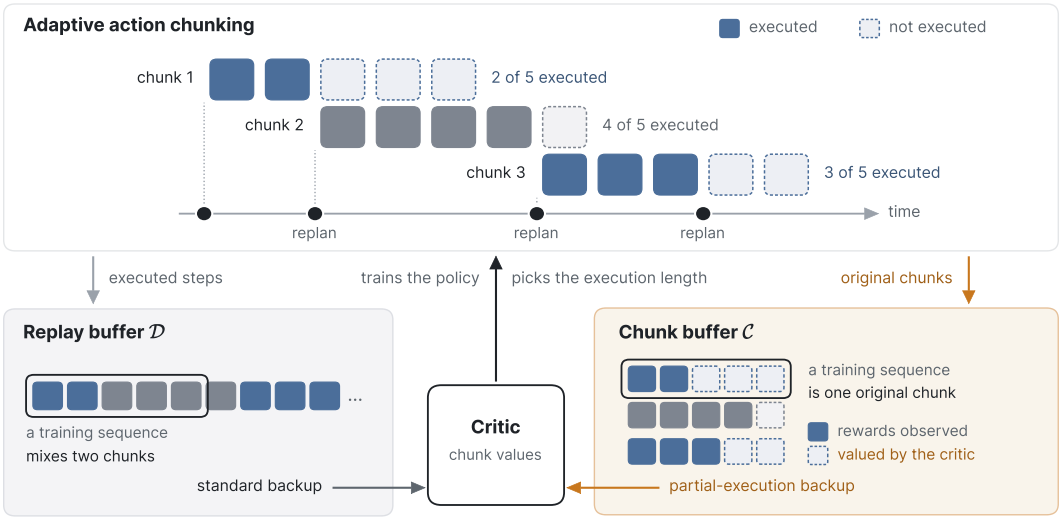


Figure 1: Overview of PEQC with $H = 5$. Filled boxes denote executed actions, and outlined boxes denote unexecuted actions. Blue and gray represent different action chunks. Replay sequences may cross chunk boundaries and are used to train all valid prefix values. During online fine-tuning, a separate chunk buffer $\mathcal{C}$ retains the full generated chunk, including its unexecuted suffix. For partial execution, the backup combines the observed prefix return with the discounted value of the original suffix, estimated by the target critic from the reached state. This provides an additional target for the original full-chunk value used in policy optimization while retaining the base prefix losses.

2025a; Shin et al., 2026a; Nagy et al., 2026) with multi-step TD targets for every prefix length $k = 1, \ldots, H$. The actor optimizes the full-chunk Q-value, which also provides the target for state-value learning. The prefix Q-values determine the execution length, while the full-chunk objective trains all H actions jointly. This allows the policy to optimize all H actions jointly while replanning after a shorter prefix.

We further complement this learning structure with a partial-execution backup that provides additional supervision for the original full-chunk value used for policy optimization. The agent stores the generated chunk together with the rewards and final state of its executed prefix. At that state, the target critic evaluates the original unexecuted suffix. The observed prefix return and the discounted suffix value form a TD target for the original full-chunk Q-value. Following the principle of learning from partial executions (Sutton et al., 1998), the backup adds a term to the critic loss without changing the actor objective or collecting additional environment transitions.

We evaluate PEQC on OGBench against the same agent trained without the partial-execution backup. We measure both success rate and normalized area under the success curve (AUC) over online training. The partial-execution backup improves AUC in several task families, including settings where final performance is similar. These results suggest that partial executions can be used to train full-chunk Q-values and improve online learning without requiring the agent to execute every chunk in full.

2 Preliminaries

Offline-to-online reinforcement learning. We consider a fully observed Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$ (Sutton & Barto, 1998). Here $\mathcal{S}$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ is the continuous action space, and d_a is the number of coordinates in one action. At environment step t, the agent observes $s_t \in \mathcal{S}$, applies $a_t \in \mathcal{A}$, receives a scalar reward r_t, and transitions to s_{t+1}. The transition distribution is $P(\cdot \mid s_t, a_t)$, r is the reward function, ρ_0 is the initial-state distribution, and $\gamma \in [0, 1)$ discounts future rewards. A policy specifies how actions are chosen. Reinforcement learning seeks a policy with high expected discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$. In offline-to-online learning, training begins with a previously collected dataset and continues with environment interaction. We use $\mathcal{D}$ for the replay buffer, initialized with the offline data and expanded with online transitions. The re-

play buffer preserves the temporal order of transitions so that consecutive trajectory segments can be sampled.

Action chunks and partial execution. An action chunk is a sequence generated from a single state (Zhao et al., 2023; Black et al., 2025; Liu et al., 2025). We denote its full length by H and write $\boldsymbol{a}_{t:t+H} = (a_t, \dots, a_{t+H-1}) \in \mathcal{A}^H$. The first k actions $\boldsymbol{a}_{t:t+k}$ form a *prefix* of this chunk. The remaining $H - k$ actions $\boldsymbol{a}_{t+k:t+H}$ form its *suffix*. With partial execution, the agent applies the first $k < H$ actions and generates a new chunk from s_{t+k} rather than continuing the original suffix. We call these two parts the *executed prefix* and *unexecuted suffix*. For example, when $H = 5$ and $k = 2$, the agent executes the first two actions and replans before executing the remaining three. The generated length H and the execution length k therefore need not be the same.

Values of action sequences. A value describes expected discounted reward. To specify what happens after a given action sequence, let $V : \mathcal{S} \to \mathbb{R}$ be a fixed, bounded *continuation value*, representing the future return from the state at which the sequence ends. For a sequence of k prescribed actions, define

$$Q^V(s_t, \boldsymbol{a}_{t:t+k}) = \mathbb{E}\left[R_t^k + \gamma^k V(s_{t+k}) \,\middle|\, s_t, \boldsymbol{a}_{t:t+k}\right], \qquad R_t^k = \sum_{j=0}^{k-1} \gamma^j r_{t+j}. \tag{1}$$

Here R_t^k is the discounted reward accumulated over those k steps. The expectation follows the environment transitions while the prescribed actions are executed without replanning. If V is the value of a continuation policy, Q^V is the return of executing the sequence and then following that policy. Temporal-difference (TD) learning fits a value prediction to observed rewards plus a value estimate at the resulting state. This use of a future value estimate is called *bootstrapping*. We use Q^V only for exact value identities, not for learned network predictions.

Prefix critics and chunking policies. A *critic* is a learned value predictor. We denote the prefix-value prediction of $Q^V(s_t, \boldsymbol{a}_{t:t+k})$ by $Q_\phi(s_t, \boldsymbol{a}_{t:t+k})$, where ϕ denotes the learnable parameters. The same critic also predicts the state value, denoted by $V_\phi(s_t)$. For $k = H$, QC-FQL, which applies Flow Q-Learning to the temporally extended action space (Li et al., 2025b; Park et al., 2025b), provides a way to learn $Q_\phi(s_t, \boldsymbol{a}_{t:t+H})$ for a chunking policy. In QC-FQL, the behavior flow policy $\mu_\theta(s, z)$ models the distribution of replay action sequences by transforming Gaussian noise into an action chunk of length H through iterative integration. A one-step policy $\mu_\omega(s, z)$, in contrast, generates an entire action chunk with a single network evaluation. Here, *one-step* refers to action generation rather than the execution of a single environment action. Both μ_θ and μ_ω take $z \sim \mathcal{N}(0, I_{Hd_a})$, where $z \in \mathbb{R}^{Hd_a}$ and I_{Hd_a} denotes the identity covariance matrix, and output H actions.

Given the execution policy μ_ω, the chunk critic $Q_\phi(s_t, \boldsymbol{a}_{t:t+H})$ is trained using the following objective:

$$\min_\phi \mathbb{E}\left[\left(Q_\phi(s_t, \boldsymbol{a}_{t:t+H}) - R_t^H - \gamma^H Q_{\bar{\phi}}(s_{t+H}, \mu_\omega(s_{t+H}, z))\right)^2\right], \tag{2}$$

where $\bar{\phi}$ denotes the parameters of a slowly updated target critic used to construct the learning target. Thus, $Q_\phi(s_t, \boldsymbol{a}_{t:t+H})$ estimates the discounted return obtained from state s_t when the specified action chunk is executed, followed by actions generated by μ_ω. The one-step policy μ_ω is optimized to maximize Q_ϕ while remaining close to the behavior flow policy μ_θ. Therefore, Q_ϕ conceptually approximates the optimal return under the behavior constraint, which discourages out-of-distribution action-chunk queries, in the spirit of behavior-constrained offline RL (Wu et al., 2019; Kostrikov et al., 2022).

We adopt the policy representation and full-chunk policy objective of QC-FQL. Our method additionally learns values for all prefixes, allowing the execution length to differ from the policy optimization horizon. We write $\boldsymbol{a}^\pi_{t:t+H} = \mu_\omega(s_t, z)$ for a generated chunk. The superscript π distinguishes it from actions sampled from replay. We suppress action clipping and ensemble indices in the main-text equations. Their implementation and episode-boundary handling are specified in Appendix A.

3 PARTIAL-EXECUTION Q-CHUNKING

PEQC learns values for all action prefixes while optimizing its policy with the value of the full chunk. Execution can stop after a shorter prefix. We first describe this learning objective and its relationship to adaptive execution. We then introduce an additional target that uses partial executions to train the value of the original full chunk.

3.1 PREFIX VALUE LEARNING AND FULL-CHUNK POLICY OPTIMIZATION

To efficiently obtain both the state-value prediction $V_\phi(s_t)$ and all prefix-value predictions $\{Q_\phi(s_t, \boldsymbol{a}_{t:t+k})\}_{k=1}^H$ in a single forward pass, we employ a causal Transformer with shared parameters ϕ (Li et al., 2025a; Shin et al., 2026a). Its causal attention mask prevents the prediction for a prefix of length k from attending to actions occurring at or after $t+k$, thereby ensuring that $Q_\phi(s_t, \boldsymbol{a}_{t:t+k})$ depends only on the corresponding action prefix. The critic learns from executed sequences in $\mathcal{D}$. For each valid prefix of length k, the target combines its observed return with the target state value at s_{t+k},

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{H}\sum_{k=1}^{H}\left(Q_\phi(s_t, \boldsymbol{a}_{t:t+k}) - R_t^k - \gamma^k V_{\bar{\phi}}(s_{t+k})\right)^2\right]. \tag{3}$$

Here $\mathbb{E}_{\mathcal{D}}$ averages over consecutive sequences sampled from replay. Each target uses the actions that were actually executed over its reward interval. We reuse each sampled segment to construct targets for every valid prefix length up to H.

Policy optimization instead evaluates the generated chunk as a whole. We fit the state value to the target critic's prediction $Q_{\bar{\phi}}(s_t, \mu_\omega(s_t, z))$ for a full chunk from the current one-step policy,

$$\mathcal{L}_V(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},\, z\sim\mathcal{N}(0, I_{Hd_a})}\left[\left(V_\phi(s_t) - Q_{\bar{\phi}}(s_t, \mu_\omega(s_t, z))\right)^2\right]. \tag{4}$$

Thus, V_ϕ supplies a full-chunk continuation estimate for value learning. We sample a starting timestep uniformly from replay and use its state as s_t in Equation 4. The state-value predictor is not explicitly trained to evaluate the policy with adaptive execution.

The one-step policy $\mu_\omega(s, z)$ is trained to increase the full-chunk value while remaining close to the behavior flow policy $\mu_\theta(s, z)$,

$$\mathcal{L}_\pi(\omega) = \mathbb{E}_{s_t\sim\mathcal{D},\, z\sim\mathcal{N}(0, I_{Hd_a})}\left[-Q_\phi(s_t, \mu_\omega(s_t, z))\right] + \alpha\mathcal{L}_{\text{Distill}}(\omega). \tag{5}$$

The coefficient $\alpha > 0$ controls the strength of the behavior constraint. The distillation loss $\mathcal{L}_{\text{Distill}}$ is the mean squared difference between the one-step output and the behavior-policy target for the same state and noise, averaged over all action coordinates. The behavior flow policy is trained with flow matching on replay sequences throughout both training phases (Park et al., 2025b; Li et al., 2025b). The clipping and gradient conventions for these losses are given in Appendix A.

The value-learning objective in Equation 3 consists of two components. First, we optimize the value prediction for the full chunk, $\mathbb{E}_{\mathcal{D}}\left[\frac{1}{H}\left(Q_\phi(s_t, \boldsymbol{a}_{t:t+H}) - R_t^H - \gamma^H V_{\bar{\phi}}(s_{t+H})\right)^2\right]$, together with the objectives in Equation 4 and Equation 5. We additionally train the remaining prefix-value predictions using the corresponding k-step targets: $\mathbb{E}_{\mathcal{D}}\left[\frac{1}{H}\sum_{k=1}^{H-1}\left(Q_\phi(s_t, \boldsymbol{a}_{t:t+k}) - R_t^k - \gamma^k V_{\bar{\phi}}(s_{t+k})\right)^2\right]$. Because all prefix values share the same Transformer parameters, the model jointly learns value representations across different chunk lengths while producing all prefix predictions in a single forward pass.

This separates the horizon of policy optimization from the number of actions executed. Unlike averaging prefix values under its execution-length distribution in the policy objective and state-value target (Shin et al., 2026a), we use the full-chunk value in both the policy objective and state-value target, while Equation 3 continues to train all valid prefixes. The H-step target connects the value at s_{t+H} to s_t through H observed rewards, following the multi-step learning principle of Q-chunking (Li et al., 2025b). At the same time, the policy objective assesses all H generated actions jointly. This does not imply that increasing H always improves learning. A longer chunk optimizes more actions chosen before intermediate states are observed. We treat H as a task-dependent hyperparameter. We analyze the effects of our policy objective and state-value target in Section 4.4.

3.2 ADAPTIVE EXECUTION

The prefix values also determine how many actions are applied before replanning. We use softmax length sampling during data collection and greedy length selection during evaluation (Shin et al., 2026a). Softmax sampling favors higher-valued prefixes while allowing other lengths to be tried. Greedy selection chooses the prefix with the highest predicted value. Neither rule changes the full-chunk learning objectives above. The exact selection probabilities are specified in Appendix A.

3.3 PARTIAL-EXECUTION BACKUP

The policy objective in Equation 5 evaluates the complete generated chunk even when only its first $k < H$ actions are executed. After replanning, however, replay contains these actions followed by actions from later chunks. The base loss already trains the executed prefix and the resulting replay sequence. It does not construct a full-chunk target for the original sequence from that partial execution.

Using the exact value definition in Section 2, fix the continuation value V, the generated chunk, and a split $k < H$ before executing the chunk. Splitting the discounted return gives

$$\begin{aligned} &Q^V(s_t, \boldsymbol{a}^\pi_{t:t+H}) - Q^V(s_t, \boldsymbol{a}^\pi_{t:t+k}) \\ &\qquad = \gamma^k \mathbb{E}\left[Q^V(s_{t+k}, \boldsymbol{a}^\pi_{t+k:t+H}) - V(s_{t+k})\right]. \end{aligned} \tag{6}$$

The expectation is over transitions while executing the prescribed prefix. Completing the original suffix and switching to the continuation represented by V can have different values. Consequently, a target for the prefix alone does not determine a target for the full chunk. Appendix B derives this identity and states its assumptions.

To address this, we maintain a buffer $\mathcal{C}$ of completed chunk records. Each record stores the starting state s_t, the generated full chunk $\boldsymbol{a}^\pi_{t:t+H}$, the actual execution length k, the rewards observed during those k steps, and the resulting state s_{t+k}. When the agent replans after $k < H$ actions, we construct the target

$$y_t = R^k_t + \gamma^k Q_{\bar{\phi}}(s_{t+k}, \boldsymbol{a}^\pi_{t+k:t+H}). \tag{7}$$

The target critic evaluates the original remaining $H - k$ actions from the observed state s_{t+k}. It does not generate a new suffix at that state. When $k = H$, the target is $R^H_t + \gamma^H V_{\bar{\phi}}(s_{t+H})$, as in the full-length term of Equation 3. Episode-boundary handling is specified in Appendix A.

The complete critic objective is

$$\begin{aligned} \mathcal{L}(\phi) = \mathcal{L}_Q(\phi) + \mathcal{L}_V(\phi) \\ + \lambda\, \mathbb{E}_{\mathcal{C}}\left[\left(Q_\phi(s_t, \boldsymbol{a}^\pi_{t:t+H}) - y_t\right)^2\right], \end{aligned} \tag{8}$$

where $\mathbb{E}_{\mathcal{C}}$ averages over sampled chunk records and $\lambda \geq 0$ controls the auxiliary loss. We set $\lambda = 1/H$, which is the nominal coefficient of one prefix term in Equation 3. This target adds supervision to the same full-chunk value used for policy improvement. It neither replaces the base prefix losses nor changes the policy or state-value objectives. Its suffix term is a learned value estimate, not an observed return.

3.4 PRACTICAL DETAILS

The auxiliary loss is used only during online fine-tuning. Executed transitions are added to $\mathcal{D}$, while a record is added to $\mathcal{C}$ when execution of a generated chunk ends. Each update uses a replay batch and a separately sampled batch from $\mathcal{C}$. The critic loss in Equation 8 is optimized together with the policy and flow-matching losses in a single optimizer step. The auxiliary loss adds critic computation but requires no additional policy samples or environment interactions. Sampling and optimization settings are given in Appendix D.

4 EXPERIMENTS

We evaluate PEQC on the state-based, single-task benchmarks of OGBench (Park et al., 2025a). Training consists of one million offline updates followed by one million environment steps of online

fine-tuning. We report final success and normalized area under the online success curve (AUC), averaged over three training seeds. Experimental settings are provided in Appendix D.

4.1 Effect of the partial-execution backup

Figure 2 compares the same learner with and without the additional full-chunk supervision from the partial-execution (PE) backup described in Section 3.3.

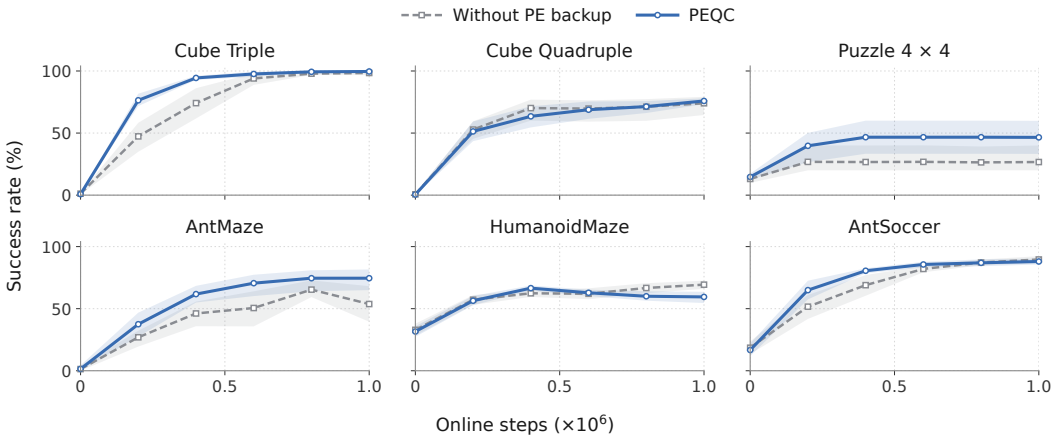


Figure 2: Online learning with and without the partial-execution backup. Lines show mean success across five tasks and three training seeds. Shading shows pointwise 95% bootstrap confidence intervals, obtained by resampling training runs within each task. Collection uses softmax length selection and evaluation uses greedy length selection.

The backup yields higher mean online AUC on Cube Triple, AntSoccer, AntMaze, and Puzzle 4×4. On Cube Triple and AntSoccer, PEQC reaches high success earlier, while the agent without the backup largely catches up by the end of training. Final scores alone therefore miss much of the gain. On AntMaze and Puzzle 4×4, the advantage persists through the final evaluation. For Puzzle, this difference mainly reflects additional successful runs on Tasks 3 and 4 rather than small improvements across all tasks.

Cube Quadruple shows no clear learning-speed advantage despite a slightly higher final score. On HumanoidMaze, the agents perform similarly early in training, but PEQC loses performance while the agent without the backup continues to improve. This late decline is more apparent in final success than in AUC, highlighting the need to consider both measures. Aggregate scores are reported in Table 4. (For additional analysis comparing our method with a variant of the partial-execution backup, see Appendix G.)

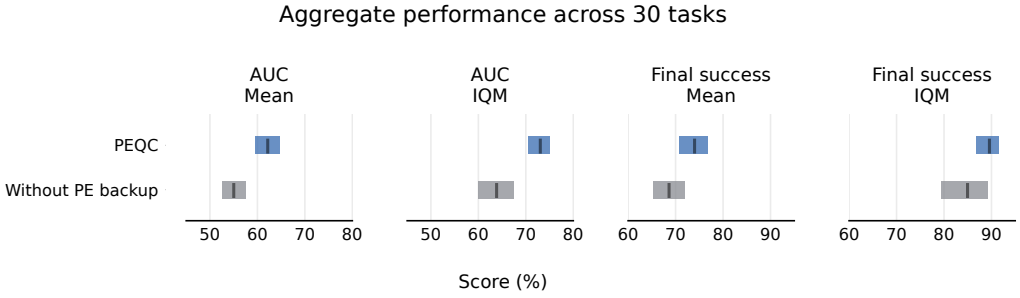


Figure 3: Aggregate online AUC and final success across 30 tasks from six task families, with three training seeds per task. Black markers show the mean and interquartile mean (IQM). Colored bars show 95% bootstrap confidence intervals from resampling paired training runs within each task.

Table 1: Five-task mean final success (%). PE denotes partial execution. Published results use their source protocols, detailed in Appendix F. † uses 10M Cube Quadruple transitions. Bold and underlined values indicate the highest and second highest means in each column, respectively.

(a) Cube manipulation

Method	Cube Triple	Cube Quadruple
RLPD	41	0
FQL	18	3
QC-FQL	53	<u>77</u>
QC	64	73
ACH	67	52†
AQC	90	**88**
ACSAC	97	73
PEQC w/o PE backup	<u>98.4</u>	74.0
PEQC	**99.6**	75.9

(b) Maze, soccer, and puzzle tasks

Method	AntMaze	Humanoid Maze	AntSoccer	Puzzle 4×4
RLPD	69	29	64	<u>64</u>
BFN	22	<u>87</u>	74	60
FQL	74	56	70	54
FINO	**76**	**92**	71	53
QC ($H = 5$)	47	32	67	38
ACH	72	82	84	**90**
PEQC w/o PE backup	53.7	69.3	**89.6**	26.7
PEQC	<u>74.5</u>	59.5	<u>88.0</u>	46.5

We aggregate results across six task families, each containing five tasks. PEQC achieves higher mean and interquartile mean (IQM) scores for both online AUC and final success (Figure 3).

4.2 Benchmark performance

Table 1 places final performance alongside published results (Gireesh et al., 2026; Shin et al., 2026a; Chen et al., 2026). These results follow their source protocols rather than our paired experimental setting.

RLPD combines offline and online replay (Ball et al., 2023). BFN selects the highest-valued candidate from a behavior flow policy (Li et al., 2025b), while FQL and FINO use flow-based policy learning (Park et al., 2025b; Shin et al., 2026b). Chunking baselines include fixed-length QC and QC-FQL (Li et al., 2025b), and adaptive-length ACH, AQC, and ACSAC (Shin et al., 2026a; Gireesh et al., 2026; Chen et al., 2026). Section 5 discusses these methods in more detail.

The agent without the backup already achieves near full success on Cube Triple, while the backup improves earlier learning. On AntMaze, PEQC is close to the reported FQL, FINO, and ACH means. On AntSoccer it exceeds the reported ACH mean, whereas on HumanoidMaze it remains below ACH and FINO. These whole-agent comparisons complement rather than replace the paired learning curves in Figure 2.

4.3 Execution length selection

We evaluate the same final PEQC checkpoints under three execution rules while keeping the policy and critic parameters fixed. Adaptive execution selects the prefix with the highest predicted value. Uniform execution samples a length uniformly from $1, \ldots, H$ at each decision, while fixed execution uses length H. Each of the 90 checkpoints is evaluated for 50 episodes per rule. This comparison changes the evaluation rule, not the training collector.

Figure 4 shows that on AntMaze and HumanoidMaze, adaptive execution exceeds fixed execution by 20.4 and 32.3 percentage points, respectively. The corresponding gains over uniform execution are 13.1 and 23.2 points. On AntSoccer, adaptive and uniform execution have similar mean success, while adaptive execution exceeds fixed execution by 4.1 points. The differences are below one point on both cube benchmarks, and all three rules achieve the same mean success on Puzzle. These results support value-based execution length selection on the two maze benchmarks, but do not show a consistent advantage across all task families. Collection length statistics are reported in Appendix H.

4.4 Policy and state-value objectives

We examine the full-chunk objectives separately from the backup in an exploratory comparison on Cube Triple Task 5. Every configuration retains all-prefix TD learning and adaptive execution, with

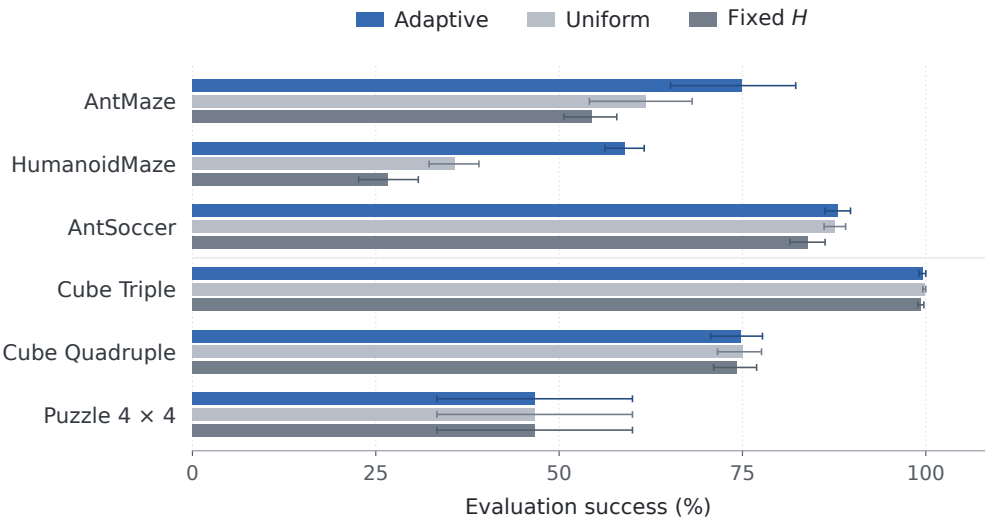


Figure 4: Success of the same final PEQC checkpoints under adaptive, uniform, and fixed-H execution. Bars show mean success across five tasks and three training seeds per task. Error bars show 95% bootstrap confidence intervals, obtained by resampling checkpoints within each task while keeping the three execution rules paired. Each checkpoint is evaluated for 50 episodes per rule.

no partial-execution backup. We replace the full-chunk value in either the actor objective or the state-value target with a softmax-weighted average of prefix values. Changing the actor objective changes which sequence values are used to guide policy improvement. Changing the state-value target instead changes the continuation estimate used in every prefix TD target in Equation 3, while the actor still optimizes full chunks. Each configuration is trained from scratch through both the offline and online phases, and neither comparison removes short-prefix supervision nor enforces a fixed execution length.

Table 2: Exploratory comparison of policy and state-value objectives on Cube Triple Task 5. Mean $\pm$ sample standard deviation across three training seeds. All configurations train every prefix value and omit the partial-execution backup. Bold and underlined values indicate the highest and second highest means in each column, respectively.

Actor Q objective	State-value target	Online AUC (%)	Final success (%)
Full chunk	Full chunk	$\mathbf{79.0} \pm 8.4$	$\mathbf{99.3} \pm 1.2$
Weighted prefixes	Full chunk	22.0 ± 23.5	49.3 ± 45.2
Full chunk	Weighted prefixes	$\underline{65.1} \pm 7.7$	$\underline{95.3} \pm 1.2$

Using the full-chunk value in both objectives gives the highest mean AUC (Table 2). This supports the design choice in Section 3.1 for this setting, but does not establish that either objective alone explains performance differences relative to published methods. Appendix D provides the objective definitions and experimental settings. Collection-rule sensitivity, including the complete greedy-collection results, is reported in Appendix I.

5 Related Work

Action chunking and policy learning. Action chunking is widely used in imitation learning to model sequences of actions from demonstrations. Action Chunking with Transformers (ACT) learns such sequences for bimanual manipulation (Zhao et al., 2023), while Diffusion Policy generates them with a diffusion model and uses receding-horizon execution (Chi et al., 2025). In reinforcement learning, Q-chunking treats an action sequence as a temporally extended action and trains the critic with the corresponding multi-step return (Li et al., 2025b). Conditioning on the actions that

produced the rewards enables multi-step value learning in the chunked action space, while a behavior constraint on whole sequences supports temporally coherent exploration. Reinforcement Learning with Prior Data (RLPD) learns online from both offline data and newly collected transitions (Ball et al., 2023). Best-of-N (BFN) selects the highest-valued action among candidates sampled from a behavior flow policy (Ghasemipour et al., 2021; Li et al., 2025b). Flow Q-Learning (FQL) learns a one-step policy through value maximization and distillation from a behavior flow policy (Park et al., 2025b). Flow Matching with Injected Noise for Offline-to-Online Reinforcement Learning (FINO) combines noise injection during flow-policy training with entropy-guided action sampling (Shin et al., 2026b). QC-FQL applies FQL's policy learning approach to action chunks (Li et al., 2025b). We use this policy parameterization in the offline-to-online setting.

Prefix value learning and policy optimization. A prefix critic estimates the values of the first k actions of a chunk for several lengths $k \leq H$, where H is the full chunk length. Transformer-based Off-Policy Episodic Reinforcement Learning (TOP-ERL) learns action-sequence values with a causal Transformer and multi-step targets (Li et al., 2025a). Adaptive Action Chunking (ACH) uses a shared critic to estimate prefix values and a state value (Shin et al., 2026a). Sample Efficient Action Chunking Reinforcement Learning (SEAR) also trains on multiple prefix targets while using the full-chunk value in its entropy-regularized policy objective (Nagy et al., 2026). Learning multiple prefix values does not require averaging them in the policy objective. In ACH, the policy objective and state-value target average prefix values under the execution-length distribution. PEQC instead uses the full-chunk value in both, while retaining TD learning for every valid replay prefix.

Adaptive execution length. The execution rule determines how many generated actions are applied before the agent replans from a new observation. ACH samples this length from a softmax over prefix values during collection and selects the highest-valued prefix during evaluation (Shin et al., 2026a). Adaptive Chunk Size Actor-Critic (ACSAC) searches over both candidate chunks and their prefix lengths using a causal Transformer critic (Chen et al., 2026). Adaptive Q-Chunking (AQC) learns critics at multiple scales and selects chunks using discount-normalized advantages (Gireesh et al., 2026). SEAR instead uses random prefix execution during collection to support receding-horizon evaluation (Nagy et al., 2026).

Learning from partial execution. The options framework represents temporally extended actions through policies and termination conditions (Sutton et al., 1999; Bacon et al., 2017). Intra-option learning estimates their values from compatible fragments of experience without requiring complete executions (Sutton et al., 1998). Harutyunyan et al. (2018) separate the termination rule used for data collection from the rule defining the option values being learned. For action repetition, Sabbioni et al. (2023) introduce an all-persistence Bellman operator that reuses experience across execution durations. When the observed execution is shorter than the duration being evaluated, their target bootstraps the value of the remaining repetitions. We study a related update for policy-generated sequences of continuous actions rather than repetitions of a single action. Our critic conditions on the generated sequence. After partial execution, PEQC evaluates the original unexecuted suffix at the resulting state and combines its estimated value with the observed return to provide additional supervision for the full chunk.

6 Conclusion

We presented PEQC, an offline-to-online reinforcement learning method that combines prefix value learning, full-chunk policy optimization, and adaptive execution. The partial-execution backup extends this base learner by using observed rewards and the estimated value of the original remaining actions to provide additional supervision for the full-chunk value used in policy optimization. Experiments on OGBench show higher mean online AUC in several task families, compared to the same learner without this additional supervision. The observed gains support learning full-chunk values from partial execution while retaining the flexibility to replan before completing a generated action sequence. Limitations and future work are discussed in Appendix C.

AI USE STATEMENT

We used large language models to assist with implementation, experiments, data analysis, research discussions, and manuscript preparation. The authors reviewed the resulting materials, made the final research decisions, and take responsibility for the content.

ETHICS STATEMENT

This work uses simulated benchmark environments and raises no ethical concerns beyond those common to reinforcement learning research.

REPRODUCIBILITY STATEMENT

The learning objectives and partial-execution backup are described in Section 3, with implementation details and episode-boundary handling in Appendix A. Appendix B states the assumptions and derives Equation 6. The datasets, architectures, hyperparameters, and training and evaluation settings are provided in Appendix D. Aggregation procedures and statistical comparisons for the main results are detailed in Appendix E. Protocols for the additional analyses are described in Appendices G–J.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *ICML*, 2023.

Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *NeurIPS*, 2025.

Qian Chen, Junqiao Zhao, Hongtu Zhou, Hang Yu, Yanping Zhao, Chen Ye, and Guang Chen. ACSAC: Adaptive chunk size actor-critic with causal transformer q-network. *arXiv:2605.11009*, 2026.

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

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 *ICML*, 2021.

Nandiraju Gireesh, Yuanliang Ju, and He Wang. Adaptive Q-Chunking for offline-to-online reinforcement learning. *arXiv preprint arXiv:2605.05544*, 2026.

Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowe. Learning with options that terminate off-policy. In *AAAI*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with Implicit Q-Learning. In *ICLR*, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *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 *ICLR*, 2025a.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *NeurIPS*, 2025b.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled Q-chunking. In *ICLR*, 2026.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *ICLR*, 2025.

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:2603.01891*, 2026.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets, 2021. URL https://arxiv.org/abs/2006.09359.

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

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-Learning. In *ICML*, 2025b.

Luca Sabbioni, Luca Al Daire, Lorenzo Bisi, Alberto Maria Metelli, and Marcello Restelli. Simultaneously updating all persistence values in reinforcement learning. In *AAAI*, 2023.

Yongjae Shin, Jongseong Chae, Seongmin Kim, Jongeui Park, and Youngchul Sung. Adaptive action chunking via multi-chunk q value estimation. *arXiv:2605.10044*, 2026a.

Yongjae Shin, Jongseong Chae, Jongeui Park, and Youngchul Sung. Flow matching with injected noise for offline-to-online reinforcement learning. In *ICLR*, 2026b.

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

Richard S Sutton, Doina Precup, and Satinder Singh. Intra-Option Learning about Temporally Abstract Actions. In *ICML*, 1998.

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.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv:1911.11361*, 2019.

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

A CRITIC AND POLICY IMPLEMENTATION DETAILS

Critic inputs and outputs. Each critic processes a state token followed by H action tokens, using separate state and action projections, positional embeddings, causal Transformer layers, and a final layer normalization. A shared scalar output layer gives $V_\phi(s_t)$ at position 0 and $Q_\phi(s_t, \boldsymbol{a}_{t:t+j})$ at position $j = 1, \dots, H$. Each token attends only to itself and preceding tokens. Thus, V_ϕ does not depend on actions, and a prefix value does not depend on later actions. For a batch of N sequences, the action input has shape (N, H, d_a). The two-critic outputs have shapes $(2, N, H)$ for prefix values and $(2, N)$ for state values.

Critic ensemble and target parameters. The two critics have independent parameters. Within each critic, state and prefix values share the Transformer and output layer. Regression targets average the target critics' predictions. Both current critics fit this target, and their individual squared residuals are averaged rather than squaring the residual of their mean. The policy objective and execution rule average the current critics' predictions. Selection scores and bootstrap values below use this mean convention with ensemble indices omitted.

With n indexing optimizer updates, target parameters follow

$$\bar{\phi}_{n+1} = (1 - \tau)\bar{\phi}_n + \tau\phi_n, \qquad \tau = 0.005,$$

where ϕ_n is the current critic before that optimizer update. This Polyak average covers the complete critic ensemble.

Execution-length selection. For a generated chunk $\boldsymbol{a}^\pi_{t:t+H}$, the probability of selecting j actions during data collection is

$$\pi_L(j \mid s_t, \boldsymbol{a}^\pi_{t:t+H}) = \frac{\exp\big(\beta Q_\phi(s_t, \boldsymbol{a}^\pi_{t:t+j})\big)}{\sum_{i=1}^{H} \exp\big(\beta Q_\phi(s_t, \boldsymbol{a}^\pi_{t:t+i})\big)}, \qquad j \in \{1, \dots, H\}. \tag{9}$$

Here π_L is computed from the critic rather than learned by a separate network. The inverse temperature is $\beta = 1$. Greedy selection chooses

$$\arg\max_{j \in \{1,\dots,H\}} Q_\phi(s_t, \boldsymbol{a}^\pi_{t:t+j}),$$

with ties resolved in favor of the shortest length. Softmax collection and greedy evaluation follow Shin et al. (2026a). Policy noise is resampled for every chunk under both rules. Greedy collection uses the evaluation rule. The episode mixture chooses greedy or softmax with equal probability at each reset and retains that rule, but not the selected length or policy noise, until the episode ends. The stored k counts actions actually executed and can be smaller than the selected length at an episode boundary.

Evaluating the unexecuted suffix. For $k < H$, the target critic receives s_{t+k} and the original remaining actions $\boldsymbol{a}^\pi_{t+k:t+H}$. The actions are left-aligned in their original order, followed by k zero action vectors, retaining input shape (N, H, d_a). The reached state occupies token 0, and the suffix uses positions $1, \dots, H - k$ rather than its original positions $k + 1, \dots, H$. We read action-token position $H - k$, or index $H - k - 1$ in the action-only prefix array. Causal attention prevents trailing padding from affecting this output. For $k = H$, the bootstrap uses the state-value output instead. The auxiliary prediction at s_t uses the original full chunk without shifting.

Trajectory boundaries and auxiliary targets. Stored bootstrap masks are applied during reward accumulation and endpoint bootstrapping. In online interaction, true termination sets the bootstrap mask to zero, while a time-limit truncation retains bootstrapping from the final observation before reset. Neither case permits a replay sequence to cross into the next episode. Invalid prefix losses are set to zero before averaging over the batch, critics, and all H prefix positions. The normalization is not changed to the number of valid prefixes.

For a completed chunk record, these conventions give

$$y_t = \begin{cases} R^k_t, & \text{if execution ends at true termination,} \\ R^k_t + \gamma^k Q_{\bar{\phi}}(s_{t+k}, \boldsymbol{a}^\pi_{t+k:t+H}), & \text{otherwise, if } k < H, \\ R^H_t + \gamma^H V_{\bar{\phi}}(s_{t+H}), & \text{otherwise, if } k = H. \end{cases} \tag{10}$$

Here R_t^k contains only observed rewards. At a time-limit truncation, the second or third case uses the last observation before reset, not the next episode's initial observation. For $k < H$, the original suffix is retained. Stored actions and the complete target are fixed during auxiliary regression.

Distillation and action clipping. The behavior flow policy generates a chunk by numerical integration. Its clipped output is the distillation target, while the one-step policy output remains unclipped in this loss,

$$\mathcal{L}_{\text{Distill}}(\omega) = \mathbb{E}_{s\sim\mathcal{D},\, z\sim\mathcal{N}(0,I_{Hd_a})}\left[\frac{1}{Hd_a}\left\|\mu_\omega(s,z) - \text{SG}[\text{clip}(\mu_\theta(s,z),-1,1)]\right\|_2^2\right], \tag{11}$$

where SG denotes stop-gradient. Both policy outputs use the same Gaussian noise. Actions passed to the critic or environment are clipped to $[-1, 1]$. The policy gradient passes through clipping and the critic's action input, without updating critic parameters. Value-regression targets are detached. The behavior policy is updated only through flow matching, not through distillation or value maximization.

B Full-chunk and prefix values

This section derives Equation 6. As in Section 2, fix a bounded continuation function V in a Markov decision process. For an action sequence of length j, define

$$Q^V(s_t, \boldsymbol{a}_{t:t+j}) = \mathbb{E}\left[\sum_{u=0}^{j-1}\gamma^u r_{t+u} + \gamma^j V(s_{t+j}) \,\middle|\, s_t, \boldsymbol{a}_{t:t+j}\right]. \tag{12}$$

The expectation uses the environment transition law while the specified actions are executed without replanning. We first consider a continuing process. The function Q^V in this derivation is an exact action-sequence value, not the learned critic Q_ϕ.

Condition on s_t, a full chunk $\boldsymbol{a}^\pi_{t:t+H}$, and a split $k < H$ chosen before execution. The chunk may have been sampled from a stochastic policy, and k may depend on the starting state and the chunk. Neither depends on future environment transition noise. Since the full chunk and its prefix prescribe the same first k actions, they induce the same distribution of R_t^k and s_{t+k}. Splitting the return and applying the Markov property gives

$$\begin{aligned} Q^V(s_t, \boldsymbol{a}^\pi_{t:t+H}) &= \mathbb{E}\left[R_t^k + \gamma^k Q^V(s_{t+k}, \boldsymbol{a}^\pi_{t+k:t+H})\right], \\ Q^V(s_t, \boldsymbol{a}^\pi_{t:t+k}) &= \mathbb{E}\left[R_t^k + \gamma^k V(s_{t+k})\right]. \end{aligned} \tag{13}$$

Both expectations condition on the same starting state, chunk, and split. Subtracting the second equality from the first proves Equation 6.

Two chunks with the same starting state and prefix have the same prefix value, but their full-chunk values can differ through their suffixes. Equation 6 compares completing the original suffix with the continuation represented by V. It is not a prefix estimation error. A correct prefix target is therefore not necessarily a target for the original full chunk.

Prescribed actions and replay segments. The expectation above executes an action sequence fixed at the starting state. It does not require deterministic transitions. A replay segment that crosses a replanning point is different, since later actions may have been selected using intermediate observations. In a stochastic environment, conditioning on those later actions can change the conditional distribution of earlier states. The decomposition therefore does not assert that arbitrary replay segments give unbiased samples of a prescribed open-loop sequence value. The original chunk in the partial-execution target is fixed before execution, but its learned suffix estimate can still be inaccurate. We do not interpret the measured gains as evidence that stochastic replay bias was corrected.

Episode boundaries. True terminal states are treated as absorbing, with zero subsequent rewards and $V = 0$, so the decomposition has zero continuation after termination. A time-limit truncation does not terminate the underlying task when bootstrapping is retained. Equation 10 specifies these cases using the observation before reset, without transitions from the next episode. The identity concerns the task dynamics, not replay trajectories conditioned on surviving to a given length.

Scope of the learned backup. Equation 7 replaces the exact suffix value with a learned target-critic estimate. It supervises the original full chunk without observing rewards along the unexecuted suffix. The identity specifies the value being estimated, but does not guarantee improved value accuracy or policy performance under function approximation.

C LIMITATIONS AND FUTURE WORK

Evaluation scope. Our main benchmark evaluation focuses on state-based, single-task offline-to-online learning on OGBench, with three training seeds per task. Extending the evaluation to visual observations and multi-task learning would help assess the applicability of partial-execution supervision beyond the current setting. Additional training seeds would also help characterize variability in learning outcomes.

Learned suffix values. The backup relies on a learned critic to evaluate the unexecuted suffix, so its target is subject to value estimation error. The exploratory diagnostics in Appendix G leave open whether the observed learning gains arise from more accurate value predictions. Controlled comparisons using common states, action chunks, and continuation policies could help clarify how the additional full-chunk supervision affects value learning and policy improvement.

Task-dependent benefits and training choices. The benefit of the partial-execution backup varies across environments, and gains during fine-tuning do not always translate into higher final success. Further work should investigate the late performance decline on HumanoidMaze (Appendix E). Our experiments keep the generated chunk length H fixed within each task family and use an auxiliary coefficient $\lambda = 1/H$. A broader study of H and λ, including auxiliary weights that vary during online training, could help identify when and how strongly to apply the additional supervision.

Computational cost. The auxiliary update reuses completed chunk records without additional environment interactions, but requires extra critic computation. Consequently, gains in environment-step efficiency do not necessarily translate into shorter wall-clock training time. Appendix J reports observed workload comparisons rather than controlled equal-compute experiments. More efficient suffix evaluation and less frequent auxiliary updates are directions for studying the trade-off between environment-step efficiency and computational cost.

D TRAINING AND EVALUATION DETAILS

Online sampling and updates. The replay buffer $\mathcal{D}$ is initialized with offline data. The auxiliary buffer $\mathcal{C}$ is empty at the start of online fine-tuning and receives a completed record when execution of a chunk ends. We use a warm-up of 5,000 environment steps, included in the one-million-step online budget. Optimization begins at step 5,000, with one joint optimizer update per subsequent primitive environment step. This schedule gives 995,001 online updates in a completed run. Each update samples 256 executed sequences from $\mathcal{D}$ and an independent batch of 256 completed records uniformly from $\mathcal{C}$. Replay sequences are sampled uniformly by their primitive start positions. The auxiliary batch is used only when the corresponding auxiliary loss is enabled. Its coefficient is $\lambda = 1/H$. No second optimizer step is added for the auxiliary loss.

Datasets and task-specific settings. Table 3 gives the settings for the main backup comparison. We use the task-family settings reported by ACH for the behavior regularization coefficient α and discount factor γ (Shin et al., 2026a). The maze and soccer experiments use the `navigate` datasets, while the cube and puzzle experiments use the `play` datasets. All six task families use their native single-task rewards without an additional binary reward conversion.

During offline training on Cube Quadruple, the resident training shard is replaced every 1,000 updates. Online replay is initialized from the resident shard and receives newly collected transitions. The corpus size is therefore distinct from the amount of offline data held in online replay. Puzzle 4×4 uses the `puzzle-4x4-play-v0` dataset with $H = 5$ in the main comparison. Earlier $H = 10$ runs are not included in these results. The values of H and α are fixed within each task family across tasks, seeds, and backup conditions. This protocol does not constitute a controlled sweep over either hyperparameter.

Table 3: Settings for the main backup comparison. Dataset sizes are nominal training sizes in millions of transitions. Each family includes five tasks and three training seeds. † denotes the Cube Quadruple corpus of 100 one-million-transition shards, with one shard resident at a time. Episode limits are measured in primitive environment steps.

Environment	Dataset size	H	α	γ	Episode limit
Cube Triple	3	10	300	0.99	1,000
Cube Quadruple	$100^{\dagger}$	10	300	0.99	1,000
Puzzle 4×4	1	5	300	0.99	500
AntMaze Giant	1	5	30	0.995	1,000
HumanoidMaze Medium	2	5	100	0.995	2,000
AntSoccer Arena	1	5	30	0.995	1,000

Architecture and optimization. The critic architecture is adapted from TOP-ERL (Li et al., 2025a). We use two Transformer blocks, four attention heads, and a model width of 128, following ACH (Shin et al., 2026a). We set the feed-forward width to 512 and use no dropout. We retain the policy architecture and base optimization settings of QC-FQL (Li et al., 2025b). The one-step policy uses four hidden layers of width 512. The behavior flow policy uses ten integration steps. We use Adam (Kingma & Ba, 2015) with learning rate 3×10^{-4} and a target-network averaging rate of 0.005. In the AntMaze, HumanoidMaze, and AntSoccer experiments, the Adam update proposed for the one-step policy is multiplied by $1/3$ during online fine-tuning. The same scaling is applied to both methods in each backup comparison. It does not change updates for the behavior policy, critic, or state value. The cube and puzzle experiments do not apply this scaling. Ensemble aggregation, action clipping, and gradient paths are described in Appendix A.

Experimental settings and initialization. The Cube Triple learning curves retain the earlier experiment, with a shared offline checkpoint within each task and seed and a fresh online start using reseeded random-number generators. The completed five-task auxiliary comparison uses a separate Cube Triple experiment, as specified in Appendix G. Its values are not substituted for the historical learning curves.

The Cube Quadruple and Puzzle results use the completed $H = 10$ and $H = 5$ experiments, respectively. Within each task and seed, the recorded offline checkpoint, initial model, replay, and initial observation are the same for the two methods. The later AntMaze, HumanoidMaze, and AntSoccer pairs also share their task-specific offline checkpoint and recorded initialization state. These checks do not imply identical subsequent trajectories or bitwise-identical execution across runs. Collection-rule comparisons are reported separately in Appendix I.

Evaluation schedule. For the online learning curves, we evaluate each policy for 50 episodes every 200,000 online environment steps. Evaluation uses greedy execution-length selection.

Aggregation. Online AUC is the trapezoidal integral of the evaluation success curve over the one-million-step online phase, divided by the online budget. Final success is measured at the last checkpoint. For task-level results, we report the mean and sample standard deviation across training seeds. For family-level results, we average tasks within each seed before computing these statistics. Paired differences are computed between runs with the same task and training seed.

Published results. Results reported by prior action-chunking studies (Gireesh et al., 2026; Chen et al., 2026; Shin et al., 2026a) retain their original experimental protocols. Differences in datasets, hyperparameters, training seeds, and evaluation schedules are recorded with the corresponding comparisons. These reference results are not treated as baselines rerun under our experimental settings.

Exploratory objective comparison. We compare the actor and state-value objectives on Cube Triple Task 5. Since the objectives differ during offline training, each configuration has its own offline checkpoint. All configurations retain the same all-prefix TD loss, flow-matching and policy distillation losses, and adaptive execution. The partial-execution backup is disabled.

We replace the full-chunk value in either the actor objective or the state-value target with a weighted average of prefix values, leaving the other objective unchanged. For the actor objective, both the prefix values and their softmax weights are computed from the current critic, with gradients stopped through the weights. For the state-value target, the prefix values are supplied by the target critic and the weights are computed from the current critic. Gradients are stopped through the entire regression target. Both use inverse temperature $\beta = 1$.

E DETAILED BACKUP COMPARISONS

Table 4: Main comparison under softmax collection. Online AUC and final success are percentages. Entries show mean $\pm$ sample standard deviation across three seed-level averages of five tasks. The values use exactly the same comparison sets as Figure 2.

	Online AUC		Final success	
Environment	Without PE backup	PEQC	Without PE backup	PEQC
Cube Triple	72.63 ± 2.09	83.59 ± 0.62	98.40 ± 2.08	99.60 ± 0.40
Cube Quadruple	60.24 ± 8.87	58.63 ± 7.79	74.00 ± 8.66	75.87 ± 3.70
Puzzle 4×4	25.31 ± 10.24	42.07 ± 9.96	26.67 ± 11.55	46.53 ± 11.32
AntMaze	43.29 ± 6.55	56.45 ± 3.34	53.73 ± 6.66	74.53 ± 8.55
HumanoidMaze	59.88 ± 3.83	58.25 ± 3.03	69.33 ± 3.23	59.47 ± 7.20
AntSoccer	68.76 ± 4.59	74.07 ± 3.05	89.60 ± 3.86	88.00 ± 2.43

Table 6 reports per-task results for the main comparison across all six task families. Figure 5 shows the corresponding learning curves. The maze and soccer results use the later softmax-collection experiment, and the Cube Quadruple and Puzzle results use the completed runs with $H = 10$ and $H = 5$, respectively. These are the same comparison sets as Figure 2 and Table 4. Within each task and seed, the methods share an offline checkpoint. The separate Cube Triple auxiliary comparison is reported in Appendix G.

For a family-level result, we first average its five task scores within each training seed. We then compute the mean and sample standard deviation across the three seed averages. For task-level results, the standard deviation is computed across that task's three training seeds. Neither statistic is a confidence interval or an uncertainty estimate over evaluation episodes.

Uncertainty and paired comparisons. We compute 95% percentile confidence intervals with the `rliable` library using 100,000 stratified bootstrap resamples (Agarwal et al., 2021). Within each of the five fixed tasks, we resample the three pairs of runs, retaining both methods and each run's complete learning curve. Figure 2 shows pointwise intervals. Table 5 reports intervals for the paired differences in online AUC and final success. Two-sided paired permutation tests enumerate all 2^{15} swaps of method labels within task–seed pairs, assuming exchangeability under the null. We apply Holm correction across the twelve family–metric comparisons. The confidence intervals are not adjusted for multiple comparisons and remain approximate with three runs per task. The AUC gains on Cube Triple and AntMaze remain significant after correction ($p < 0.05$). Other comparisons do not meet this threshold, but this does not imply that their effects are absent.

Table 5: Paired differences for Figure 2. Differences are PEQC minus the agent without the PE backup, in percentage points. Brackets give 95% bootstrap confidence intervals. The p-values are Holm-adjusted across the twelve comparisons.

	Online AUC		Final success	
Environment	Difference [95% CI]	p	Difference [95% CI]	p
Cube Triple	$+10.96$ $[6.16, 16.27]$	0.016	$+1.20$ $[-0.13, 3.07]$	1.000
Cube Quadruple	-1.61 $[-6.05, 3.08]$	1.000	$+1.87$ $[-2.27, 8.40]$	1.000
Puzzle 4×4	$+16.76$ $[5.32, 28.59]$	0.547	$+19.87$ $[6.40, 33.33]$	1.000
AntMaze	$+13.16$ $[7.59, 18.89]$	0.008	$+20.80$ $[7.33, 34.00]$	0.331
HumanoidMaze	-1.63 $[-4.20, 1.31]$	1.000	-9.87 $[-14.80, -5.33]$	0.051
AntSoccer	$+5.31$ $[2.48, 8.43]$	0.137	-1.60 $[-4.80, 1.73]$	1.000

Table 6: Per-task results for the main backup comparison in Figure 2. Entries are mean $\pm$ sample standard deviation across three training seeds. Online AUC and final success are percentages.

	Online AUC		Final success	
Task	Without PE backup	PEQC	Without PE backup	PEQC
Cube Triple				
Task 1	79.40 ± 18.89	90.13 ± 0.12	100.00 ± 0.00	100.00 ± 0.00
Task 2	58.67 ± 14.16	83.47 ± 2.01	98.67 ± 1.15	100.00 ± 0.00
Task 3	84.53 ± 1.40	87.53 ± 1.17	100.00 ± 0.00	99.33 ± 1.15
Task 4	65.00 ± 9.38	74.33 ± 5.10	95.33 ± 8.08	99.33 ± 1.15
Task 5	75.53 ± 10.69	82.47 ± 2.55	98.00 ± 2.00	99.33 ± 1.15
Cube Quadruple				
Task 1	89.93 ± 0.76	90.13 ± 0.23	100.00 ± 0.00	100.00 ± 0.00
Task 2	77.00 ± 4.76	76.07 ± 3.35	98.00 ± 2.00	96.67 ± 2.31
Task 3	54.20 ± 38.50	56.67 ± 21.03	74.00 ± 41.62	86.67 ± 12.86
Task 4	80.07 ± 4.20	70.27 ± 20.21	98.00 ± 0.00	96.00 ± 4.00
Task 5	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Puzzle 4×4				
Task 1	95.40 ± 1.51	95.93 ± 0.76	100.00 ± 0.00	100.00 ± 0.00
Task 2	0.40 ± 0.53	0.20 ± 0.20	0.00 ± 0.00	0.00 ± 0.00
Task 3	0.80 ± 0.72	53.93 ± 46.54	0.00 ± 0.00	66.67 ± 57.74
Task 4	29.93 ± 51.85	60.27 ± 52.19	33.33 ± 57.74	66.00 ± 57.17
Task 5	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
AntMaze				
Task 1	19.47 ± 11.87	37.67 ± 32.70	58.67 ± 21.57	59.33 ± 44.56
Task 2	46.53 ± 19.62	67.20 ± 7.43	60.00 ± 48.54	92.00 ± 5.29
Task 3	14.13 ± 7.84	18.93 ± 4.92	24.00 ± 24.00	29.33 ± 13.61
Task 4	60.33 ± 19.06	74.93 ± 7.86	94.00 ± 0.00	97.33 ± 2.31
Task 5	76.00 ± 4.23	83.53 ± 2.50	32.00 ± 52.00	94.67 ± 3.06
HumanoidMaze				
Task 1	73.33 ± 6.77	62.47 ± 2.76	82.00 ± 11.14	56.67 ± 11.02
Task 2	79.13 ± 4.80	77.80 ± 5.82	91.33 ± 7.02	92.00 ± 5.29
Task 3	77.40 ± 4.60	75.27 ± 2.32	87.33 ± 3.06	74.67 ± 5.03
Task 4	15.73 ± 8.05	14.53 ± 3.06	14.00 ± 6.93	12.00 ± 3.46
Task 5	53.80 ± 3.86	61.20 ± 9.70	72.00 ± 9.17	62.00 ± 20.88
AntSoccer				
Task 1	77.07 ± 10.06	79.07 ± 4.11	96.00 ± 2.00	87.33 ± 7.02
Task 2	70.40 ± 13.69	80.07 ± 2.41	88.00 ± 4.00	92.00 ± 3.46
Task 3	70.47 ± 7.92	69.47 ± 9.60	86.67 ± 13.32	88.67 ± 6.43
Task 4	56.20 ± 8.90	66.60 ± 8.17	84.67 ± 4.16	80.00 ± 6.00
Task 5	69.67 ± 7.79	75.13 ± 2.10	92.67 ± 4.16	92.00 ± 5.29

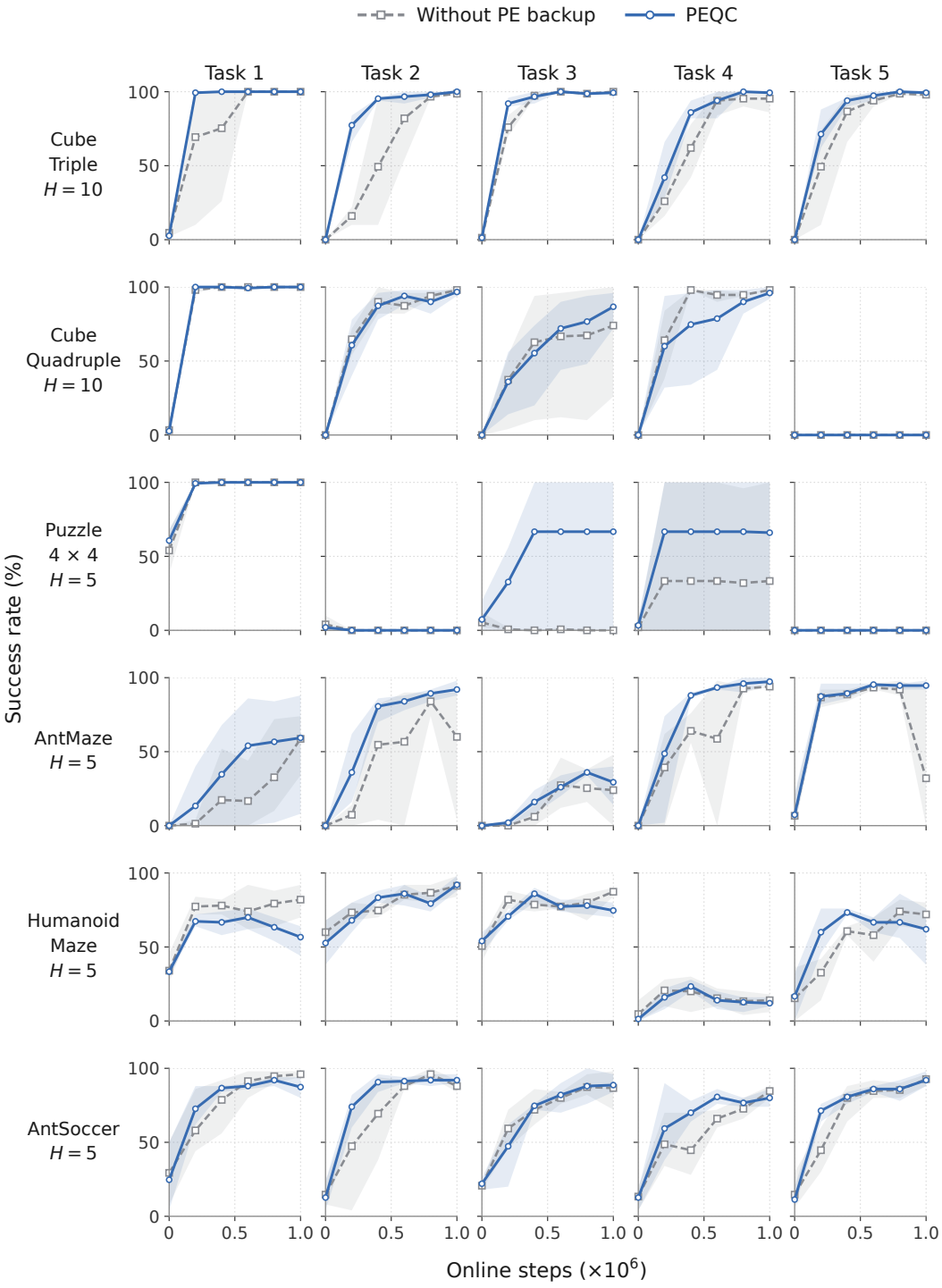


Figure 5: Online learning curves for individual tasks. Rows correspond to task families and columns to Tasks 1–5. Lines show mean success across three training seeds per task. Shading shows pointwise 95% bootstrap confidence intervals, obtained by resampling training runs within each task. These are the same runs as Figure 2.

F PUBLISHED BENCHMARK RESULTS

Table 1 reports mean final success on the six task families. Cube entries are taken from Table 1 of Gireesh et al. (2026) except for ACSAC, which uses Table 1 of Chen et al. (2026), and ACH, which uses Table 1 of Shin et al. (2026a). The other entries are from Tables 1 and 4 of Shin et al. (2026a). The two QC rows retain their source-specific configurations. We do not pool differently reported QC results or substitute results measured with a different dataset size.

Tables 7 and 8 report final success on the six evaluated task families. They retain IQL (Kostrikov et al., 2022), FQL-n (Park et al., 2025b; Li et al., 2025b), DQC (Li et al., 2026), and the fixed-$H = 10$ configuration omitted from the compact main-text table, together with the available uncertainty. Navigation-reference uncertainty denotes 95% confidence intervals over ten seeds, whereas our uncertainty denotes sample standard deviations of three seed-level task averages. These statistics are not interchangeable. Our own complete AUC and final statistics are reported in Appendix E.

Table 7: Five-task mean final success (%) on the cube benchmarks. Published values retain their source protocols and are means over four seeds, except ACH, which uses ten. ACH uses 10M transitions on Cube Quadruple, whereas our runs use the 100M dataset. Our entries show mean ± sample standard deviation over three seed-level task averages. Bold and underlined values indicate the highest and second highest means in each column, respectively.

Method	Cube Triple	Cube Quadruple
IQL	0	0
RLPD	41	0
FQL	18	3
FQL-n	1	36
QC-FQL	53	<u>77</u>
QC	64	73
DQC	0	20
ACSAC	97	73
AQC	90	**88**
ACH	67	52
PEQC w/o PE backup	<u>98.40</u> ± 2.08	74.00 ± 8.66
PEQC	**99.60** ± 0.40	75.87 ± 3.70

Table 8: Five-task mean final success (%) on the maze, soccer, and Puzzle benchmarks. Published entries retain the protocols of Shin et al. (2026a). Their 95% confidence intervals and our sample standard deviations describe different statistics, as indicated below. Our entries use the main softmax-collection runs and Puzzle $H = 5$.

Method	AntMaze	AntSoccer	HumanoidMaze	Puzzle 4×4
Published results, mean ± 95% CI, ten seeds				
RLPD	69±9	64±4	29±9	<u>64</u>±12
BFN	22±6	74±7	<u>87</u>±6	60±7
FQL	74±7	70±5	56±3	54±6
FINO	**76**±4	71±5	**92**±4	53±7
QC (H=5)	47±8	67±2	32±4	38±4
QC (H=10)	20±6	12±2	0±0	20±5
ACH	72±2	84±1	82±1	**90**±8
This work, mean ± sample SD, three seed averages				
PEQC w/o PE backup	53.73 ± 6.66	**89.60** ± 3.86	69.33 ± 3.23	26.67 ± 11.55
PEQC	<u>74.53</u> ± 8.55	<u>88.00</u> ± 2.43	59.47 ± 7.20	46.53 ± 11.32

G AUXILIARY-TARGET COMPARISON DETAILS

Completed five-task comparison. Table 9 reports online AUC and final success for the learner without an auxiliary backup, the executed-prefix backup, and PEQC. Each task family includes five

tasks and three training seeds. All configurations retain the base prefix losses, full-chunk policy objective, and state-value objective. The two auxiliary updates use completed chunk records, a batch size of 256, and coefficient $1/H$ within a single joint optimizer step. The executed-prefix update supervises the value of the actions actually applied, whereas PEQC supervises the original full chunk using Equation 7. The executed-prefix target is $R_t^k + \gamma^k V_{\bar{\phi}}(s_{t+k})$.

For each metric, we average the five tasks within each seed and then report the mean and sample standard deviation across the three seed averages. The Cube Triple comparison uses a separate set of runs from the historical learning curves in Figure 2. The AntMaze and HumanoidMaze comparisons use the later softmax-collection runs. Results from these comparisons are not combined with earlier runs.

Table 9: Completed auxiliary-backup comparison. Both metrics are percentages and are summarized as mean $\pm$ sample SD across three seed means, with five tasks averaged within each seed. Every row uses 15 completed runs.

Family	H	Method	Online AUC	Final success
Cube Triple	10	PEQC w/o PE backup	76.40 ± 5.15	99.47 ± 0.46
		Executed-prefix backup	80.67 ± 7.82	99.73 ± 0.46
		PEQC	83.93 ± 3.50	99.33 ± 0.83
AntMaze	5	PEQC w/o PE backup	43.29 ± 6.55	53.73 ± 6.66
		Executed-prefix backup	53.61 ± 3.34	70.80 ± 14.86
		PEQC	56.45 ± 3.34	74.53 ± 8.55
HumanoidMaze	5	PEQC w/o PE backup	59.88 ± 3.83	69.33 ± 3.23
		Executed-prefix backup	56.32 ± 1.80	58.93 ± 0.23
		PEQC	58.25 ± 3.03	59.47 ± 7.20

PEQC has a higher mean AUC than the executed-prefix backup in each comparison, but both auxiliary backups remain below the agent without the backup on HumanoidMaze. These comparisons evaluate two ways of learning from completed chunk records. They change both the supervised action sequence and the continuation target, rather than isolating the accuracy of the suffix estimate alone.

Earlier auxiliary-target comparisons. Table 10 reports an earlier experiment on Cube Triple Task 2. In addition to the executed-prefix update and PEQC, it includes a replay full-chunk update and a fresh-suffix update. The replay full-chunk update samples extra executed H-step sequences from $\mathcal{D}$ and applies the standard full-length target. The fresh-suffix update samples new remaining actions at the reached state, joins them to the observed prefix, and uses the resulting sequence for both the full-chunk prediction and the suffix bootstrap. Unlike PEQC, the fresh-suffix update requires a new policy sample. All auxiliary updates use the same nominal coefficient $1/H$, a batch size of 256, and one joint optimizer step. Their computational costs need not be equal.

This earlier comparison includes one completed run for each of seeds 2512, 6172, and 9261. It was conducted during method development and is separate from the completed five-task comparison above. Interrupted attempts are not concatenated with completed learning curves.

Table 10: Earlier auxiliary-target comparison on Cube Triple Task 2. Entries show mean $\pm$ sample standard deviation across three training seeds. These runs are separate from the five-task comparison in Table 9.

Configuration	Online AUC (%)	Final success (%)
PEQC w/o PE backup	58.67 ± 14.16	98.67 ± 1.15
Replay full-chunk update	44.80 ± 38.18	66.67 ± 57.74
Executed-prefix backup	76.80 ± 9.46	98.67 ± 1.15
Fresh-suffix update	80.67 ± 5.40	98.67 ± 2.31
PEQC	83.47 ± 2.01	100.00 ± 0.00

Earlier fresh-suffix comparisons also cover Cube Triple Task 4 and HumanoidMaze Task 1. In the order of no auxiliary update, fresh-suffix update, and PEQC, mean online AUC is 65.00, 72.33, and 74.33 on Cube Triple Task 4, and 69.80, 67.13, and 57.80 on HumanoidMaze Task 1. Final success in the same order is 95.33%, 96.67%, and 99.33% on Cube Triple Task 4, and 72.00%, 72.00%, and 60.00% on HumanoidMaze Task 1. These earlier HumanoidMaze results do not replace the later main comparison. They show that retaining the original suffix is not uniformly better than generating a new continuation in the comparisons reported here.

Controlled policy-update comparison. We examine how additional critic supervision affects subsequent policy updates on AntMaze, HumanoidMaze, and AntSoccer. For each task and seed, all conditions start from the same PEQC checkpoint and use the same newly collected data and replay minibatches. We use checkpoints at 200K online steps for AntMaze and AntSoccer, and 600K for HumanoidMaze. The conditions differ only in the auxiliary critic loss: no auxiliary loss, an executed-prefix backup, or the partial-execution backup.

We perform 100 critic updates while holding the actor and target networks fixed. We then freeze each critic and perform 10 actor updates from the same initial actor parameters and optimizer state, using the same state batches, noise, and behavior-policy targets.

For evaluation, we execute the full chunk generated by each updated actor and then use the same frozen reference actor with full-length execution for the rest of the rollout. We compare the resulting discounted return with that of the initial actor under the same evaluation protocol. Each task–seed pair uses eight held-out states from two episodes, two actor noise samples per state, and four continuation rollouts per sample. Rollouts use $H = 5$ and $\gamma = 0.995$, terminate at a terminal state or after 3,200 steps, and use no value bootstrap.

Table 11: Controlled policy-update comparison. Entries report the change in discounted return relative to the initial actor after 10 actor updates. We average the five tasks within each training seed and report the mean $\pm$ sample standard deviation across three seeds. Positive values indicate improvement over the initial actor. Each evaluated actor supplies only the initial full chunk, with a shared frozen reference policy used thereafter.

Environment	No auxiliary loss	Executed-prefix	PE backup
AntMaze	-0.390 ± 0.523	-0.135 ± 0.116	$+0.027 \pm 1.084$
HumanoidMaze	$+0.268 \pm 0.903$	$+0.006 \pm 0.887$	-0.548 ± 0.842
AntSoccer	$+0.916 \pm 3.087$	$+1.839 \pm 1.137$	$+3.415 \pm 0.208$

Table 11 shows that the PE backup produces a larger mean return improvement than both alternatives on AntSoccer, with this ordering holding for all three seed-level task averages. On AntMaze, its mean return change is close to zero but higher than those of the alternatives. On HumanoidMaze, the PE backup yields a negative mean change, while learning without an auxiliary loss gives the highest mean change. These comparisons show that additional full-chunk supervision can support useful actor updates, with benefits that vary across the tested settings.

H Execution-length measurements

Final-checkpoint evaluation. The comparison in Section 4.3 uses 90 final PEQC checkpoints, covering six task families, five tasks per family, and three training seeds per task. Each checkpoint is evaluated for 50 episodes under each rule, for a total of 13,500 evaluation episodes. Model parameters and optimizer states remain fixed. Table 12 reports success rates, and Table 13 reports the paired differences.

Adaptive execution selects the prefix with the largest ensemble-mean Q-value. Uniform execution samples a length from $1, \ldots, H$ independently at each decision, and fixed execution selects H. The policy generates an entire chunk under every rule. The selected actions are applied until that length is reached or an episode boundary interrupts execution. Policy sampling, action clipping, and episode termination handling are unchanged across the three rules.

Table 12: Success (%) of the same final PEQC checkpoints under three execution rules. Means ± sample SD across three seed-level averages of five tasks. There are 50 episodes per checkpoint and rule. Only evaluation changes.

Family	H	Adaptive	Uniform	Fixed H
AntMaze	5	74.9 ± 6.7	61.9 ± 9.5	54.5 ± 2.9
HumanoidMaze	5	58.9 ± 4.0	35.7 ± 2.0	26.7 ± 4.7
AntSoccer	5	88.0 ± 1.2	87.6 ± 2.2	83.9 ± 1.8
Cube Triple	10	99.6 ± 0.7	99.9 ± 0.2	99.3 ± 0.2
Cube Quadruple	10	74.8 ± 4.7	75.1 ± 4.4	74.3 ± 3.4
Puzzle 4×4	5	46.7 ± 11.5	46.7 ± 11.5	46.7 ± 11.5

Evaluation initialization. Each rule uses the same initial environment states for a checkpoint. Actor randomness is initialized separately for each episode, with an independent generator for uniform length selection. Subsequent states and decision counts may differ between rules.

The original learning-curve evaluations used an actor random key carried across the 50 episodes. This re-evaluation instead initializes it separately for each episode, so changes in the number of decisions in one episode do not change the actor's initial random key in the next. In addition, the Cube Triple checkpoints come from the completed five-task auxiliary comparison, not the historical main learning-curve experiment. The re-evaluation scores therefore remain separate from the final scores reported with the learning curves.

Table 13: Paired success differences in percentage points for the final-checkpoint evaluation. Differences are formed within each task and seed, averaged over the five tasks, and summarized as mean ± sample SD across the three seed means. These are descriptive differences, not confidence intervals or significance tests.

Family	Adaptive minus uniform	Adaptive minus fixed H
AntMaze	$+13.1 \pm 6.1$	$+20.4 \pm 4.5$
HumanoidMaze	$+23.2 \pm 5.8$	$+32.3 \pm 2.0$
AntSoccer	$+0.4 \pm 2.1$	$+4.1 \pm 2.9$
Cube Triple	-0.3 ± 0.5	$+0.3 \pm 0.6$
Cube Quadruple	-0.3 ± 1.4	$+0.5 \pm 2.3$
Puzzle 4×4	$+0.0 \pm 0.0$	$+0.0 \pm 0.0$

For each task and seed, a success difference is formed between two rules applied to the same checkpoint. We average these differences over the five tasks within each seed, then report the mean and sample standard deviation across the three seed averages. In Puzzle, the same seven checkpoints achieve 100% success under all three rules and the remaining eight achieve 0%. The return and episode length still differ for many checkpoints. Equal success rates therefore do not imply identical behavior. The comparison measures execution-rule effects at evaluation. It does not compare policies trained with uniform and value-based collectors. The rules also change the average execution length and replanning frequency, so the success differences do not isolate prefix-value ranking accuracy. Decision-level length traces were retained for this re-evaluation. The collection statistics and original evaluation measurements below refer to separate records.

Collection execution lengths. We use the same 45 task–seed pairs as the softmax AntMaze, HumanoidMaze, and AntSoccer performance comparisons. We reconstruct actual execution length k from consecutive decision steps and episode ends, rather than treating the selected length as always executed. Chunks beginning within the first 5,000 steps and unfinished chunks at the training budget are excluded. Each completed chunk has equal weight within a run. We then average tasks within seed and report the mean and sample standard deviation across three training seeds. Table 14 summarizes collection execution statistics, and Figure 6 shows the execution-length distribution.

Table 14: Collection execution statistics with $H = 5$. The suffix-bootstrap column counts partial chunks with nonzero continuation masks. It describes completed records, not the records actually sampled in auxiliary minibatches.

Method	Partial (%)	Mean k/H	Suffix bootstrap (%)
AntMaze			
PEQC w/o PE backup	80.07 ± 0.07	0.5941 ± 0.0007	79.92 ± 0.07
PEQC	80.48 ± 0.15	0.5950 ± 0.0017	80.30 ± 0.16
HumanoidMaze			
PEQC w/o PE backup	78.70 ± 0.13	0.5912 ± 0.0007	78.62 ± 0.13
PEQC	79.91 ± 0.06	0.5901 ± 0.0004	79.84 ± 0.06
AntSoccer			
PEQC w/o PE backup	79.32 ± 0.23	0.5858 ± 0.0021	79.01 ± 0.23
PEQC	80.76 ± 0.23	0.5815 ± 0.0029	80.43 ± 0.23

Uniform selection over five lengths would yield 80% partial execution and a mean k/H of 0.60 without interruptions. This is an analytic reference, not an additional training run. Similar marginal distributions do not imply equal state visitation or equal conditional length distributions.

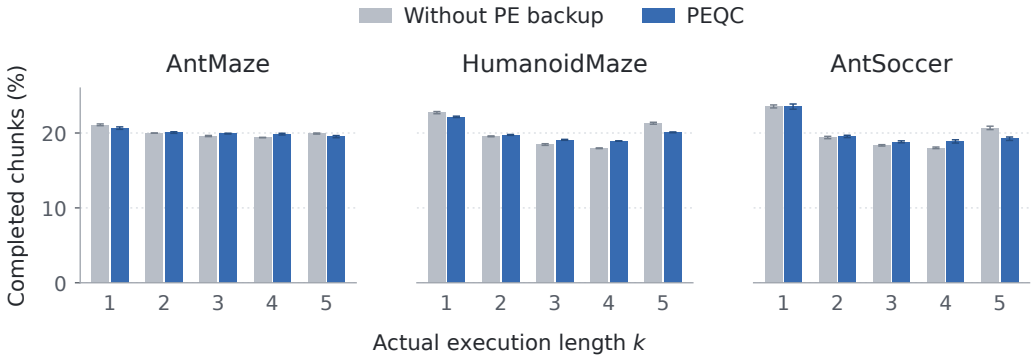


Figure 6: Actual execution lengths during softmax collection with $H = 5$. Bars and error bars show the mean and sample standard deviation across three seed-level averages of five tasks. Only completed post-warm-up chunks are included.

For the original learning-curve runs, mean execution length at the final greedy evaluation is computed as the total number of primitive steps divided by the total number of policy calls across 50 episodes in each run. Per-decision traces were not retained for those evaluations, so their execution length histograms and partial execution rates cannot be inferred from these means. The ratios k/H for the agent without the backup and PEQC are 0.4921 and 0.5500 on AntMaze, 0.5250 and 0.5109 on HumanoidMaze, and 0.5385 and 0.5050 on AntSoccer. These historical measurements are separate from the new execution rule comparison in Section 4.3.

Execution-length distribution at evaluation. Figure 7 shows the execution lengths selected by PEQC during final-checkpoint evaluation on AntMaze, HumanoidMaze, and AntSoccer, all with $H = 5$. The greedy selector uses all five lengths, with $k = 1$ selected most frequently in each task family. Full-length execution ($k = H$) accounts for approximately 19–21% of selections.

I Collection rule sensitivity

Collection rules. We compare softmax, greedy, and episode-level mixture collection for PEQC and the learner without the PE backup on AntMaze, HumanoidMaze, and AntSoccer. Each configuration uses five tasks per family and three training seeds. The mixture selects softmax or greedy mode with equal probability at each episode reset and retains that mode for the episode. Greedy

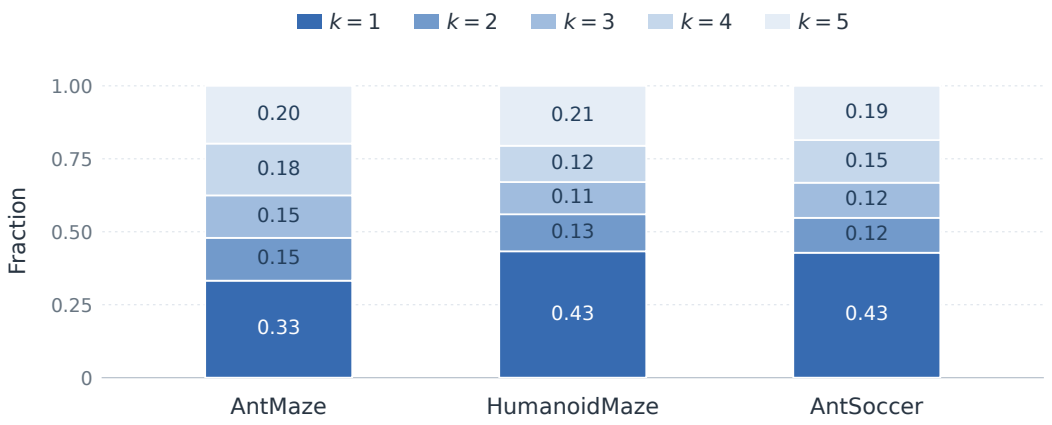


Figure 7: Execution-length distribution of PEQC at the final checkpoint (1M online steps). Each stacked bar shows the fractions of selected lengths $k \in \{1, \ldots, 5\}$ under greedy adaptive evaluation. Fractions are computed over decisions in 50 episodes per checkpoint, averaged over five tasks within each seed, and then across three training seeds.

chooses the highest-valued prefix at each decision, not one fixed length for the episode. Chunk generation remains stochastic. Every collector is evaluated using greedy execution lengths. These are separately trained agents, not the fixed-checkpoint execution-rule comparison.

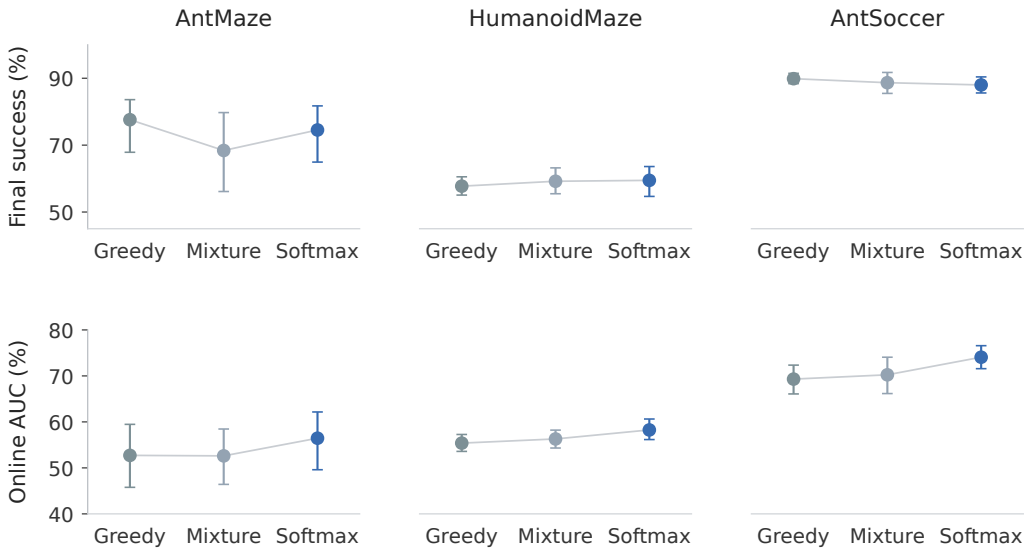


Figure 8: Final success (top) and online AUC (bottom) of PEQC under three collection rules. Points show means across five tasks and three training seeds per task. Bars show 95% bootstrap confidence intervals, obtained by resampling training runs within each task.

For PEQC, softmax collection has the highest observed mean AUC in all three families. Greedy has the highest final mean success in AntMaze and AntSoccer. No rule has the highest mean on both metrics across all families. Table 15 reports both learners under all three collection rules. For the learner without the PE backup, greedy achieves the highest mean AUC and final success on AntMaze and AntSoccer, while softmax is highest on HumanoidMaze. All configurations use $H = 5$ and the family-specific training settings of the main comparison. The PE backup uses $\lambda = 0.2$.

Table 15: Collector sensitivity over five tasks and three training seeds. We average the five tasks within each seed and report the mean ± sample standard deviation across the three seeds.

Family	Method	Greedy	Mixture	Softmax
Online AUC (%)				
AntMaze	PEQC w/o PE backup	51.45 ± 2.83	45.33 ± 4.52	43.29 ± 6.55
	PEQC	52.72 ± 6.09	52.61 ± 3.34	56.45 ± 3.34
HumanoidMaze	PEQC w/o PE backup	56.12 ± 0.97	56.96 ± 2.79	59.88 ± 3.83
	PEQC	55.39 ± 2.73	56.29 ± 2.49	58.25 ± 3.03
AntSoccer	PEQC w/o PE backup	72.75 ± 3.29	70.80 ± 3.33	68.76 ± 4.59
	PEQC	69.31 ± 2.55	70.24 ± 4.31	74.07 ± 3.05
Final success (%)				
AntMaze	PEQC w/o PE backup	76.00 ± 5.60	68.00 ± 13.95	53.73 ± 6.66
	PEQC	77.60 ± 10.74	68.40 ± 12.32	74.53 ± 8.55
HumanoidMaze	PEQC w/o PE backup	63.47 ± 0.83	64.93 ± 4.39	69.33 ± 3.23
	PEQC	57.73 ± 0.83	59.20 ± 4.72	59.47 ± 7.20
AntSoccer	PEQC w/o PE backup	90.80 ± 1.74	89.07 ± 4.09	89.60 ± 3.86
	PEQC	89.87 ± 1.85	88.67 ± 3.26	88.00 ± 2.43

J WALL-CLOCK TIME ANALYSIS

Figure 9 shows wall-clock learning curves for AntMaze, HumanoidMaze, and AntSoccer using the same training runs as Figure 2.

We measure wall-clock time from the timestamp of the initial online checkpoint to the completion of each evaluation. Offline training and setup before this checkpoint are excluded. The measurements include online training, data collection, evaluation, checkpoint saving, and runtime overhead.

For each environment, we compare both methods over the time interval covered by every run. We linearly interpolate success between recorded evaluations without extrapolation. We first average the five tasks within each seed, then report the mean and sample standard deviation across the three seeds.

Time-normalized AUC is the area under the success curve over this common interval, divided by its duration. We compute the completion time ratio for each task and seed. We then average these ratios across tasks and seeds. Table 16 reports the common time intervals and final evaluation times. Table 17 reports time-normalized AUC.

Table 16: Wall-clock time measured from the initial online checkpoint. Final evaluation times are averaged across tasks and seeds. Ratios are computed for each task and seed and then averaged.

	Common interval (minutes)	Final evaluation (minutes)		
Environment		w/o PE backup	PEQC	Paired ratio
AntMaze	4.49–126.12	129.80	171.60	1.323
HumanoidMaze	6.28–132.96	139.69	188.67	1.352
AntSoccer	2.54–120.19	124.12	177.62	1.432

Table 17: Mean time-normalized AUC (%) over the observed common interval. Differences use unrounded seed-level values. The uncertainty is the sample standard deviation of the three paired seed differences.

Environment	w/o PE backup	PEQC	Difference
AntMaze	43.93	51.19	$+7.25 \pm 7.35$
HumanoidMaze	59.78	58.05	-1.73 ± 3.15
AntSoccer	67.92	67.58	-0.34 ± 1.56

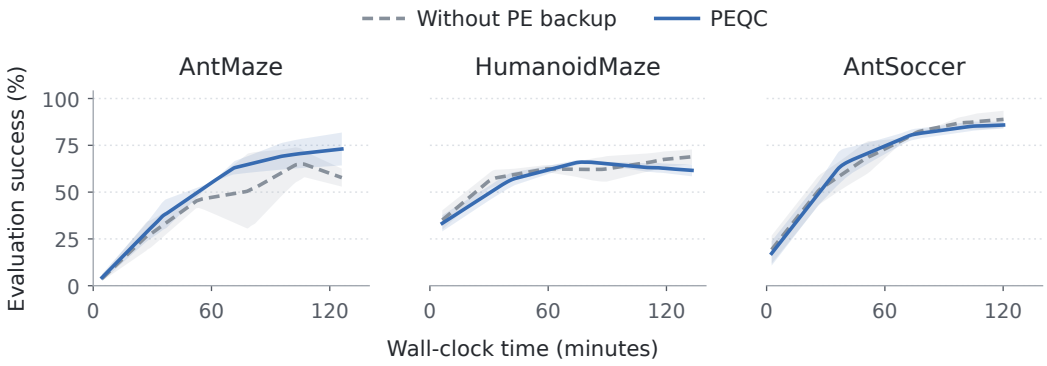


Figure 9: Wall-clock learning curves over the common recorded interval. Shading shows the sample standard deviation across three seed-level task means. Timing includes evaluation, saving, compilation, and runtime contention.