HIERARCHICAL TIME-AWARE BOOTSTRAPPING FOR OFF-POLICY SUBGOAL VALUE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Off-policy hierarchical reinforcement learning must estimate the values of high-level decisions while the low-level policy changes. HIRO adapts replay data through subgoal relabeling, but after a label change, the value update targets the relabeled subgoal instead of the subgoal the high-level policy originally needed to update. We propose Hierarchical Time-aware Bootstrapping (HTB), which evaluates specified subgoals under the current low-level policy while retaining accumulated task rewards. Remaining execution time distinguishes subgoal continuation from a new high-level decision. Together with primitive-action conditioning, it enables off-policy Bellman updates based on the stationary environment transition law. HTB combines these one-step updates with multi-step suffix returns and truncated relabeling, reducing dependence on intermediate value estimates. A shared value component supports learning across actions, while nonnegative residuals constrain upward corrections relative to that component. At a fixed mixture weight of .95, HTB achieves 32.8% AntFall success versus 9.6% for matched local HIRO over five paired seeds at 10M environment steps. Ablations identify contributions from recursive continuation and mixed supervision; fixed-policy tests show more accurate predictions for actions whose returns were excluded from fitting.

1 INTRODUCTION

Goal-conditioned hierarchical reinforcement learning (HRL) selects subgoals with a high-level policy and executes them with a low-level policy. As the latter changes, the same subgoal induces different transitions and returns, making high-level replay nonstationary.

Hierarchical Actor-Critic (HAC) corrects replay with hindsight actions under idealized lower-level execution. Its correction relies on a restrictive assumption: the high-level reward must be independent of the low-level execution path, as explicitly required by Levy et al. (2019, Sec. 4.3). The standard goal-reaching cost is -1 before success and 0 at the goal. Endpoint hindsight preserves this path-independent objective, but fails to preserve general accumulated trajectory rewards. For example, two policies may reach the same spatial subgoal with accumulated rewards of 1 and 10; substituting the endpoint does not make their returns interchangeable. This mechanism is therefore not a general correction for hierarchical nonstationarity.

HIRO addresses general trajectory rewards through likelihood-based relabeling: it chooses a subgoal that explains the recorded primitive actions under the current low-level policy (Nachum et al., 2018). This also changes which subgoal receives value supervision. When the label changes, that regression supplies no direct target for the original high-level decision. Yet the high-level policy needs to evaluate its recorded decisions and future proposals, including subgoals that do not explain the historical trajectory. We ask: *how can replay update the value of a specified subgoal under the current low-level policy?*

We introduce *Hierarchical Time-aware Bootstrapping* (HTB). Conditioning on the recorded primitive action exposes the stationary environment transition law. Remaining execution time specifies whether the current subgoal continues or a new high-level decision follows. Together, they provide primitive-step Bellman updates for a chosen subgoal while preserving the hierarchy's decision period. HTB mixes these updates with multi-step suffix returns and truncated relabeling, replacing intermediate predictions with observed rewards. A shared subgoal value transfers updates across

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.

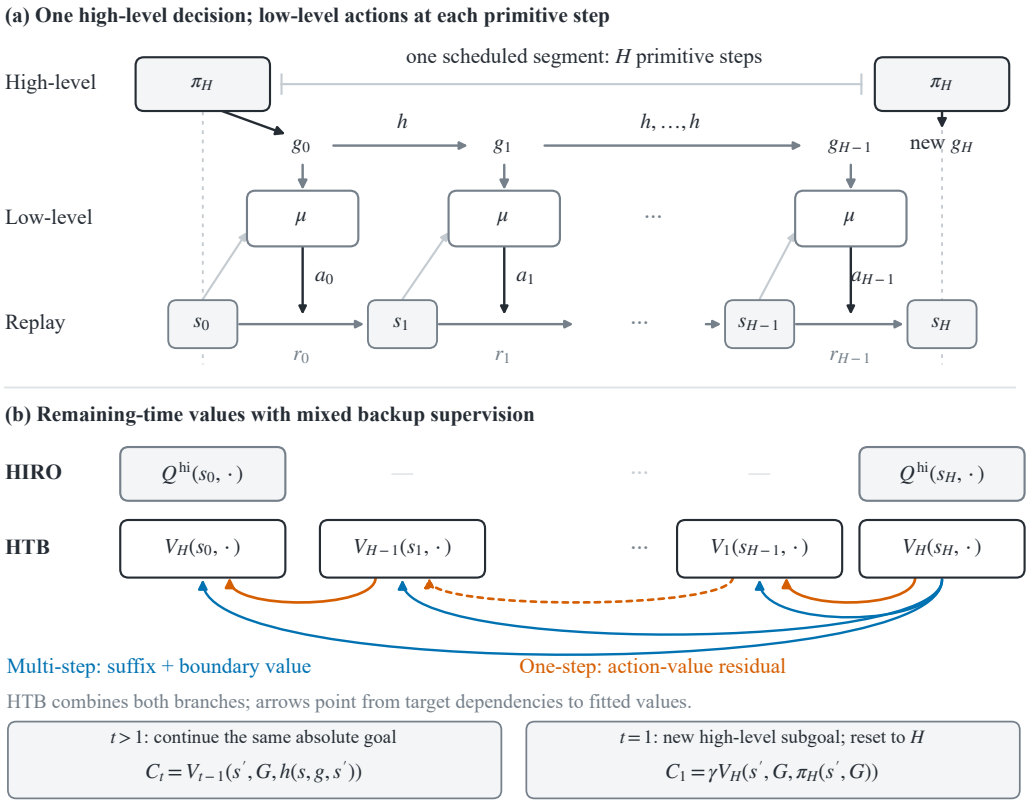


Figure 1: **Subgoal values can be learned at every primitive step.** The high-level policy acts every H steps. HTB learns $V_{H-i}(s_i, G, g_i)$ throughout each segment using a mixture of its remaining return (blue) and a one-step backup (orange). The current subgoal continues until the next high-level decision. Arrows run from target dependencies to the value being updated; displayed continuations assume no termination.

primitive actions; action-value residuals supply action-specific corrections, with nonnegativity constraining upward amplification.

Our contributions are the time-aware Bellman formulation, its mixed-backup and shared-value implementation, and evidence from task success, component ablations and action-value prediction under limited supervision.

2 Subgoal evaluation under a changing low-level policy

Let s denote the environment state, G the task goal, g a subgoal, and a a primitive action. A high-level policy $\pi_H(s, G)$ selects a subgoal every H steps; the low-level policy $\mu(a \mid s, g)$ acts at each step. For achieved-goal coordinates $\phi(s)$, the goal-transition function $h(s, g, s') = g + \phi(s) - \phi(s')$ keeps the absolute destination unchanged. Replay stores a segment $\tau = (s_0, a_0, r_0, \ldots, s_L)$, its original subgoal g^{orig}, and its observed length $L \leq H$. A segment ends at the scheduled boundary or episode termination, so $L < H$ indicates early termination. Here r is the scaled task reward, not the low-level intrinsic reward.

HIRO chooses a relabeled subgoal $\tilde{g}$ by approximately maximizing the likelihood of the recorded action sequence under the current low-level policy. Denoting its high-level critic by Q^{hi}, the resulting regression has the form

$$\mathcal{L}_{\text{corr}} = \mathbb{E}_{\tau \sim \mathcal{D}} \left[\ell\big(Q^{\text{hi}}(s_0, G, \tilde{g}), Y_\tau\big) \right], \tag{1}$$

where $\mathcal{D}$ is the replay distribution, ℓ is a regression loss, and Y_τ is a stopped-gradient return target. The supervised input is therefore the relabeled subgoal. When $\widetilde{g} \neq g^{\text{orig}}$, the original high-level decision receives no direct target from this regression, even though parameter sharing may change its prediction indirectly.

Our objective is to evaluate an explicitly chosen subgoal g under the current low-level policy, including the original replay subgoal or a proposal from the current high-level policy. Its value covers the remaining execution duration and subsequent high-level decisions. Because different subgoals induce different future actions, we learn primitive-action values from recorded transitions and evaluate them at the current policy's action. Remaining time identifies whether the continuation executes the chosen subgoal or returns control to the high-level policy. This construction separates the subgoal receiving supervision from the subgoal selected by likelihood-based correction.

3 Hierarchical Time-aware Bootstrapping

3.1 Remaining-time Bellman updates

Let $t \in \{1, \dots, H\}$ be the number of primitive steps until the next scheduled high-level decision. For fixed policies, $V_t(s, G, g)$ is the expected task return from executing subgoal g for these t steps and following the hierarchy afterward. Define $Q_t(s, G, g, a)$ by fixing the first primitive action to a. Thus V_t is a subgoal value, while Q_t also conditions on the primitive action. Both estimate the external task return, distinct from the intrinsic-reward objective used to train the low-level policy. At a decision boundary, V_H evaluates the high-level action.

Within a high-level segment, rewards are summed without discounting. A factor γ is applied at the next high-level decision, matching the training objective. With terminal indicator d, the Bellman equations are

$$Q_t(s, G, g, a) = \mathbb{E}[r + (1 - d)C_t(s', G, h(s, g, s')) \mid s, a, G], \tag{2}$$

$$C_t(s', G, g') = \begin{cases} V_{t-1}(s', G, g'), & t > 1, \\ \gamma V_H(s', G, \pi_H(s', G)), & t = 1, \end{cases} \qquad V_t = \mathbb{E}_{a \sim \mu(\cdot|s,g)} Q_t(s, G, g, a). \tag{3}$$

The two cases distinguish continuing the current subgoal from choosing the next one (Figure 1). For example, with $H = 10$, a transition at step $i = 4$ updates V_6, whose one-step continuation is V_5. A value function that cannot distinguish these remaining durations can assign incompatible targets to the same input (Pardo et al., 2018) (Appendix A.2).

Proposition 1 (Off-policy primitive-step evaluation)**.** *Assume a stationary Markov environment with joint transition law $P(s', r, d \mid s, a, G)$ independent of the commanded subgoal. Suppose replay sampling preserves this conditional law, and the subgoal to be evaluated is fixed or selected independently of the transition outcome given (s, a, G). For any fixed bounded continuation, a replay transition has the conditional Bellman expectation in Equation* (2)*, independently of the data-collection policy.*

The same recorded transition can therefore support evaluation of a specified subgoal different from its collection subgoal. Current-policy evaluation uses Q_t on the current policy's action support. The conditional transition kernel remains stationary while the continuation tracks the improving hierarchy. Appendix A develops the signed residual recursion and the sampling and coverage conditions behind this construction.

3.2 Multi-step returns with truncated goal relabeling

The implementation uses a deterministic low-level actor, denoted $\mu(s, g)$. A one-step recursion repeatedly bootstraps through intermediate value estimates. A multi-step target replaces these intermediate estimates with observed rewards and bootstraps once at the next high-level boundary. For a suffix starting at step i, let $t = H - i$ and define

$$y^{\text{L}}_{t,i} = \sum_{k=i}^{L-1} r_k + (1 - d_{L-1})\gamma \bar{V}_H(s_L, G, \widehat{g}_H(s_L)). \tag{4}$$

The implementation uses twin value networks $V_{t,j}$, $j \in \{1, 2\}$. Bars denote target networks, $\bar{V}_t = \min_j \bar{V}_{t,j}$, and $\widehat{g}_H$ is the target high-level action with target-policy smoothing (Fujimoto et al., 2018).

Because the suffix was collected by an earlier low-level policy, we adapt its subgoal using HIRO-style likelihood correction. The likelihood must use the same suffix as the return:

$$\widetilde{g}_i = \arg\min_{g \in \mathcal{C}_i} \sum_{k=i}^{L-1} \|a_k - \mu(s_k, g + \phi(s_i) - \phi(s_k))\|_2^2 , \tag{5}$$

where $\mathcal{C}_i$ contains the transported replay subgoal, the achieved displacement, and Gaussian candidates around that displacement. This is *truncated goal relabeling*: actions before i are excluded. Figures abbreviate this operation as suffix correction. Each suffix thus selects a goal for its own action window; the full segment defines a different likelihood objective. The multi-step loss fits $V_t(s_i, G, \widetilde{g}_i)$ to $y^{\mathrm{L}}_{t,i}$, while the one-step update retains its explicitly specified subgoal.

The observed reward sum removes the learned intermediate continuations from this target. Its remaining estimation errors include the boundary value and approximate likelihood correction; Appendix A.7 separates these sources. Early termination removes the boundary bootstrap. Remaining time is scheduled time $H - i$, even when $L < H$; only observed steps contribute.

3.3 Shared subgoal values and action-value residuals

Primitive-action conditioning introduces an additional input dimension. When the low-level policy changes, the new action may have little direct training data. Inspired by dueling value decomposition (Wang et al., 2016), we share a subgoal value across primitive actions:

$$\widehat{Q}_{t,j}(s, G, g, a) = V_{t,j}(s, G, g) - A_t(s, G, g, a), \tag{6}$$

$$\Delta\widehat{Q}_{t,j}(s, G, g, a) = \Delta V_{t,j}(s, G, g) \quad \text{for every } a \text{ when } A_t \text{ is fixed.} \tag{7}$$

The network A_t estimates an *action-value residual*. For fixed policies, its exact signed form is $A^{\mu,\pi}_t = V^{\mu,\pi}_t - Q^{\mu,\pi}_t$, the negative of the conventional primitive-action advantage. The learned A_t is a nonnegative approximation. The shared $V_t(s, G, g)$ acts as an action-independent value offset (bias), while A_t supplies action-specific corrections. Updating this offset contributes the same shift to every action value, including actions not supervised by that update (Equation (7); Figure 2c).

For a specified subgoal g_i, put $g_i^+ = h(s_i, g_i, s_{i+1})$. HTB corrects the next-step continuation using the next residual:

$$\bar{C}_{t,i} = \begin{cases} \bar{V}_{t-1}(s_{i+1}, G, g_i^+) - \bar{A}_{t-1}(s_{i+1}, G, g_i^+, \mu(s_{i+1}, g_i^+)), & t > 1, \\ \gamma \bar{V}_H(s_{i+1}, G, \widehat{g}_H(s_{i+1})), & t = 1, \end{cases} \tag{8}$$

$$y^A_{t,i} = \left[\bar{V}_t(s_i, G, g_i) - r_i - (1 - d_i)\bar{C}_{t,i}\right]_+ , \tag{9}$$

where $[x]_+ = \max(x, 0)$. At fixed (s_i, G, g_i, t), the one-step value target averages over the low-level policy's primitive action:

$$\begin{aligned} y^{\mathrm{S}}_{t,i} &= \mathbb{E}_{a \sim \mu(\cdot|s_i, g_i)}\left[\bar{V}_t(s_i, G, g_i) - \bar{A}_t(s_i, G, g_i, a)\right] \\ &= \bar{V}_t(s_i, G, g_i) - \bar{A}_t(s_i, G, g_i, \mu(s_i, g_i)) \quad \text{for deterministic } \mu. \end{aligned} \tag{10}$$

With target-policy smoothing, the implementation uses the sampled target

$$\widehat{y}^{\mathrm{S}}_{t,i} = \bar{V}_t(s_i, G, g_i) - \bar{A}_t(s_i, G, g_i, \widehat{a}_i). \tag{11}$$

Here $\widehat{a}_i$ adds clipped smoothing noise to the current low-level action and clips to the action bounds. It targets the smoothed policy; Equation (10) gives the noise-free case. The residual network is fitted to $y^A_{t,i}$ at the *recorded* action a_i. It is then evaluated at the *current-policy* action to construct $\widehat{y}^{\mathrm{S}}_{t,i}$, the target for $V_{t,j}(s_i, G, g_i)$. This distinction is what allows recorded transitions to evaluate a specified subgoal under the current policy. All targets are stopped-gradient; the residual loss updates A and the value losses update V. The next-state action in $\bar{C}_{t,i}$ is unsmoothed.

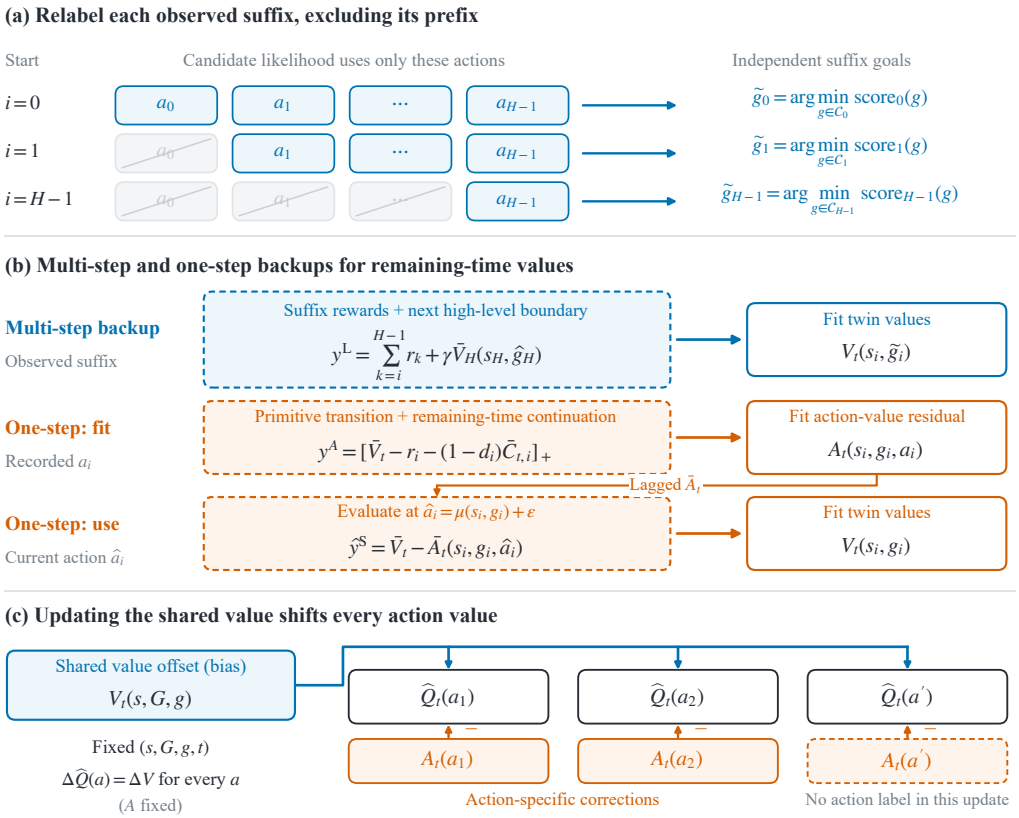


Figure 2: **Multi-step and one-step updates can supervise different subgoals.** (a) Each suffix is relabeled using only its recorded actions. (b) The multi-step target trains the relabeled subgoal; the one-step estimator fits the recorded action and evaluates the current-policy action for the specified subgoal. (c) Updating V changes the shared value offset in every primitive-action value, including actions without direct supervision; for fixed A, $\Delta\widehat{Q}(a) = \Delta V$. Repeated task-goal inputs and action clipping are omitted; the diagram shows a nonterminal segment.

Nonnegative residuals. The practical algorithm constrains both the residual output and target to be nonnegative. For a scalar reference q and Bellman target b, exactly fitting the projected residual gives $q - [q-b]_+ = \min(q, b)$. The reconstructed value therefore cannot exceed its current reference. This one-sided correction filters upward target amplification relative to the learned reference, including amplification introduced by maximization. The cap is reference-relative: its effect on overestimation and underestimation depends on the accuracy of that reference. Appendix A characterizes the projection and its signed counterpart.

3.4 Combined objective and ablations

Using smooth-L_1 loss ℓ and $\lambda \in [0, 1]$, the value objective is

$$\mathcal{L}_V = \lambda\, \mathbb{E}_{\mathcal{D}} \sum_{j=1}^{2} \ell\big(V_{t,j}(s_i, G, \tilde{g}_i), y^{\mathrm{L}}_{t,i}\big) + (1-\lambda)\, \mathbb{E}_{\mathcal{D}} \sum_{j=1}^{2} \ell\big(V_{t,j}(s_i, G, g_i), \hat{y}^{\mathrm{S}}_{t,i}\big)\,. \tag{12}$$

We average valid samples within each remaining-time branch and sum over branches. The weight λ controls multi-step learning; $\lambda = 1$ removes one-step value supervision. For $\lambda < 1$, the one-step

Table 1: **Controlled variants of the HTB update.** All variants use remaining-time values and the same $V - A$ architecture. The last two variants use the nonrecursive residual estimator.

Variant	One-step subgoal	Next residual	Output / target
HTB	Replay	Included	Nonnegative / clipped
No next residual	Replay	Omitted	Nonnegative / clipped
Signed residual	Replay	Omitted	Signed / signed
Policy subgoal	Current proposal	Omitted	Nonnegative / clipped

term directly updates the specified subgoal whenever its prediction differs from the target, even if relabeling selects another subgoal for the multi-step loss (Appendix A.8).

Table 1 defines the ablations. The default uses the transported replay subgoal in the one-step loss. The current-policy-subgoal variant uses a stopped-gradient proposal $\pi_t(s_i, G)$. Disabling relabeling changes only the subgoal used in the multi-step loss. Only π_H executes during interaction; shorter-duration actors provide proposals for the corresponding replay updates. Algorithm 1 and implementation details appear in Appendix B.

4 Experiments

The experiments test whether specified-subgoal value updates improve hierarchical control, which backup components produce the gains, and how the shared value representation supports action-conditioned estimation. Push and Fall combine ant locomotion with object manipulation. Reacher, Maze, and MazeW provide goal-reaching tasks with dense and sparse rewards. Push/Fall comparisons use five paired seeds and 10 million environment steps; Reacher/Maze comparisons use eight paired seeds and one million steps. Success is measured at the final episode step for Push/Fall, and averaged over the last twenty evaluations for Reacher/Maze. The main comparisons use a fixed multi-step weight $\lambda = .95$ and nominal 95% paired-bootstrap intervals. Highlighted component settings were selected retrospectively; the appendix includes the full mixture and relabeling sweeps, with both effect directions. Appendices C, D, and E give protocols, complete results, and statistical procedures.

4.1 Task success at a fixed backup configuration

HTB improves Fall success at the same interaction budget as the local HIRO implementation (Figure 3). With truncated relabeling, it reaches 32.8% success, compared with 9.6% for HIRO. The paired improvement is 23.2 percentage points [8.8, 39.2]. On Push, the respective means are 14.0% and 0.0%; the interval [0.0, 32.4] includes zero. The return curves and integrated-return comparisons are reported separately in Figure 8 and Table 11.

The fixed-setting comparison also extends to Reacher and Maze (Table 8). On Reacher-v1 with sparse rewards, with relabeling disabled in all compared algorithms, HTB achieves 96.1% success versus 76.0% for the archived HIRO baseline, a difference of 20.1 points [2.5, 42.1]. The multi-step-only baseline reaches 95.4% on this task, so high success is also attainable with multi-step-only supervision. These fixed-setting results evaluate complete algorithms; the matched controls below assess residual recursion and mixed supervision.

4.2 Residual recursion and mixed backups improve success

To measure the contribution of recursive continuation, the first control removes only the next-step residual from Equation (8). It retains the same subgoal, value decomposition, nonnegative constraint, relabeling condition, and mixture weight. Restoring residual recursion improves success by 32.8 points [14.4, 47.6] on Fall and 26.9 points [12.0, 40.8] on Reacher-v0 dense (Figure 4a).

The second control compares the mixed objective at $\lambda = .95$ with multi-step-only learning at $\lambda = 1$, keeping the remaining-time architecture and relabeling condition fixed. The mixture improves Fall success by 19.2 points [8.0, 31.6] with relabeling and 19.6 points [1.2, 43.2] without it (Figure 4b).

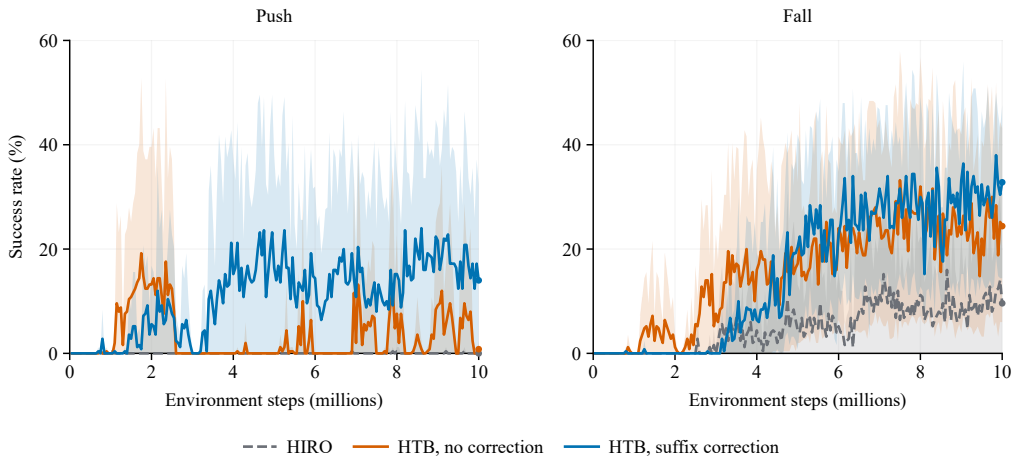


Figure 3: **HTB improves success on Fall at a fixed configuration.** Success over training, averaged across five paired seeds. Both HTB conditions use $\lambda = .95$; HIRO uses its original off-policy correction. Shading denotes pointwise 95% empirical bootstrap intervals. No temporal smoothing is applied.

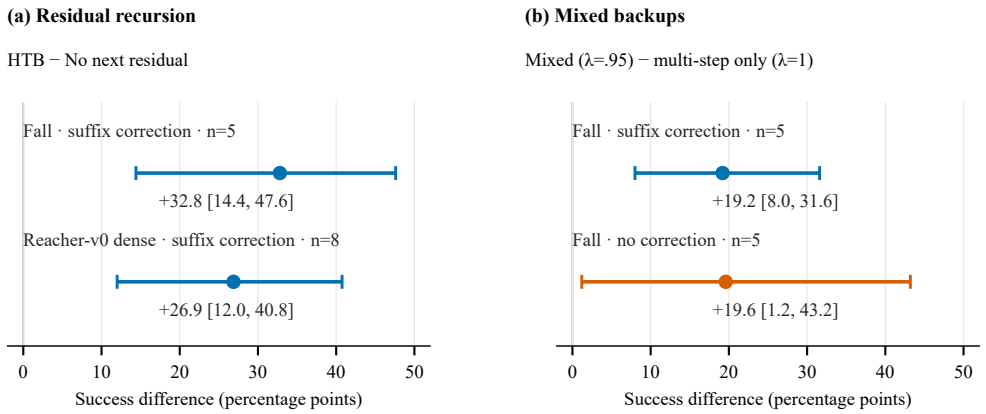


Figure 4: **Residual recursion and mixed backups contribute to task success.** (a) Full HTB minus the variant without the next-step residual, both at $\lambda = .95$. (b) Mixed backups at .95 minus multi-step-only learning at 1. Points are paired mean success differences; bars are nominal 95% bootstrap intervals.

The intervention changes both objective weights and therefore measures their combined effect. Complete controls, including tasks without improvement, appear in Tables 10 and 9.

The nonnegative residual provides a separate, one-sided constraint. In the nonrecursive ablation, imposing nonnegative outputs and targets increases Reacher-v1 dense success with relabeling at $\lambda = .6$ by 38.4 points [15.0, 63.9]. The complete task grid shows both positive and negative effects (Tables 14–16). The factorial and finite-MDP diagnostics in Appendix F clarify this constraint's role: lower estimates can reduce optimism while increasing underestimation and total estimation error.

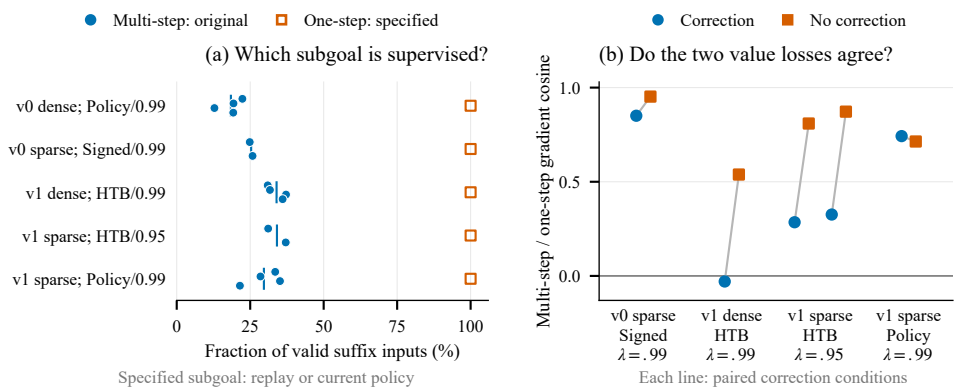


Figure 5: **Direct subgoal supervision and loss compatibility are distinct.** (a) Circles measure retention of the original subgoal after multi-step correction; hollow squares mark the designated one-step input rule. (b) Each line connects correction/no-correction measurements from one paired repeat, grouped by task, variant and λ. Each point is the multi-step–one-step gradient cosine within that model (Equation (13)), averaged over two minibatches. *Signed* and *Policy* are the nonrecursive signed-residual and current-policy-subgoal variants.

4.3 Why separate subgoal supervision from behavior correction?

Figure 5 tests two properties of the combined update: which subgoal receives a target, and whether the two losses make compatible parameter updates. We analyze 512 replay transitions per trained policy instance, across 28 instances.

(a) Relabeling redirects supervision away from the original subgoal. Among the 16 correction-trained instances, only 27.9% of valid multi-step inputs retain the subgoal recorded in replay. The other targets supervise different subgoals. HTB's one-step loss instead uses its explicitly designated subgoal; the hollow squares show this input rule, not measured optimizer counts. Thus the one-step branch supplies direct supervision that likelihood correction alone does not preserve.

(b) Preserving the subgoal does not ensure compatible updates. For 10 policy instances, we compare the unweighted multi-step and one-step losses $\mathcal{L}_L, \mathcal{L}_S$ on the same replay minibatch. Both gradients use the same online parameters θ_V, covering the twin values and all remaining-time heads:

$$c_{L,S} = \frac{\langle \nabla_{\theta_V} \mathcal{L}_L, \nabla_{\theta_V} \mathcal{L}_S \rangle}{\|\nabla_{\theta_V} \mathcal{L}_L\|_2 \|\nabla_{\theta_V} \mathcal{L}_S\|_2}. \tag{13}$$

Positive values indicate aligned descent directions; zero indicates orthogonality and negative values indicate conflict. For Reacher-v1 sparse HTB at $\lambda = .95$, the mean cosine is 0.306 with correction and 0.841 without it. The losses therefore align more strongly in the no-correction condition for this setting. Panel (a) motivates explicit subgoal supervision; panel (b) shows why its compatibility with the multi-step loss is a separate design question. Task-success effects are tested by the controlled training comparisons.

The performance ablations show that the effect of relabeling depends on the backup mixture. On Reacher-v1 sparse at $\lambda = .95$, disabling relabeling raises HTB success from 32.38% to 96.13%, a paired improvement of 63.75 points [49.38, 74.25]. Across the interior Reacher grid, 60 of 64 configurations have higher mean success without relabeling. In Push/Fall, the benefit of correction reverses as λ changes (Figures 9–10). This interaction supports HTB's separation of multi-step behavior correction from one-step supervision at the specified subgoal: their weights and correction settings control distinct learning effects.

Table 2: **Shared values improve estimates for actions without direct labels.** MAE against complete rollout returns (mean ± SD across 3 policy seeds; lower is better). Left: all three action types on trajectories excluded from fitting and calibration. Right: uniformly sampled actions at training inputs (s, G, g, t), whose return labels enter no loss.

	Separate test trajectories		Actions excluded from fitting	
Task	Direct-Q	Shared $V - A$	Direct-Q	Shared $V - A$
Reacher-v0 dense	70.60 ± 48.07	60.22 ± 36.09	50.45 ± 35.64	19.29 ± 18.79
Reacher-v1 sparse	9.49 ± 0.36	8.86 ± 0.91	8.50 ± 0.35	1.13 ± 0.03

4.4 Can shared values estimate actions without direct supervision?

We test whether fitting a few actions at a state–subgoal input improves predictions for another action whose return was excluded from fitting. The high- and low-level policies are frozen. From states sampled along their trajectories, we execute a candidate primitive action for one step, then follow the fixed hierarchy until the episode ends. The accumulated task reward gives a reference for $Q_t(s, G, g, a)$, with the training reward scale and boundary discount and no critic bootstrap.

We compare a direct-Q critic with a shared $V - A$ critic. The latter uses an unconstrained residual to isolate value sharing. Parameter counts are nearly equal; batches and supervision are identical. At each (s, G, g, t), the policy action and a Gaussian-perturbed action supply training returns. Both critics also receive the same return from following the policy, supervising V or direct-Q at the policy action. A third, uniformly sampled action supplies a test return that never enters a loss. Table 2 reports two tests: all three action types on separate evaluation trajectories, and the excluded actions at the training inputs.

After 5000 updates, sharing lowers mean absolute error (MAE) in both tests for all 6 policy instances. For actions excluded from fitting, dense-task MAE decreases from 50.45 to 19.29, and sparse-task MAE from 8.50 to 1.13. This supports the intended benefit of the shared value: learning from other actions improves estimates beyond the directly supervised choices.

The test-trajectory result persists after exact-input deduplication. Subgoal-selection regret varies across tasks, separating this estimation benefit from decision quality. Appendix F provides the full protocols, learning curves and sensitivity analyses.

5 Related work

HIRO corrects recorded behavior (Nachum et al., 2018), while HAC uses path-independent goal-reaching rewards under idealized lower-level execution (Levy et al., 2019). Other relabeling methods address preferences and demonstrations (Singh et al., 2024; Singh & Namboodiri, 2025). Intra-option learning reuses primitive experience across extended actions (Sutton et al., 1998; 1999); goal-conditioned values, hindsight goals and subgoal generation provide complementary mechanisms (Schaul et al., 2015; Andrychowicz et al., 2017; Zhang et al., 2020; Kim et al., 2021). Time-aware values and timed subgoals motivate temporal conditioning (Pardo et al., 2018; Gürtler et al., 2021); HTB uses the next high-level decision as its continuation boundary. Dueling networks and normalized advantages share values across actions (Wang et al., 2016; Gu et al., 2016); HTB adds a separately supervised action-value residual. Twin critics and smoothing follow TD3 (Fujimoto et al., 2018); the loss mixture differs from Retrace's importance-weighted return operator (Munos et al., 2016).

6 Discussion and conclusion

HTB combines specified-subgoal evaluation, time-aware backups and shared action values while retaining task rewards. Its conditional identities require Markov states and action coverage; joint actor–critic convergence remains open. Sparse Maze and MazeW motivate better update-goal selection. The task and estimation results support its central capability: replay correction need not determine which subgoal receives value supervision.

AI USE STATEMENT

We used generative AI tools for readability editing of the manuscript. We have not used generative AI tools for proposing or refining the hypotheses, designing or critiquing the methodology or experiments, implementing the methods, interpreting the results, formulating the mathematical claims, or writing the proofs; generating synthetic datasets and qualitative data analysis are not applicable to this work. We have reviewed all AI-assisted work: every number in the paper was read back from the released derived data files by an author, the figure-generation code was checked against those files, and every theorem statement and proof was written and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work is a methodological study of reinforcement learning algorithms. It involves no human subjects, personal data, or real-world deployment. We believe that no additional ethical issues arise beyond the general question of how empirical claims in this field are supported, which this paper addresses directly.

REPRODUCIBILITY STATEMENT

Appendix A gives the assumptions and proofs of the conditional Bellman results. Appendix B specifies the backup and termination conventions. The empirical appendices record sample counts, measurement protocols, paired comparisons, and analysis limitations. Appendix F specifies the frozen-policy and finite-MDP diagnostics. Validation scripts regenerate the statistics from retained seed-level results.

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 *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5049–5059. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, pp. 1726–1734, 2017. doi: 10.1609/aaai.v31i1.10916. URL https://ojs.aaai.org/index.php/AAAI/article/view/10916.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2829–2838. PMLR, 2016. URL https://proceedings.mlr.press/v48/gu16.html.

Nico Gürtler, Dieter Büchler, and Georg Martius. Hierarchical reinforcement learning with timed subgoals. In *Advances in Neural Information Processing Systems*, volume 34, pp. 21732–21743. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/b59c21a078fde074a6750e91ed19fb21-Abstract.html.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26091–26104. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a766f56d2da42cae20b5652970ec04ef-Abstract-Conference.html.

Junsu Kim, Younggyo Seo, and Jinwoo Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 28336–28349. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/ee39e503b6bedf0c98c388b7e8589aca-Abstract.html.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

Rémi Munos, Thomas Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 29, pp. 1054–1062. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c3992e9a68c5ae12bd18488bc579b30d-Abstract.html.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, pp. 3303–3313. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1emus0qF7.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4045–4054. PMLR, 2018. URL https://proceedings.mlr.press/v80/pardo18a.html.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34866–34891. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/6d7c4a0727e089ed6cdd3151cbe8d8ba-Abstract-Conference.html.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320. PMLR, 2015. URL https://proceedings.mlr.press/v37/schaul15.html.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

Utsav Singh and Vinay Purushothaman Namboodiri. PEAR: Primitive enabled adaptive relabeling for boosting hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a42fa12a321b3594317b1a60a6dbccae-Abstract-Conference.html.

Utsav Singh, Wesley A. Suttle, Brian M. Sadler, Vinay P. Namboodiri, and Amrit Singh Bedi. PIPER: Primitive-informed preference-based hierarchical reinforcement learning via hindsight relabeling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 45681–45695. PMLR, 2024. URL https://proceedings.mlr.press/v235/singh24e.html.

Richard S. Sutton, Doina Precup, and Satinder Singh. Intra-option learning about temporally abstract actions. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 556–564, 1998. URL https://icml.cc/Conferences/1998/papers/paper162.html.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3540–3549. PMLR, 2017. URL https://proceedings.mlr.press/v70/vezhnevets17a.html.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1995–2003. PMLR, 2016. URL https://proceedings.mlr.press/v48/wangf16.html.

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21579–21590. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f5f3b8d720f34ebebceb7765e447268b-Abstract.html.

A Remaining-time Bellman construction and its scope

A.1 Fixed-policy value definition

Fix a stationary low-level policy μ and high-level policy π_H, a high-level decision interval H, a task goal G, and $0 \le \gamma < 1$. The environment is Markov in s; if termination depends on a clock, that clock is included in the state for the statement below. The joint law of next state, scaled task reward, and terminal indicator is $P(ds', dr, dd \mid s, a, G)$. This law does not depend on the high-level query g. Assume bounded rewards. Relative subgoals are transported by $h(s, g, s') = g + \phi(s) - \phi(s')$.

Starting from (s, G, g, t), the low-level policy acts under the transported subgoal until termination or the scheduled boundary after t primitive steps. At a nonterminal boundary, the high-level policy chooses a new subgoal using π_H and resets the countdown to H. If $b(k)$ counts boundaries crossed before primitive step k, define

$$V_t^{\mu,\pi}(s, G, g) = \mathbb{E}\left[\sum_{k=0}^{T-1} \gamma^{b(k)} r_k \;\middle|\; s_0 = s, G, g_0 = g, t_0 = t\right], \tag{14}$$

where T is the terminal time, with an infinite sum for continuing tasks. The bound $H\|r\|_\infty/(1-\gamma)$ ensures existence. This definition evaluates the specified low-level policy, including during learning. Define $Q_t^{\mu,\pi}(s, G, g, a)$ by fixing the first primitive action to a and following the same hierarchy afterward. Superscripts are omitted when the policies are understood.

Proposition 2 (Remaining-time Bellman identity)**.** *The values in Equation* (14) *satisfy Equations* (2)–(3)*. Unrolling from a countdown t reaches the next high-level decision boundary with an undiscounted sum of within-segment rewards and a single factor γ on the continuation. On a trajectory that terminates earlier, the continuation is zero.*

Proof. Condition on the first action and the transition (s', r, d). If $d = 1$, the return is r. If $d = 0$ and $t > 1$, no high-level decision boundary has been crossed, so the remaining return is $V_{t-1}(s', G, h(s, g, s'))$ without a discount factor. If $d = 0$ and $t = 1$, a boundary is crossed and the remaining return is $\gamma V_H(s', G, \pi_H(s', G))$. Taking the conditional expectation gives Equation (2); integrating the first action under μ gives Equation (3). Repeat the same argument until the first boundary or terminal transition. All intermediate continuations have multiplier one. Only a nonterminal boundary contributes γ, proving the unrolled form. □

The identity gives the implementation's boundary-discounted objective: within-segment rewards have unit weights, and each subsequent high-level segment contributes another factor γ. The long replay target in Equation (4) has this unrolled form. A replay suffix follows its historical low-level policy, so equality to the current-policy expectation requires on-policy sampling or a valid off-policy estimator. Likelihood-based candidate selection provides an approximate correction.

A.2 Remaining time distinguishes subgoal values

Proposition 3 (Remaining-time aliasing)**.** *Suppose the retained input $x = (s, G, g, a)$ does not distinguish remaining times t and u. If $Q_t(x) \neq Q_u(x)$, every predictor $q(x)$ satisfies* $\max\{|q(x) - Q_t(x)|, |q(x) - Q_u(x)|\} \ge |Q_t(x) - Q_u(x)|/2$.

Proof of Proposition 3. For the same retained input x, the triangle inequality gives $|Q_t(x) - Q_u(x)| \le |Q_t(x) - q(x)| + |q(x) - Q_u(x)|$. The larger of the two errors is at least half their required total. The midpoint $q(x) = (Q_t(x) + Q_u(x))/2$ attains the bound for this pair. □

A stationary Markov example realizes a nonzero gap. Let the physical environment have one self-loop state s, two primitive actions a_1, a_0, and rewards $r(s, a_1, G) = 1$, $r(s, a_0, G) = 0$ for fixed G. There is no termination and $\phi(s) = 0$, so relative-goal transport leaves each goal unchanged. Use goals $g_1 = 1$ and $g_0 = 0$, with deterministic low-level policy $\mu(s, g_1) = a_1$ and $\mu(s, g_0) = a_0$. The stationary boundary high-level policy always chooses g_0. For any $H \ge 2$ and $0 \le \gamma < 1$, the boundary value under g_0 is zero: $V_H(s, G, g_0) = \gamma V_H(s, G, g_0)$.

Start a queried goal g_1 with remaining time t and first action a_1. It yields exactly t unit rewards before the high-level policy switches to g_0; all subsequent rewards vanish. Hence

$$V_t(s, G, g_1) = Q_t(s, G, g_1, a_1) = t, \qquad Q_1(s, G, g_1, a_1) = 1, \quad Q_2(s, G, g_1, a_1) = 2. \tag{15}$$

The physical state, task goal, queried subgoal, and first action are identical, yet the values differ. Their distinction is the time at which control returns to the high-level policy.

The representational requirement is that remaining time be recoverable from the value input. An elapsed clock combined with a known schedule can also satisfy it. The lower bound applies when two distinct phases are aliased; it establishes the need to distinguish their boundary semantics, independently of the neural encoding used to do so.

A.3 Primitive-action conditioning permits off-policy primitive-step updates

For any bounded fixed continuation family F, define

$$b_t^F(s, G, g, a, s', r, d) = r + (1 - d)C_t^F(s', G, h(s, g, s')), \tag{16}$$

$$(\mathcal{B}_t F)(s, G, g, a) = \int b_t^F(s, G, g, a, s', r, d)\, P(ds', dr, dd \mid s, a, G), \tag{17}$$

where C_t^F replaces V by F in Equation (3), with the high-level policy held fixed. The continuation may depend on the query even though the physical transition law does not.

Proof of Proposition 1. Let β be any data-collection low-level policy, including a low-level policy from an earlier training iteration. The sampling and query-selection assumptions preserve the outcome law after conditioning on selection and on the query. Under the Markov assumption, conditioning on (s, a, G) removes the dependence of (s', r, d) on β and on the collection goal. Its conditional law is $P(\cdot \mid s, a, G)$. Substituting this law into the conditional expectation of b_t^F gives $\mathcal{B}_t F$. This equality holds for every fixed query g, including a query different from the collection goal. It does not change the distribution of the sampled action a. To integrate under a new low-level policy μ, the conditional function must be identified on actions to which μ assigns probability. That step needs coverage or a justified generalization assumption. □

The invariant quantity is the conditional one-step sampling law; the continuation value tracks the policies being evaluated. Replay-goal and frozen high-level-policy proposal queries use pre-outcome information and satisfy the query-selection condition. Goals selected by likelihood correction from a realized suffix can depend on its outcomes, so their multi-step estimator uses the separate approximate correction described above. Evaluating a current low-level policy additionally requires coverage of its actions or a justified generalization assumption, as in the proof.

A.4 Shared value updates and action-dependent estimation errors

For exact fixed-policy values, put $A_t^{\mu,\pi} = V_t^{\mu,\pi} - Q_t^{\mu,\pi}$. For one estimated value head $V_{t,j}$ and residual network A_t, define $e_V = V_{t,j} - V_t^{\mu,\pi}$, $e_A = A_t - A_t^{\mu,\pi}$, and $e_Q = (V_{t,j} - A_t) - Q_t^{\mu,\pi}$, all at the same appropriate inputs. Then

$$e_Q(s, G, g, a) = e_V(s, G, g) - e_A(s, G, g, a), \quad |e_Q(s, G, g, a)| \le |e_V(s, G, g)| + |e_A(s, G, g, a)|. \tag{18}$$

Substitution of $Q_t^{\mu,\pi} = V_t^{\mu,\pi} - A_t^{\mu,\pi}$ proves both statements. Holding A_t fixed, a change in $V_{t,j}$ produces the same additive change in every reconstructed action value at (s, G, g, t), including actions without direct supervision, as stated in Equation (7). When both networks change, $\Delta\widehat{Q}_{t,j}(a) = \Delta V_{t,j} - \Delta A_t(a)$. Here the shared bias is an action-independent *value offset*, not a particular neural-layer bias parameter.

The decomposition exposes two contributions to action-value accuracy: a shared value error and an action-specific residual error. Updating V transfers its common component across actions, while accuracy at actions absent from the data still depends on generalization of A. Thus improved V alone supplies no uniform action-value or high-level ranking guarantee. A direct neural action-value network may also change unsampled predictions; the distinction here is the explicit shared component. The implementation assigns separate parameters to V and A and detaches every target.

The residual-regression loss therefore has zero gradient with respect to online V; the long/short value losses train that component.

A.5 A SIGNED RESIDUAL REPRESENTATION

Let $q_t(s, G, g)$ be a fixed reference value and define

$$A_t^F(s, G, g, a) = q_t(s, G, g) - (\mathcal{B}_t F)(s, G, g, a). \tag{19}$$

Then the following reconstruction holds for any query and low-level policy:

$$q_t(s, G, g) - \mathbb{E}_{a\sim\mu(\cdot|s,g)} A_t^F(s, G, g, a) = \mathbb{E}_{a\sim\mu(\cdot|s,g)} (\mathcal{B}_t F)(s, G, g, a). \tag{20}$$

For a deterministic low-level policy the action expectation is a single evaluation at $\mu(s, g)$. Equation (20) follows by substitution, so it holds independently of how the query was selected. In a population least-squares regression with sufficient coverage and a realizable function class, the target $q_t - b_t^F$ identifies A_t^F. Together with Proposition 1, this gives an explicit route from replay transitions to the queried low-level policy return.

The identification statement above applies to realizable population least-squares regression. Practical HTB uses smooth-L_1 regression, finite neural networks, target networks, and a nonnegative residual except in the signed-residual ablation. Nonnegativity is a stabilizing constraint on this estimator: the exact $V_t - Q_t$ can be negative when an individual action outperforms the low-level policy's value. The signed-residual ablation retains the signed target and output to test this constraint.

For a deterministic low-level policy with exact policy values, the signed residual at its own action satisfies $A_t^{\mu,\pi}(s, G, g, \mu(s, g)) = 0$. Consequently, the extra next-state signed-residual term used to motivate HTB leaves the exact fixed-policy Bellman identity unchanged. Away from this limit, HTB changes the fitted recursion by replacing the next reference critic with its residual-corrected value. We assess that estimator choice empirically.

Proposition 4 (Action-conditioned residual recursion)**.** *Fix a deterministic low-level policy μ, a bounded boundary value $B(s, G)$, and bounded reference functions $q_t(s, G, g)$ for $t = 1, \ldots, H$. Define $Q_t^{\mu,B}$ as the conditional return that executes its first action a, then follows μ until the next scheduled boundary or termination, and receives continuation γB at a nonterminal boundary. There is a unique family of exact signed residuals, denoted here by $A_t = q_t - Q_t^{\mu,B}$, satisfying*

$$A_1(s, G, g, a) = q_1(s, G, g) - \mathbb{E}[r + (1 - d)\gamma B(s', G) \mid s, a, G], \tag{21}$$

$$\begin{aligned} A_t(s, G, g, a) = q_t(s, G, g) - \mathbb{E}[r + (1 - d)\,\{&q_{t-1}(s', G, g') \\ &- A_{t-1}(s', G, g', \mu(s', g'))\} \mid s, a, G\,], \qquad t > 1, \end{aligned} \tag{22}$$

where $g' = h(s, g, s')$. The reconstructed value $q_t(s, G, g) - A_t(s, G, g, \mu(s, g))$ equals the value of the specified subgoal under μ and boundary B. This holds for every query on which the conditional expectations are identified.

Proof. For $t = 1$, the definition of $Q_1^{\mu,B}$ gives the first equation and determines A_1 uniquely. Suppose the statement holds at $t - 1$. Then

$$q_{t-1}(s', G, g') - A_{t-1}(s', G, g', \mu(s', g')) = Q_{t-1}^{\mu,B}(s', G, g', \mu(s', g')). \tag{23}$$

The right side is the continuation return of the low-level policy after the first transition. Substitution into Equation (22) gives $A_t = q_t - Q_t^{\mu,B}$ and uniquely determines A_t from A_{t-1}. Induction proves the family and its reconstruction at the current low-level policy's action. Proposition 1 applies to each conditional expectation under its coverage assumptions. □

The proposition characterizes the signed conditional recursion. Practical HTB instantiates it with frozen target references and a target-high-level decision boundary value during each update. Its projected targets and delayed residual estimates give an approximate, constrained version of that recursion. The no-next-residual and signed-residual variants use q_{t-1} directly as their next-step continuation, omitting the residual correction inside that continuation.

A.6 NONNEGATIVE PROJECTION GIVES A ONE-SIDED CORRECTION

The nonnegative output guarantees $\widehat{Q}_{t,j} = V_{t,j} - A_t \le V_{t,j}$ pointwise, regardless of the quality of the learned reference. A stronger scalar identity holds when the projected residual target is fitted exactly.

Proposition 5 (One-sided residual correction)**.** *For real values q, b, let $p(q,b) = q - [q-b]_+$. Then*

$$p(q,b) = \min(q,b), \qquad p(q,b) \le q, \qquad |p(q,b) - p(q,b')| \le |b - b'|. \tag{24}$$

If an estimated residual A approximates $[q-b]_+$ to error at most ϵ, then $|q - A - p(q,b)| \le \epsilon$.

Proof. If $q \ge b$, subtraction of $q - b$ gives b; if $q < b$, the clipped residual is zero and the result is q. The minimum with a fixed scalar is nonexpansive in its other argument. Finally, $|q - A - p(q,b)| = |A - [q-b]_+| \le \epsilon$. □

The correction is nonexpansive with its reference held fixed. This local bound leaves the coupled actor–critic iteration, which also updates the reference, outside its scope. Projection before taking expectations differs from projecting a conditional mean: for a random backup B, $q - \mathbb{E}[(q-B)_+] = \mathbb{E}[\min(q,B)]$, which can be strictly below $\min(q, \mathbb{E}[B])$. The practical target therefore has a conservative component whose cost and benefit depend on the task and on λ.

Scope for maximization amplification. If noisy candidate maximization raises a scalar backup b above q, the projected reconstruction removes that upward increment. The remaining error depends on the reference: $q = 3$, $b = 4$, and true value 1 still give the overestimate $\min(q,b) = 3$, while a low reference can suppress a legitimate improvement. The implementation's high-level policy maximizes $V_{t,1}$, and its low-level policy uses the intrinsic-reward critic for policy optimization. The local cap therefore characterizes the correction, not the global bias of either policy-optimization process.

A.7 INTERMEDIATE ERROR SOURCES IN PRIMITIVE AND SUFFIX BACKUPS

Consider a fixed low-level policy up to the next scheduled boundary and a fixed boundary continuation B. Let $\widehat{B}$ satisfy $\|\widehat{B} - B\|_\infty \le \epsilon_B$. Suppose an approximate primitive recursion introduces an additive operator residual η_j at remaining time j, with $\|\eta_j\|_\infty \le \epsilon_j$. For $j > 1$ the correct within-segment expectation operator is nonexpansive in its continuation; at $j = 1$ the boundary continuation has coefficient γ and a terminal mask. Consequently,

$$\|\widehat{V}_t - V_t\|_\infty \le \sum_{j=1}^{t} \epsilon_j + \gamma \epsilon_B. \tag{25}$$

The base error is at most $\epsilon_1 + \gamma\epsilon_B$. Each earlier step adds at most ϵ_j because within-segment discounting is one. Induction proves Equation (25).

For a suffix sampled under that same low-level policy, the observed reward sum contains none of these learned intermediate continuations. Replacing only B by $\widehat{B}$ in its direct target gives

$$\left| \mathbb{E}[\widehat{Y}^{\mathrm{L}} - Y^{\mathrm{L}} \mid s, G, g, t] \right| \le \gamma \epsilon_B. \tag{26}$$

The reward terms cancel pathwise, leaving the masked boundary error. Under the stated operator-error bounds and common sampling policy, Equations (25)–(26) isolate the dependence on learned intermediate continuations. Actual estimator errors additionally depend on reward variance, fitting error, and any mismatch between historical and current trajectory distributions; the $V - A$ architecture alone supplies no bound on these terms. HTB's truncated relabeling adapts historical suffixes approximately, while its mixture retains the primitive route for an explicitly specified subgoal.

A.8 DIRECT VALUE UPDATES FOR SPECIFIED SUBGOALS

Proposition 6 (Query routing)**.** *Fix a replay suffix, detached query g, detached short target $\widehat{y}^{\mathrm{S}}$, and $\lambda < 1$. For each critic, the short term in Equation* (12) *supplies the output derivative*

$$(1-\lambda)\, \ell_1'\big(V_{t,j}(s,G,g), \widehat{y}^{\mathrm{S}}\big) \tag{27}$$

Algorithm 1 Learning from a high-level replay segment

1: Sample a segment τ of observed length $L \leq H$.
2: **for** each observed step $i = 0, \ldots, L-1$ **do**
3: Set $t = H - i$ and choose query g_i (replay or current high-level policy).
4: Set $\widetilde{g}_i$ by suffix correction, or keep the replay goal.
5: Form detached $y^A_{t,i}$, $\widehat{y}^{\mathrm{S}}_{t,i}$ and $y^{\mathrm{L}}_{t,i}$;
using masked returns and excluding unobserved padding.
6: **end for**
7: Fit A at recorded actions using y^A.
8: Fit twin critics with $\lambda\mathcal{L}_{\mathrm{L}} + (1-\lambda)\mathcal{L}_{\mathrm{S}}$.
9: Improve the time-indexed actors; Polyak-update target networks.

at the queried input (s, G, g), independently of the long-path goal selected by correction. If its residual is nonzero, this derivative is nonzero for the smooth-L_1 loss.

Proof. Differentiate the short term while holding its query and target fixed. The chain rule gives Equation (27). The corrected goal is an argument only of the long term and cannot replace the short term's input. The smooth-L_1 derivative vanishes only at zero residual. □

The proposition identifies which input receives a direct loss derivative. The resulting parameter update combines both losses, whose gradients can cancel, and parameter sharing can change predictions at other inputs. A high-level policy proposal can serve as g because the implementation detaches it before forming the residual and critic losses. At $\lambda = 1$, the short derivative is zero, so this supervision mechanism is absent.

B Implementation of the remaining-time updates

B.1 Networks, losses, and update order

The Markov-state construction is implemented with each benchmark's observation protocol. Push/Fall observations contain ant position and velocity together with elapsed time and omit the movable object's state. Because hidden object configurations can depend on collection history, the resulting observation-based update is an approximation to Proposition 1; its conditional-law guarantee requires a sufficient Markov state. All compared Push/Fall conditions use the same observation protocol.

For $H = 10$, each horizon has its own actor, twin values $V_{t,j}$, and residual network, with width-300 ReLU hidden layers. Their inputs are (s, G), (s, G, g), and (s, G, g, a), respectively. Parameters are stored in a batched horizon axis, but no trainable weights are shared across horizons. The residual's final activation is ReLU except in the signed-residual variant, which uses a linear output. The low-level policy uses the same observation preprocessing in candidate correction and residual queries as it uses to select actions.

For each sampled high-level segment, we construct all observed suffixes. Training proceeds in the following order:

1. Select each suffix's long-path goal, and compute target high-level policy boundary values using the minimum of the two target critics.
2. Choose the short-path query, transport it by one primitive step, and form the one-step residual target. Apply the terminal mask after selecting the boundary or within-segment continuation.
3. Update the online residual network at the recorded primitive action using smooth-L_1 loss.
4. Compute the long and short critic losses. The short target uses the *target* residual network, including in the iteration immediately after the online residual update. Minimize their λ-weighted combination.
5. Update each high-level actor to maximize its first online critic and then Polyak-update the target actors, critics, and residual networks.

Within a horizon, we average loss over valid samples; we sum these averages over active horizons. The rollout calls only the H-step actor. Actors for shorter remaining durations are trained at replay suffix states and provide query goals for the current-policy-subgoal variant; they do not resample the high-level policy's goal during a segment.

B.2 Exact distinctions between variants

All variants share the long target in Equation (4) and short target in Equation (11). For all nonrecursive variants, replace the $t > 1$ branch in Equation (8) by $\bar{V}_{t-1}(s_{i+1}, G, g_i^+)$. For the signed-residual variant, remove the positive-part operation from Equation (9) and use an unconstrained residual output. For the current-policy-subgoal variant, set

$$g_i = \mathrm{stopgrad}(\pi_t(s_i, G)), \qquad g_i^+ = g_i + \phi(s_i) - \phi(s_{i+1}). \tag{28}$$

The same detached query enters the residual and short critic paths. These losses do not create an extra actor objective through the query. In all replay-query variants, $g_i = g_0^{\mathrm{orig}} + \phi(s_0) - \phi(s_i)$.

The full HTB next-state residual is evaluated at the current low-level policy's deterministic action with no smoothing noise. The short critic target evaluates the target residual at the current low-level policy's action plus clipped Gaussian smoothing noise, with the resulting action clipped to its bounds. Boundary targets use the target high-level policy with its own clipped smoothing noise and action bounds. Neither expression uses a target low-level actor.

When $\lambda = 1$, the short critic term is skipped. The residual network continues training but does not affect actor or critic targets. A seeded implementation test verifies equality of actor and critic updates with the independent remaining-time baseline. This is an implementation regression test, not a claim that separately collected HIRO and countdown runs are identical.

B.3 Suffix correction and early termination

For each valid suffix starting at i, the candidate set contains its transported original subgoal, the observed displacement $\phi(s_L) - \phi(s_i)$, and K Gaussian candidates centered on that displacement. The Push/Fall protocol uses $K = 8$, for ten total candidates; the Reacher/Maze implementation defaults to $K = 10$, for twelve total candidates. Random candidates are clipped to the subgoal bounds. The selection rule is

$$\widetilde{g}_i = \arg\min_{g \in \mathcal{C}_i} \sum_{k=i}^{L-1} \left\| a_k - \mu(s_k, g + \phi(s_i) - \phi(s_k)) \right\|_2^2 . \tag{29}$$

Each horizon independently samples and scores its candidates on its observed suffix. Disabling correction returns the transported replay subgoal for the long path; it leaves the short-path construction unchanged. Since candidate selection consumes random numbers, paired seeds across correction settings do not imply identical realized replay or exploration trajectories.

The countdown is scheduled time to a high-level policy decision, not observed time to termination. If a segment terminates after $L < H$ primitive actions, only branches $t = H, H-1, \ldots, H-L+1$ are valid. The last observed transition has $d_i = 1$, so its residual target contains only r_i after the reference value. All suffix long targets end at that same terminal state and have zero continuation. Unobserved padding is excluded from losses and from candidate likelihoods. If termination coincides with $t = 1$, the terminal mask overrides boundary bootstrapping. These conventions preserve the semantics of a high-level policy that planned an H-step segment before termination.

C Training and evaluation protocol

The Push/Fall experiments use a shared environment protocol and PyTorch implementation. All compared methods use the same environment semantics and optimizer/update conventions. This comparison does not imply an exact reproduction of the original HIRO training software. Table 3 reports the common training settings.

Training uses Adam and smooth-L_1 critic losses. Updates are executed in episode batches: the low-level policy receives one gradient iteration per collected primitive step, and the high-level policy

Table 3: **Push/Fall training settings.** Values are shared by HIRO and the remaining-time variants unless the component exists only in the latter.

Setting	Value
Environment steps	10,000,000
High-level period / update interval	10 / 10 primitive steps
Batch size (both levels)	128
Replay capacity (both levels)	200,000 transitions each
Actor learning rate (both levels)	10^{-4}
Critic learning rate (both levels)	10^{-3}
Action-residual learning rate	10^{-3}
High-level boundary / low-level discount	0.99 / 0.95
High-level / low-level reward scale	0.1 / 1.0
Polyak coefficient	0.005
Target action noise standard deviation / clip	0.2 / 0.5
Training exploration standard deviation	1.0
Random / total correction candidates	8 / 10
Evaluation interval / episodes	50,000 steps / 50
Episode duration	500 primitive steps
Warmup episodes	0

receives one per high-level policy update interval. The actor and target networks update at every corresponding gradient iteration. The high-level policy critic and action-residual optimizers use weight decay 10^{-4}.

Evaluation calls the high-level policy and low-level policy without exploration noise and without parameter updates. Push/Fall episodes always run for the fixed duration: reaching the task goal does not end evaluation early. Success is measured on the final step, and return sums the environment rewards over the episode. The environment uses dense distance reward, relative subgoals, and no stochastic position perturbation.

The Reacher/Maze evaluation instead stops an episode at its first successful step and reports success and steps to termination. The comparisons use one million environment steps and eight paired seeds. We do not pool the Reacher/Maze success measure with Push/Fall terminal success. The Reacher tail statistic averages the retained evaluation points within the reported terminal training window before averaging over seeds.

D Empirical protocol and complete paired results

Metrics and analysis units. Push and Fall use the curve entries `eval0/avg_ep_rew` and `eval0/perc_env_goal_achieved` at exactly 10 million environment steps. Each configuration is evaluated with 5 seeds (777–781) at this evaluation point. The comparison includes 74 logical configurations including the two HIRO references, and 36 correction/no-correction pairs. The method variants have identical seed-level returns and success at $\lambda = 1$; removing the repeated method labels leaves 32 distinct comparison rows. The figure shows this endpoint once per environment, and its uncertainty calculation counts each distinct seed outcome once.

For Reacher, the metric is `eval0/perc_env_goal_achieved`, averaged over the last twenty evaluation entries ending at exactly one million steps. Each eligible comparison contains eight paired seeds (777–784). The primary interior-λ set contains 64 configurations and 512 configuration–seed pairs. It excludes the endpoints because $\lambda = 1$ removes short-path critic supervision and $\lambda = 0$ removes long-path critic supervision. The latter may still execute correction and consume random numbers, so endpoint differences need not measure a pure change in the weighted critic objective. The mechanism analysis focuses on the interior grid.

Uncertainty in figures. We report variation across seeds and configuration choices explicitly, in line with the evaluation concerns discussed by Agarwal et al. (2021). The plotted intervals are percentile intervals with nominal coverage 95%. For each Push/Fall configuration, we enumerate all 3125 with-replacement resamples of its five paired seed differences and take the 2.5th and 97.5th

Table 4: **Reacher conclusions persist across analysis sets.** Gains are mean success-rate differences in percentage points; wins count positive configuration means.

Analysis set	Settings	Wins	Mean gain
Interior mixing weights	64	60	+26.82
Full grid, deduplicated	84	68	+20.88
Full logical grid	88	71	+20.49
Single-setting exclusion, logical grid	87	71	+20.74
Single-setting exclusion, deduplicated	83	68	+21.14

percentiles using linear interpolation. Enumeration removes Monte Carlo variation from bootstrap computation; finite-seed coverage remains nominal. For the Reacher correction-effect strata in Figure 10, a fixed generator seed resamples configurations and then paired seed differences within each chosen configuration for ten thousand replicates. Configuration means receive equal weight. These intervals describe this finite grid and have no simultaneous multiple-test coverage. The full sweep characterizes variation across settings.

Analysis-set sensitivity. The full grid contains 88 logical comparisons. A single-setting exclusion analysis removes Reacher-v1 sparse current-policy-subgoal/$\lambda = 0$, leaving 87 comparisons. This setting has eight paired seeds and is included in the full-grid analysis. Separately, HTB and its no-next-residual variant at $\lambda = 1$ have identical paired seed values in every reward–environment stratum, so collapsing those repeated endpoint rows gives 84 configurations. Table 4 compares the full, single-setting-exclusion, deduplicated, and interior analysis sets. The correction-effect aggregate uses the interior grid, which excludes both endpoints.

Reference setting and benchmark interpretation. Table 12 gives the same fixed HTB/$\lambda = .95$ setting in both environments and correction modes. These are comparisons against this repository's HIRO implementation under the local paper-environment protocol specified in Appendix C. Both sides of each Push/Fall comparison use the same five seed identifiers; additional HIRO training seeds remain outside these paired analyses.

An additional within-Fall interaction for the no-next-residual variant is the difference between the no-correction effects at $\lambda = .6$ and .9. Its empirical bootstrap uses the same resampled seed indices in all four arms, enumerating the five-seed resamples. This exploratory contrast has a nominal interval without simultaneous multiple-test coverage.

Routing and gradient diagnostics. Subgoal-retention diagnostics evaluate each fitted policy instance separately. Each analysis samples four batches of 128 high-level transitions, without replacement inside each batch; batches can overlap. Valid suffix queries use the implementation's residual subgoal at their own start state. The retention indicator is $\mathbb{1}[\|g^{\mathrm{orig}} - \widetilde{g}\|_2 \le 10^{-6}]$. We first compute retention within each trained model, then average over the correction-enabled models. Applying correction to no-correction models is a counterfactual diagnostic; their actual long-path update retains the original goal. The diagnostic measures subgoal retention on reconstructed valid suffixes; value generalization is evaluated separately in Appendix F.

For gradient diagnostics, the unweighted multi-step loss $\mathcal{L}_L$ and one-step loss $\mathcal{L}_S$ use the same sampled replay minibatch and valid suffix mask. Their Huber losses sum the two online value critics, with the implementation's common remaining-time reduction. Targets are detached, and each loss is differentiated separately with respect to the same θ_V: all parameters of both online value critics across the ten independent remaining-time heads. Neither the residual network nor either actor is part of this gradient vector; target-network parameters are excluded as well. The long target uses the recorded suffix reward and target-critic boundary bootstrap; the short target uses the current target critic and residual network. Target-high-level policy smoothing is sampled according to the diagnostic's fixed random seed. The offline short-path query uses the current low-level policy's unsmoothed action, while training adds action smoothing. The cosines therefore describe this reconstructed-loss diagnostic, not the exact stochastic training gradients. Equation (13) concatenates these parameters before computing a cosine. Positive scalar mixture weights do not change that cosine. Each point compares the two objectives *within one fitted model*; connecting correction and no-correction points

Table 5: **Value changes at fixed subgoal inputs.** Mean absolute critic change across earlier and later models. R0/R1 denote Reacher-v0/v1; d/s denote dense/sparse. Values use each run's reward scale.

Setting	Seed	Correction	Steps (millions)	Original	Corrected	Current
Rv0/s/Signed	781	yes	0.95–1.00	0.316	0.315	0.322
Rv0/s/Signed	784	yes	0.70–0.75	0.333	0.327	0.336
Rv0/s/Signed	781	no	0.90–1.00	0.214	0.205	0.199
Rv0/s/Signed	784	no	0.90–1.00	0.950	0.945	0.989
Rv1/d/HTB	783	yes	0.95–1.00	3.366	3.324	3.394
Rv1/s/HTB	779	yes	0.95–1.00	2.667	0.789	1.080
Rv1/s/HTB	781	yes	0.95–1.00	1.133	0.976	0.988
Rv1/s/Policy	781	yes	0.95–1.00	0.558	0.558	0.548
Rv1/d/HTB	783	no	0.90–1.00	10.097	10.041	9.999
Rv1/s/HTB	779	no	0.90–1.00	0.309	0.308	0.321
Rv1/s/HTB	781	no	0.90–1.00	0.520	0.519	0.539
Rv1/s/Policy	781	no	0.90–1.00	0.525	0.521	0.536

compares their alignments, not gradients from different models. The horizon-specific diagnostic slices the independent horizon heads and counts negative cosines over correlated minibatch–head combinations. Each plotted value averages two minibatches. Figure 5b contains 5 pairs matched by environment, reward, method, mixture weight, seed, and training budget. Pair eligibility does not depend on cosine or success. Gradient diagnostics cover twelve policy instances, including two outside this matched display; the other sixteen instances provide subgoal-retention measurements only.

Exploratory association and fixed-query drift. Across the same 5 equal-budget pairs, the Spearman correlation between correction retention and the no-correction success advantage is -0.800. This exploratory association describes the selected settings and does not identify a causal effect. We also hold the later model's probe states and three goal sets fixed and evaluate earlier and later critics on those inputs. Table 5 reports absolute changes. The statistic measures aggregate prediction drift from all learning between the two training stages. It neither isolates low-level-policy changes nor measures value-estimation or Bellman error.

Table 6 lists every logical pair in the comparison grid, including the endpoint controls and comparisons favoring correction.

Current-policy queries and additional task coverage. Table 7 shows two current-policy-subgoal/$\lambda = .99$ configurations, where the short branch queries the current high-level policy's goal. These selected illustrations characterize the variant at two settings; its aggregate effect uses the full grid. The full Reacher aggregate includes all current-policy-subgoal interior weights. Under the same endpoint/window rule, the fixed HTB/$\lambda = .95$ no-correction setting on AntMaze-v1 achieves 73.75% dense and 19.38% sparse success, each over eight seeds. The Maze experiments compare algorithm variants without correction; they do not constitute paired correction ablations. The complete Maze/MazeW grid appears in the eight-task comparison, including the MazeW settings with low success. Specified-subgoal supervision addresses value updates; selecting useful subgoals and exploring remain complementary requirements.

Horizon-loss scales. The loss-scale analysis covers 1377 distinct run identities with at least 99,000 high-level updates. It averages losses within each run over early, middle, and late thirds of that run's update range, then aggregates across seeds. The count measures the number of analyzed runs; repeated endpoint configurations share the same underlying seed outcomes. Figure 6 shows the late-phase signed-residual/$\lambda = .6$ profiles: correction-enabled runs have much larger long- and short-path critic losses in these settings. The number of seeds differs between conditions. These profiles describe loss scale, while the cosine diagnostic measures update direction and the rollout-return comparison measures critic accuracy. Scalar loss magnitudes alone provide neither of those measurements nor a causal account of task performance.

Table 6: **Complete Push/Fall correction ablation.** Differences are no correction minus correction. Success differences are in percentage points. Entries average the paired seed differences. Repeated $\lambda = 1$ rows preserve the full grid and share the same seed outcomes.

Env.	Rule	λ	Δ return	Δ success
Fall	No next residual	0	-971.5	+0.0
Fall	No next residual	0.6	+5938.8	+0.0
Fall	No next residual	0.9	-767.3	-9.2
Fall	No next residual	0.95	+2850.9	+14.4
Fall	No next residual	0.99	+4225.6	+13.2
Fall	No next residual	1	-309.2	-8.8
Fall	HTB	0	+699.5	+0.0
Fall	HTB	0.6	+2522.4	+0.0
Fall	HTB	0.9	-2089.1	-25.2
Fall	HTB	0.95	-315.0	-8.4
Fall	HTB	0.99	+21.0	-20.8
Fall	HTB	1	-309.2	-8.8
Fall	Signed residual	0	+1037.8	+0.0
Fall	Signed residual	0.6	+5481.4	+4.4
Fall	Signed residual	0.9	-894.8	-5.2
Fall	Signed residual	0.95	+683.3	-10.0
Fall	Signed residual	0.99	+1586.2	-2.0
Fall	Signed residual	1	-309.2	-8.8
Push	No next residual	0	-1057.7	-0.8
Push	No next residual	0.6	+3458.8	+0.0
Push	No next residual	0.9	-115.0	-9.6
Push	No next residual	0.95	-1155.4	-16.8
Push	No next residual	0.99	-144.9	+0.0
Push	No next residual	1	+355.6	-13.2
Push	HTB	0	-113.6	+0.0
Push	HTB	0.6	+4349.2	+0.0
Push	HTB	0.9	-65.6	+5.2
Push	HTB	0.95	-432.7	-13.2
Push	HTB	0.99	+2.8	+6.4
Push	HTB	1	+355.6	-13.2
Push	Signed residual	0	+381.6	+0.0
Push	Signed residual	0.6	+3378.9	+0.0
Push	Signed residual	0.9	+1489.8	-1.2
Push	Signed residual	0.95	+401.3	-2.8
Push	Signed residual	0.99	-126.6	-16.8
Push	Signed residual	1	+355.6	-13.2

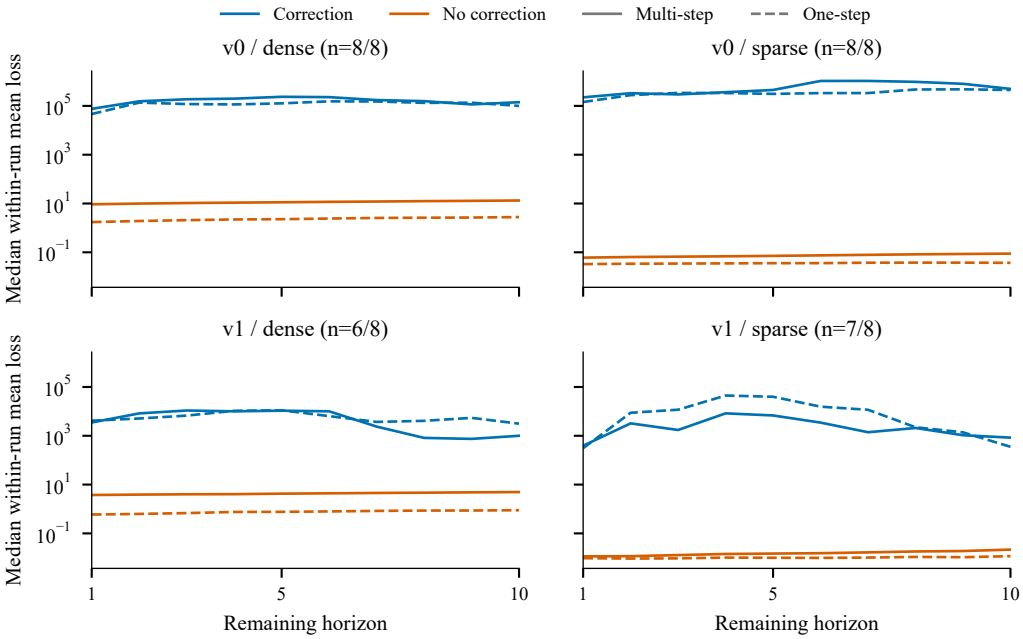


Figure 6: **Late-phase signed-residual/.6 loss scales differ between correction modes.** Each curve is the median across seed-level phase-mean losses; the vertical axis is logarithmic. Panel counts give the number of correction/no-correction seeds. The late phase is the final third of each run's high-level updates. The profiles measure scalar loss magnitude; gradient alignment is evaluated in Figure 5.

Table 7: **Current-policy goal queries can also benefit from disabling correction.** Eight-seed mean final-window success percentages for two current-policy-subgoal/.99 examples. These are illustrative configurations within the full grid.

Task	Correction	No correction	Difference (pp)
Reacher-v0 / dense	60.00	88.13	+28.12
Reacher-v1 / sparse	26.00	81.63	+55.62

Table 8: **Fixed-configuration success across eight goal-reaching tasks.** Success (%), mean $\pm$ seed SD over eight seeds, with correction disabled. Multi-step only is the remaining-time baseline at $\lambda = 1$; HTB uses $\lambda = .95$. HIRO uses the local high-level-policy implementation. Dashes mark conditions without an eight-seed HIRO comparison, not zero success.

Environment	Reward	HIRO	Multi-step only	HTB/.95
Reacher-v0	dense	94.3 ± 2.2	92.6 ± 4.0	95.4 ± 3.2
Reacher-v0	sparse	32.1 ± 32.1	54.3 ± 37.8	48.6 ± 41.8
Reacher-v1	dense	96.5 ± 2.2	97.1 ± 1.7	97.3 ± 2.3
Reacher-v1	sparse	76.0 ± 30.3	95.4 ± 7.9	96.1 ± 3.4
Maze-v1	dense	—	60.6 ± 18.8	73.8 ± 9.5
Maze-v1	sparse	—	35.0 ± 39.9	19.4 ± 31.8
MazeW-v2	dense	—	26.9 ± 6.3	29.3 ± 6.5
MazeW-v2	sparse	—	14.8 ± 15.1	10.4 ± 11.4

E Component controls, complete task coverage, and learning curves

Task coverage and comparison units. The Maze comparison contains 768 runs: two environments, two rewards, four methods, six mixture weights, and eight seeds. Every curve reaches exactly one million environment steps and uses the same success metric. Table 8 uses all eight seeds in each fixed setting. The HIRO-without-correction controls comprise 32 Reacher runs with the same seed identifiers and tail-window metric. They use the distinct HIRO high-level policy implementation. The remaining-time $\lambda = 1$ endpoint is reported as a separate baseline. Matched budgets and seeds support a comparison of complete algorithms, including their implementation differences. Maze HIRO results are not included because they do not meet the eight-seed comparison criterion.

Component averages and figure uncertainty. Table 9 reports mean paired differences, giving equal weight to each configuration and each seed. For Figure 7, we first average paired differences across the fixed evaluated configurations within each seed, then resample the seed columns jointly. This preserves the dependence created by reused seed identifiers and controls. A second interval jointly resamples configuration rows and seed columns using 20,000 draws with generator seed 20260923. This interval describes sensitivity to the evaluated grid. Every configuration has equal weight. The interior grid is $\lambda \in \{.6, .9, .95, .99\}$; the short-branch contrast uses only the fixed .95 setting against 1. Repeated $\lambda = 1$ method aliases are excluded from component aggregates.

What the component controls isolate. The No next residual versus Signed residual contrast changes the nonnegative residual output and projected residual target together. It measures their joint effect on task performance, with both favorable and unfavorable cases included. Signed value-estimation error is measured separately by the rollout-return diagnostic. HTB versus No next residual changes the next-step residual in the implemented recursion. Current-policy subgoal versus No next residual changes the query from replay to the current high-level policy. All retain the same value/action-value residual factorization. The .95–1 contrast keeps the remaining-time architecture but changes both objective weights. The original grid therefore tests recursion, query selection, and the backup mixture within a common factorized architecture. Appendix F separately tests direct-Q versus shared-$V - A$ learning and the output-constraint-by-target-projection factorial comparison under fixed policies. Tables 14, 15, and 16 give every interior projection contrast; Tables 17, 18, and 19 give the recursion contrasts. Tables 20 and 21 give the complete query-distribution controls.

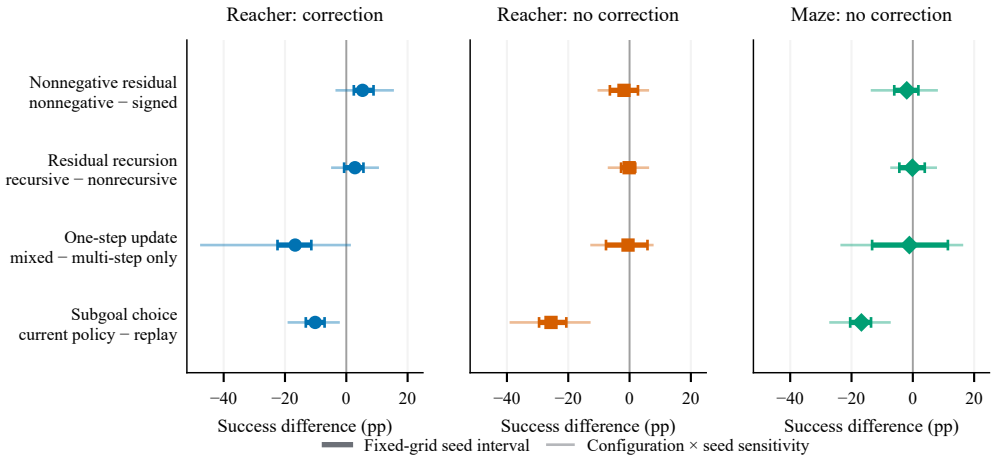


Figure 7: **Component effects depend on the task and query distribution.** Positive differences favor the left-hand condition. Thick bars resample eight shared seed blocks on the fixed grid; thin bars also resample configurations. Residual constraints, recursion, and query contrasts average all sixteen task–weight settings per panel; the short-path contrast averages four tasks. Nonnegative and query contrasts use the nonrecursive base. Intervals are nominal 95%.

Fixed short-branch control. Table 10 reports every task and mode at the fixed default. Fall mean success increases under both modes. Reacher effects vary across tasks, and some correction-enabled tasks have lower success with mixed backups. Eligibility is independent of the effect direction, so the complete table locates both the benefit on Fall and the task dependence of the mixture.

Learning curves and integrated return. The fixed-setting success curves in Figure 3 and return curves in Figure 8 show 200 equally spaced evaluations from 50,000 to ten million environment steps for each of the thirty runs. The same five seeds are used for each arm. Bands enumerate the empirical bootstrap of seed curves at each time point, giving pointwise coverage; simultaneous coverage is not assessed. All evaluation points are shown without a moving average. Table 11 compares the time-normalized integral over $[50{,}000, 10{,}000{,}000]$ with the final return. This distinguishes sample-efficiency behavior from the endpoint comparison.

Parameter sharing across remaining-time horizons. Table 13 compares 180 runs of the multi-step-only baseline. Independent horizon actors/critics, a shared time-conditioned critic, and shared time-conditioned actor/critic networks all retain suffix expansion and correction. The ten-seed comparison changes parameter sharing across horizons, whereas the $V - A$ comparison tests sharing across actions at a fixed remaining time. The eight overlapping independent-network Reacher seed values are the same observations as the Multi-step only reference. All curves reach one million environment steps.

Full correction sweeps. Figure 9 displays every Push/Fall correction contrast and Figure 10 gives Reacher strata. Both effect directions are represented throughout. The complete endpoint table and Reacher endpoint sensitivities are retained in Appendix D. For full HTB on Push, the paired change in the no-correction-minus-correction return effect between $\lambda = .6$ and .9 is 4414.8 [2517.6, 6176.2]. Each interaction pairs the same five seeds across all four conditions. The corresponding Fall interaction is 4611.5 [1681.1, 7276.2].

Table 9: **Interior-grid component averages.** Differences are left minus right. Reacher/Maze use success percentage points; Push/Fall use return. Correction is on/off. K counts configurations. Each entry is the mean paired difference across the evaluated configurations. No family or metric is pooled with another.

Contrast	Tasks	Corr.	K	Mean difference
Recursive − nonrecursive	Maze	off	16	-0.1
Recursive − nonrecursive	Push/Fall	off	8	-151.3
Recursive − nonrecursive	Push/Fall	on	8	1136.0
Recursive − nonrecursive	Reacher	off	16	-0.1
Recursive − nonrecursive	Reacher	on	16	2.8
Nonnegative − signed	Maze	off	16	-2.0
Nonnegative − signed	Push/Fall	off	8	77.1
Nonnegative − signed	Push/Fall	on	8	-209.4
Nonnegative − signed	Reacher	off	16	-1.8
Nonnegative − signed	Reacher	on	16	5.3
Policy − replay	Maze	off	16	-16.8
Policy − replay	Reacher	off	16	-25.6
Policy − replay	Reacher	on	16	-10.2
Combined − long-only	Maze	off	4	-1.1
Combined − long-only	Push/Fall	off	2	187.0
Combined − long-only	Push/Fall	on	2	584.0
Combined − long-only	Reacher	off	4	-0.5
Combined − long-only	Reacher	on	4	-16.7

Table 10: **The short branch has task-dependent effects.** Fixed HTB/$\lambda = .95$ minus Multi-step only at $\lambda = 1$. Success differences use percentage points; returns use the environment's unscaled reward. All entries pair the same seeds within a task and mode.

Environment	Reward	Correction	Metric	Mean difference
Maze-v1	dense	off	Success	13.1
Maze-v1	sparse	off	Success	-15.6
MazeW-v2	dense	off	Success	2.4
MazeW-v2	sparse	off	Success	-4.4
Fall	dense	off	Return	344.7
Fall	dense	off	Success	19.6
Fall	dense	on	Return	350.5
Fall	dense	on	Success	19.2
Push	dense	off	Return	29.3
Push	dense	off	Success	0.8
Push	dense	on	Return	817.6
Push	dense	on	Success	0.8
Reacher-v0	dense	off	Success	2.8
Reacher-v0	dense	on	Success	-2.9
Reacher-v0	sparse	off	Success	-5.6
Reacher-v0	sparse	on	Success	-2.9
Reacher-v1	dense	off	Success	0.1
Reacher-v1	dense	on	Success	-0.6
Reacher-v1	sparse	off	Success	0.8
Reacher-v1	sparse	on	Success	-60.4

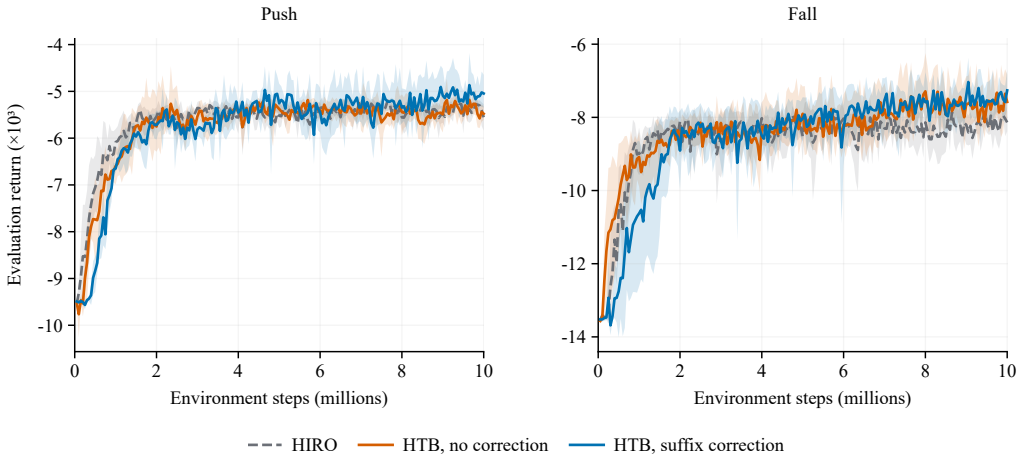


Figure 8: **Final return and early learning can favor different settings.** Actual returns at all 200 evaluations from 0.05M to 10M steps. Both HTB conditions use $\lambda = .95$. Lines average five matched seeds; bands are pointwise 95% empirical bootstrap intervals. No temporal smoothing is applied.

Table 11: **Integrated return and final return at the fixed configuration.** HTB/.95 minus HIRO, five paired seeds. AUC is the trapezoidal return integral divided by the duration of its observed step window. Each cell gives the mean paired difference.

Task	HTB correction	Normalized AUC difference	Final-return difference
Push	on	-84.1	508.3
Push	off	-82.2	75.5
Fall	on	69.4	875.4
Fall	off	226.6	560.4

Table 12: **Fixed HTB settings against the Push/Fall HIRO reference.** Five-seed means at ten million steps; both HTB modes use $\lambda = .95$. The HIRO rows use paper-style correction.

Environment	Setting	Return	Success (%)
Push	HIRO	-5553.6	0.0
Push	HTB, suffix correction	-5045.3	14.0
Push	HTB, no correction	-5478.1	0.8
Fall	HIRO	-8137.8	9.6
Fall	HTB, suffix correction	-7262.4	32.8
Fall	HTB, no correction	-7577.4	24.4

Table 13: **Horizon-sharing effects vary across goal-reaching tasks.** Multi-step only algorithms with correction, final-window success (%), mean ± SD over ten seeds. The architecture intervention differs from separating V and A within an action-conditioned value.

Task	Reward	Independent	Shared critic	Shared actor/critic
Reacher-v0	dense	87.7±4.5	81.6±8.2	86.7±5.5
Reacher-v0	sparse	12.7±10.6	24.3±27.3	9.8±11.2
Maze-v1	dense	33.0±22.3	17.0±21.6	39.0±27.6
Maze-v1	sparse	27.5±32.6	9.0±22.3	7.5±22.0
MazeW-v2	dense	21.1±7.2	14.8±7.0	22.6±6.7
MazeW-v2	sparse	6.8±9.2	7.3±7.7	9.5±10.6

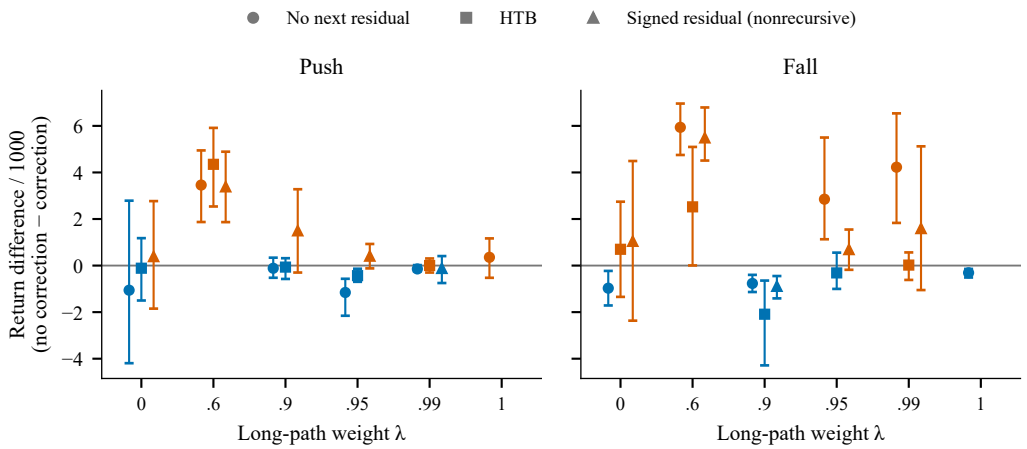


Figure 9: **The benefit of correction reverses with the mixture.** Positive final-return differences favor disabling correction. Points average 5 paired seeds; bars enumerate 3125 empirical bootstrap resamples. Symbols identify methods; identical $\lambda = 1$ results are shown once per environment.

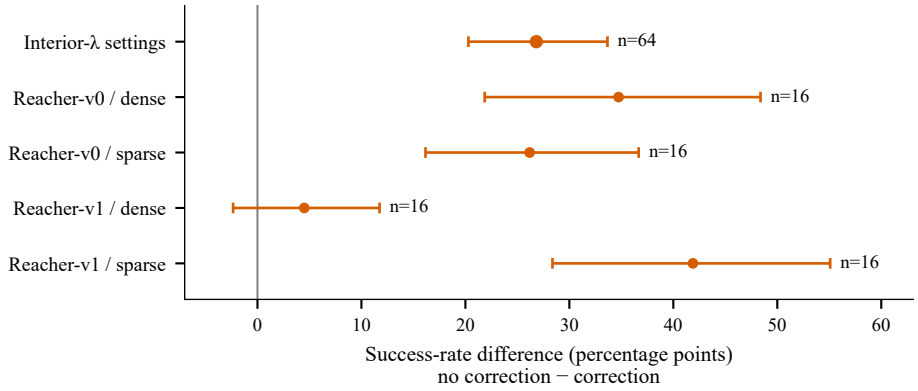


Figure 10: **The correction effect depends on the task.** Mean final-window success differences on the interior-λ grid. Intervals resample configurations and their 8 paired seeds; n counts configurations. Each configuration has equal weight. This correction-effect bootstrap differs from the shared-seed-block component sensitivity defined in this appendix.

Table 14: **Complete Reacher nonnegative-residual contrasts.** No next residual minus Signed residual; success difference in percentage points, eight paired seeds. Both conditions use the nonrecursive base. Correction is on/off. All interior weights and both effect directions appear.

Task	Reward	Corr.	λ	Mean difference
Reacher-v0	dense	off	0.6	0.1
Reacher-v0	dense	off	0.9	-2.5
Reacher-v0	dense	off	0.95	2.9
Reacher-v0	dense	off	0.99	-0.9
Reacher-v0	dense	on	0.6	18.5
Reacher-v0	dense	on	0.9	34.5
Reacher-v0	dense	on	0.95	-11.1
Reacher-v0	dense	on	0.99	-8.8
Reacher-v0	sparse	off	0.6	-4.0
Reacher-v0	sparse	off	0.9	11.3
Reacher-v0	sparse	off	0.95	6.6
Reacher-v0	sparse	off	0.99	-17.5
Reacher-v0	sparse	on	0.6	0.1
Reacher-v0	sparse	on	0.9	-1.5
Reacher-v0	sparse	on	0.95	6.1
Reacher-v0	sparse	on	0.99	-1.6
Reacher-v1	dense	off	0.6	0.0
Reacher-v1	dense	off	0.9	0.0
Reacher-v1	dense	off	0.95	0.0
Reacher-v1	dense	off	0.99	-1.0
Reacher-v1	dense	on	0.6	38.4
Reacher-v1	dense	on	0.9	14.1
Reacher-v1	dense	on	0.95	8.0
Reacher-v1	dense	on	0.99	-1.9
Reacher-v1	sparse	off	0.6	-25.3
Reacher-v1	sparse	off	0.9	-1.5
Reacher-v1	sparse	off	0.95	7.3
Reacher-v1	sparse	off	0.99	-4.6
Reacher-v1	sparse	on	0.6	3.3
Reacher-v1	sparse	on	0.9	-4.6
Reacher-v1	sparse	on	0.95	6.4
Reacher-v1	sparse	on	0.99	-15.0

Table 15: **Complete Maze nonnegative-residual contrasts.** No next residual minus Signed residual on the nonrecursive base; success difference in percentage points, eight paired seeds.

Task	Reward	Corr.	λ	Mean difference
Maze-v1	dense	off	0.6	-63.1
Maze-v1	dense	off	0.9	21.3
Maze-v1	dense	off	0.95	13.8
Maze-v1	dense	off	0.99	8.1
Maze-v1	sparse	off	0.6	-8.1
Maze-v1	sparse	off	0.9	-0.6
Maze-v1	sparse	off	0.95	5.0
Maze-v1	sparse	off	0.99	8.8
MazeW-v2	dense	off	0.6	-4.3
MazeW-v2	dense	off	0.9	0.8
MazeW-v2	dense	off	0.95	2.4
MazeW-v2	dense	off	0.99	-6.4
MazeW-v2	sparse	off	0.6	-0.6
MazeW-v2	sparse	off	0.9	0.4
MazeW-v2	sparse	off	0.95	-7.0
MazeW-v2	sparse	off	0.99	-1.9

Table 16: **Complete Push/Fall nonnegative-residual contrasts.** No next residual minus Signed residual on the nonrecursive base; final-return difference, five paired seeds.

Task	Reward	Corr.	λ	Mean difference
Fall	dense	off	0.6	112.2
Fall	dense	off	0.9	174.9
Fall	dense	off	0.95	276.3
Fall	dense	off	0.99	421.1
Fall	dense	on	0.6	-345.2
Fall	dense	on	0.9	47.4
Fall	dense	on	0.95	-1891.3
Fall	dense	on	0.99	-2218.2
Push	dense	off	0.6	243.9
Push	dense	off	0.9	438.7
Push	dense	off	0.95	-905.7
Push	dense	off	0.99	-144.9
Push	dense	on	0.6	164.0
Push	dense	on	0.9	2043.5
Push	dense	on	0.95	650.9
Push	dense	on	0.99	-126.5

Table 17: **Complete Reacher residual-recursion contrasts.** HTB minus No next residual; success difference in percentage points, eight paired seeds.

Task	Reward	Corr.	λ	Mean difference
Reacher-v0	dense	off	0.6	2.0
Reacher-v0	dense	off	0.9	0.3
Reacher-v0	dense	off	0.95	0.9
Reacher-v0	dense	off	0.99	2.1
Reacher-v0	dense	on	0.6	-5.0
Reacher-v0	dense	on	0.9	10.1
Reacher-v0	dense	on	0.95	26.9
Reacher-v0	dense	on	0.99	16.4
Reacher-v0	sparse	off	0.6	-12.6
Reacher-v0	sparse	off	0.9	12.5
Reacher-v0	sparse	off	0.95	-3.9
Reacher-v0	sparse	off	0.99	-4.4
Reacher-v0	sparse	on	0.6	0.6
Reacher-v0	sparse	on	0.9	1.3
Reacher-v0	sparse	on	0.95	-3.0
Reacher-v0	sparse	on	0.99	1.3
Reacher-v1	dense	off	0.6	0.4
Reacher-v1	dense	off	0.9	-0.6
Reacher-v1	dense	off	0.95	-0.6
Reacher-v1	dense	off	0.99	1.0
Reacher-v1	dense	on	0.6	-0.8
Reacher-v1	dense	on	0.9	-1.3
Reacher-v1	dense	on	0.95	1.9
Reacher-v1	dense	on	0.99	1.5
Reacher-v1	sparse	off	0.6	-13.3
Reacher-v1	sparse	off	0.9	6.8
Reacher-v1	sparse	off	0.95	3.8
Reacher-v1	sparse	off	0.99	3.6
Reacher-v1	sparse	on	0.6	-3.1
Reacher-v1	sparse	on	0.9	-2.0
Reacher-v1	sparse	on	0.95	-20.9
Reacher-v1	sparse	on	0.99	21.3

Table 18: **Complete Maze residual-recursion contrasts.** HTB minus No next residual; success difference in percentage points, eight paired seeds.

Task	Reward	Corr.	λ	Mean difference
Maze-v1	dense	off	0.6	15.6
Maze-v1	dense	off	0.9	-8.8
Maze-v1	dense	off	0.95	4.4
Maze-v1	dense	off	0.99	-2.5
Maze-v1	sparse	off	0.6	0.0
Maze-v1	sparse	off	0.9	5.0
Maze-v1	sparse	off	0.95	-6.3
Maze-v1	sparse	off	0.99	2.5
MazeW-v2	dense	off	0.6	-1.8
MazeW-v2	dense	off	0.9	-1.0
MazeW-v2	dense	off	0.95	1.3
MazeW-v2	dense	off	0.99	-3.1
MazeW-v2	sparse	off	0.6	-12.5
MazeW-v2	sparse	off	0.9	3.1
MazeW-v2	sparse	off	0.95	-3.0
MazeW-v2	sparse	off	0.99	4.6

Table 19: **Complete Push/Fall residual-recursion contrasts.** HTB minus No next residual; final-return difference, five paired seeds.

Task	Reward	Corr.	λ	Mean difference
Fall	dense	off	0.6	-1226.5
Fall	dense	off	0.9	-1053.1
Fall	dense	off	0.95	3.5
Fall	dense	off	0.99	-332.5
Fall	dense	on	0.6	2189.9
Fall	dense	on	0.9	268.7
Fall	dense	on	0.95	3169.4
Fall	dense	on	0.99	3872.1
Push	dense	off	0.6	360.6
Push	dense	off	0.9	15.1
Push	dense	off	0.95	871.8
Push	dense	off	0.99	150.5
Push	dense	on	0.6	-529.9
Push	dense	on	0.9	-34.4
Push	dense	on	0.95	149.2
Push	dense	on	0.99	2.8

Table 20: **Complete Reacher query-distribution contrasts.** Current-policy subgoal minus No next residual on the nonrecursive base; success difference in percentage points, eight paired seeds. Negative values favor replay goals over current high-level policy queries.

Task	Reward	Corr.	λ	Mean difference
Reacher-v0	dense	off	0.6	-68.4
Reacher-v0	dense	off	0.9	-13.1
Reacher-v0	dense	off	0.95	-12.1
Reacher-v0	dense	off	0.99	-4.5
Reacher-v0	dense	on	0.6	7.9
Reacher-v0	dense	on	0.9	-39.6
Reacher-v0	dense	on	0.95	-24.4
Reacher-v0	dense	on	0.99	-11.4
Reacher-v0	sparse	off	0.6	-13.9
Reacher-v0	sparse	off	0.9	-39.3
Reacher-v0	sparse	off	0.95	-43.9
Reacher-v0	sparse	off	0.99	-25.4
Reacher-v0	sparse	on	0.6	-1.9
Reacher-v0	sparse	on	0.9	-0.4
Reacher-v0	sparse	on	0.95	-8.1
Reacher-v0	sparse	on	0.99	-2.6
Reacher-v1	dense	off	0.6	-37.1
Reacher-v1	dense	off	0.9	-1.5
Reacher-v1	dense	off	0.95	-1.0
Reacher-v1	dense	off	0.99	-0.1
Reacher-v1	dense	on	0.6	-6.9
Reacher-v1	dense	on	0.9	-20.0
Reacher-v1	dense	on	0.95	-2.4
Reacher-v1	dense	on	0.99	-0.1
Reacher-v1	sparse	off	0.6	-57.0
Reacher-v1	sparse	off	0.9	-46.9
Reacher-v1	sparse	off	0.95	-46.5
Reacher-v1	sparse	off	0.99	0.6
Reacher-v1	sparse	on	0.6	-5.0
Reacher-v1	sparse	on	0.9	2.3
Reacher-v1	sparse	on	0.95	-21.1
Reacher-v1	sparse	on	0.99	-28.8

Table 21: **Complete Maze query-distribution contrasts.** Current-policy subgoal minus No next residual on the nonrecursive base; success difference in percentage points, eight paired seeds.

Task	Reward	Corr.	λ	Mean difference
Maze-v1	dense	off	0.6	-6.3
Maze-v1	dense	off	0.9	-65.6
Maze-v1	dense	off	0.95	-42.5
Maze-v1	dense	off	0.99	-21.9
Maze-v1	sparse	off	0.6	0.0
Maze-v1	sparse	off	0.9	-18.8
Maze-v1	sparse	off	0.95	-21.3
Maze-v1	sparse	off	0.99	-4.4
MazeW-v2	dense	off	0.6	-19.0
MazeW-v2	dense	off	0.9	-15.0
MazeW-v2	dense	off	0.95	-14.1
MazeW-v2	dense	off	0.99	-4.8
MazeW-v2	sparse	off	0.6	-17.3
MazeW-v2	sparse	off	0.9	-11.6
MazeW-v2	sparse	off	0.95	-9.1
MazeW-v2	sparse	off	0.99	3.1

F SUPPLEMENTARY VALUE-ESTIMATION DIAGNOSTICS

These diagnostics isolate the estimation mechanisms behind the design in Section 4. Matched-supervision critic fitting tests whether a shared value improves generalization across primitive actions. A four-arm comparison separates residual-output and target constraints, and a finite MDP measures their interaction with maximization feedback. The end-to-end task-success experiments remain the performance evaluation.

F.1 FROZEN-POLICY ROLLOUT RETURNS

We use 3 trained policies per task from Reacher-v0 dense and Reacher-v1 sparse, giving 6 policy units. The source dense policies use the signed-residual, nonrecursive variant with relabeling at $\lambda = .9$; the sparse policies use full HTB without relabeling at $\lambda = .95$. All policies were trained for one million environment steps. The high-level and low-level actors remain fixed throughout the diagnostic.

Each unit contains 32 episodes, divided by complete episode into 16 training, 8 calibration, and 8 test episodes. The physical starting state is fixed and task goals are sampled on reset. We retain 960 simulator snapshots per unit at early, middle, and late episode stages, with no overlap of state–task-goal–remaining-time inputs between splits. Each snapshot has 8 candidate subgoal slots: the inherited subgoal, the current H-step high-level proposal, the clipped displacement to the task goal, and uniformly sampled subgoals within the allowed bounds. For each candidate subgoal, 3 primitive actions are evaluated: the deterministic low-level action, a clipped Gaussian perturbation, and a uniformly sampled action. This gives 23040 return labels per unit.

We restore the full simulator and wrapper state before each branch, including episode time and random-generator state. Restoration is checked against repeated action sequences. Each tested action is applied for one primitive step from the restored state, which may lie anywhere along an episode. It is followed by the frozen low-level policy for the remaining subgoal duration; the frozen high-level policy selects subsequent subgoals. Branches run to the original episode termination without a critic bootstrap. Rewards use the training reward scale .1, and discount .99 is applied only at high-level decision boundaries. The environment and evaluation policies are deterministic, so each label is the complete return for its prescribed branch. The generalization distribution consists of new episode snapshots with the same fixed starting state and randomized task goals; new starting states and out-of-distribution tasks are outside this evaluation.

F.2 MATCHED-SUPERVISION REPRESENTATION COMPARISON

The direct estimator has twin Q networks with 3055340 trainable parameters; the factorized estimator has twin V networks and a shared signed residual network with 3057030 parameters. Both retain separate remaining-time branches. They start at the same prediction $Q = 0$; the factorized model uses $V = .1$ and $A = .1$. Both optimize the same action-value labels and a unit-weight policy-value anchor. The anchor trains V in the factorized estimator and $Q(s, G, g, \mu(s, g))$ in the direct estimator, using the same policy-return label. The controlled intervention is the shared-value learning design: identical supervision is applied through estimator-specific action-value and policy-value outputs.

Returns for the policy action and its Gaussian perturbation are available for training; the uniformly sampled action's return is excluded from every training and anchor loss. Training uses the same mini-batch stream, optimizer settings, and 5000 gradient updates. No actor is updated and the supervised-return experiment uses no target network. Calibration episodes are kept separate from the final test split. Table 2 separates two tests. *Separate test trajectories* evaluates all three action types at states sampled from trajectories excluded from fitting and calibration. *Actions excluded from fitting* keeps (s, G, g, t) at the training inputs and tests the uniformly sampled actions whose returns entered no loss. Their exact full (s, G, g, a, t) inputs are also absent from fitting. This second test asks whether supervision at other actions improves the predicted return for the excluded action at the same input. Figure 11 shows the resulting learning curves. Table 22 gives the complete final comparison.

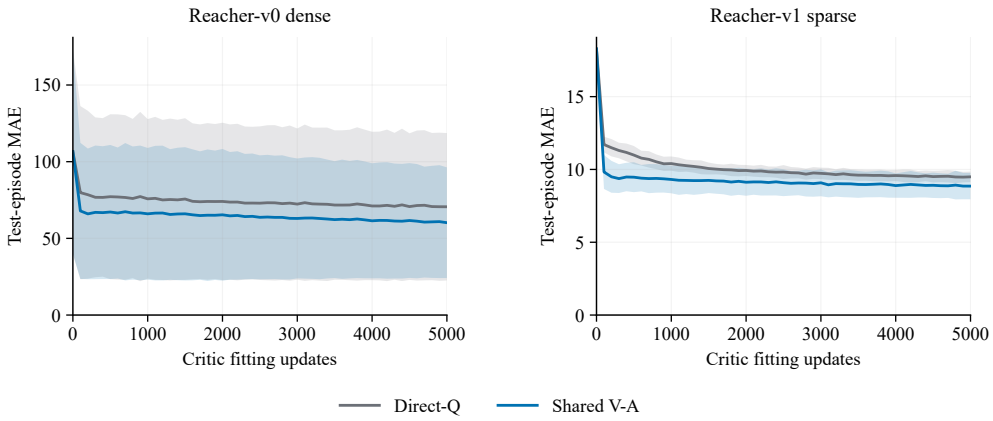


Figure 11: **Value decomposition reduces error under matched supervision.** MAE on separate test trajectories during fixed-policy fitting, averaged over 3 policy seeds per task. Shading is one standard deviation across policy seeds. The horizontal axis counts critic-fitting updates under frozen policies.

Table 22: **Complete neural diagnostic results.** Mean ± SD across 3 policy seeds, using candidate-slot weighting. MAE and signed error use separate test trajectories; Action MAE evaluates their uniformly sampled primitive actions, whose labels are excluded in the representation comparison. Signed error is prediction minus rollout return. Regret is measured among the evaluated subgoal candidates.

Task	Estimator	MAE	Action MAE	Signed error	Regret
Reacher-v0 dense	Direct-Q	70.60±48.07	66.30±38.61	−14.21±30.58	22.87±21.86
Reacher-v0 dense	Shared V-A	60.22±36.09	60.38±36.12	−11.30±17.37	23.54±23.28
Reacher-v0 dense	Output+target	72.75±48.04	73.00±48.09	30.72±12.15	19.67±17.42
Reacher-v0 dense	Output only	73.25±47.82	73.51±47.88	32.75±12.38	20.81±19.25
Reacher-v0 dense	Target only	72.86±48.11	73.10±48.17	30.48±12.30	21.23±19.42
Reacher-v0 dense	Signed residual	73.35±48.08	73.60±48.15	32.17±12.25	21.59±20.04
Reacher-v1 sparse	Direct-Q	9.49±0.36	10.41±0.19	1.77±1.39	1.29±0.20
Reacher-v1 sparse	Shared V-A	8.86±0.91	8.85±0.91	−0.86±1.65	1.33±0.34
Reacher-v1 sparse	Output+target	12.60±0.35	12.60±0.36	8.18±0.38	1.29±0.26
Reacher-v1 sparse	Output only	12.60±0.34	12.60±0.36	8.18±0.39	1.27±0.28
Reacher-v1 sparse	Target only	12.28±0.36	12.27±0.38	7.32±0.45	1.30±0.24
Reacher-v1 sparse	Signed residual	12.52±0.36	12.52±0.37	7.91±0.43	1.27±0.24

F.3 Separating residual output and target constraints

A second experiment crosses nonnegative versus signed output with projected versus signed target. These four estimators use a common nonrecursive residual update, $\lambda = .6$, the same initialization, balanced remaining-time batches, and target-network updates. They learn from the original primitive transitions in the training episodes; the counterfactual rollout labels are reserved for evaluation. The policies remain fixed, no relabeling or target-action noise is used, and the rollout bank supplies evaluation only.

For all neural estimators we evaluate the minimum of the twin reconstructed action values. Subgoal regret compares the rollout value of the critic-selected candidate with the best candidate's rollout value at the same snapshot, using the deterministic policy action. It is distinct from average calibration error. The two representation estimators and the four projection estimators have different supervision protocols and are compared within their respective experiment families.

Table 23: **Exact-input deduplication preserves the MAE comparison.** Shared $V - A$ minus direct-Q test-episode MAE; negative values favor the shared representation.

Task	Policy seed	Slot-weighted difference	Deduplicated difference
Reacher-v0 dense	777	-3.15	-3.13
Reacher-v0 dense	778	-24.92	-25.01
Reacher-v0 dense	779	-3.08	-3.00
Reacher-v1 sparse	777	-0.77	-0.77
Reacher-v1 sparse	778	-0.03	-0.03
Reacher-v1 sparse	779	-1.11	-1.12

The shared representation reduces test-episode MAE in all six policy units; candidate-selection regret varies across tasks. Joint nonnegative output and target constraints slightly improve MAE on the dense task and increase it on the sparse task. These results separate the gain in action-value estimation from the effects of a conservative constraint and the resulting subgoal ranking. In the projection experiment, action-type labels identify evaluation interventions; those actions may also occur in the separately collected transition bank.

F.4 Duplicate-candidate sensitivity

At a high-level boundary, the inherited and current-policy subgoals can coincide. Each bank contains 96 duplicated float32 input groups, leaving 22944 distinct (s, G, g, a, t) inputs from 23040 slots. All duplicate return labels and predictions agree exactly at every evaluation point during fitting. Candidate sets contain seven or eight distinct subgoals; different perturbed actions under the same subgoal are not duplicate inputs.

The primary analysis uses candidate-slot weighting. The sensitivity analysis merges exactly equal float32 input vectors for prediction error and exactly equal candidate subgoals within a snapshot for selection metrics. No merge depends on the outcome. Table 23 shows that the representation comparison retains the same direction in every policy unit. Policy seeds are the outer replication unit, with query rows and snapshots nested within them. The three policy seeds per task define the scope of this comparison.

F.5 Finite-MDP maximization-feedback experiment

We separately test repeated maximization in a stochastic finite MDP with six states, eight primitive actions, and a high-level duration of three steps. Subgoals specify repeated primitive actions. Exact dynamic programming computes the optimal hierarchical values from the transition matrix and reward function, independently of the learned decomposition. The factorial comparison crosses a learned maximizing boundary decision with a fixed optimal boundary policy, and signed with projected residuals. Both decision conditions have the same optimal-value target. A single tabular estimator isolates this mechanism from the twin neural architecture used by full HTB.

The grid uses two, four, or eight candidate subgoals; reward-noise standard deviations of 0, $.3$, or 1; and 30 optimization-noise seeds per condition. Stochastic transitions remain noisy when reward noise is zero. All estimators start at the same exact value to isolate subsequent estimation drift, and use matched samples and update settings. An additional exact-expectation control removes both noise sources. The factorial design contains 1092 conditions.

Table 24 shows a one-sided estimation effect: projection lowers the estimates and increases MAE throughout this grid. Its interaction with maximizing versus fixed-policy backups has mixed directions. The result characterizes the downward constraint on learned values in both backup settings. The exact-expectation controls retain the common fixed point. The comparison evaluates both boundary-policy conditions and their paired interaction, including underestimation, regret, and calibration against the learned policy's true value.

Table 24: **Nonnegative projection trades optimism for underestimation.** Final mean $\pm$ SD across 30 noise seeds with repeated boundary maximization. Selected error is the critic-selected subgoal's prediction minus its exact optimal-policy value; MAE averages all evaluated values. The complete candidate-count and reward-noise grid is shown.

		Selected signed error		MAE	
Candidates	Noise	Signed	Projected	Signed	Projected
2	0.0	0.002 ± 0.017	-0.268 ± 0.019	0.019 ± 0.006	0.270 ± 0.016
2	0.3	0.006 ± 0.028	-0.466 ± 0.032	0.032 ± 0.010	0.464 ± 0.029
2	1.0	0.059 ± 0.071	-1.241 ± 0.083	0.096 ± 0.034	1.245 ± 0.086
4	0.0	0.004 ± 0.015	-0.247 ± 0.017	0.017 ± 0.005	0.246 ± 0.014
4	0.3	0.010 ± 0.022	-0.448 ± 0.027	0.031 ± 0.007	0.446 ± 0.027
4	1.0	0.053 ± 0.074	-1.240 ± 0.091	0.096 ± 0.021	1.239 ± 0.083
8	0.0	0.003 ± 0.014	-0.247 ± 0.015	0.017 ± 0.004	0.248 ± 0.014
8	0.3	0.022 ± 0.027	-0.438 ± 0.029	0.033 ± 0.008	0.437 ± 0.024
8	1.0	0.153 ± 0.069	-1.145 ± 0.073	0.135 ± 0.036	1.160 ± 0.061

G Connections to other hierarchical methods

HTB separates the choice of a high-level command from its value update. This distinction connects its evaluator to methods that learn subgoal representations, discover skills, or plan over latent goals. The connections below follow from their policy interfaces and the conditional Bellman construction in Appendix A; they describe possible algorithmic combinations, not additional performance results.

Learned subgoal representations. Nachum et al. (2019) learn goal representations for a fixed-duration hierarchy built on HIRO. Their objective concerns which representation supports effective hierarchical control; HTB concerns how to evaluate a specified command as its low-level policy changes. The two mechanisms could be combined by conditioning the time-aware critic on the learned subgoal. This requires consistent goal semantics in replay, correct transport of relative goals, and the Markov and action-coverage conditions of the primitive backup. Adjacency constraints (Zhang et al., 2020) and landmark-guided proposals (Kim et al., 2021) likewise act on subgoal selection and can remain separate from the choice of subgoal supervised by the critic.

Unsupervised skills. DIAYN learns behaviorally diverse skills (Eysenbach et al., 2019), while DADS learns skills with predictable outcomes and a skill-dynamics model (Sharma et al., 2020). In a fixed-duration controller over such skills, a skill identifier can replace the spatial subgoal in the time-aware value input and remain constant within its execution segment. The high-level critic would still use accumulated task rewards, independently of the intrinsic objective used to discover the skills. For stochastic skill policies, the one-step value target uses the policy expectation in Equation (10). This provides an evaluation interface when skills are jointly refined with the high-level policy; frozen skills do not themselves create low-level-policy nonstationarity.

Recurrent latent-goal hierarchies. FeUdal Networks learn directional goals and a recurrent worker conditioned on a history of manager goals (Vezhnevets et al., 2017). This motivates value updates for latent commands, but its pooled-goal execution differs from holding one goal for a fixed segment. A time-aware extension must represent the relevant recurrent state and command history and use the actual command-switching rule in its continuation. A single spatial goal and countdown do not encode that execution state.

Offline hierarchical policies. HIQL learns a latent subgoal policy and a low-level policy from an action-free offline value function (Park et al., 2023). Although its high-level training target looks several steps ahead, both policies are queried at every primitive step during execution. Combining this policy interface with temporally extended HTB evaluation would therefore require explicit execution boundaries and action-labeled transitions. Offline action coverage and policy-extraction constraints remain necessary: the primitive conditional identity does not supply information about actions absent from the dataset.

Model-based hierarchical control. Director selects latent goals at fixed intervals and trains hierarchical policies through imagined trajectories in a learned world model (Hafner et al., 2022). Its execution clock offers a natural interface for remaining-time-conditioned goal evaluation. A replay-based extension could attach an HTB-style critic to these goals while accounting for the worker's action distribution and model error. Director's own actor updates use on-policy imagined trajectories; stationarity of real primitive transitions does not remove changes in a learned dynamics model.

Options with learned termination. Option-Critic already learns primitive-action-conditioned intra-option values together with option policies and termination functions (Bacon et al., 2017). Its termination rule determines whether the next step continues the option or invokes the high-level policy. This is the same distinction made explicit by HTB's remaining-time state for scheduled boundaries. Extending the mixed backups to learned termination would replace the deterministic boundary switch by the appropriate termination-weighted continuation, preserving the chosen task reward and discount convention.