HIERARCHICAL CONTRASTIVE CONTROL: LEARNING SUBGOALS VIA FUTURE-STATE DENSITY ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical reinforcement learning (HRL) provides a principled framework for solving long-horizon decision-making tasks by jointly learning a high-level policy that predicts subgoals and a low-level policy that executes primitive actions. However, HRL suffers from training instability because the high-level transition dynamics and rewards change as the low-level policy improves, leading to unstable higher-level temporal-difference learning. We investigate an alternative formulation of hierarchical goal-conditioned RL based on future-state density estimation via contrastive learning, where the high-level policy predicts subgoals that maximize the induced future-state density assigned to the final goal, instead of relying on higher-level temporal-difference bootstrapping. However, directly applying this formulation to hierarchical control is non-trivial, as the high-level policy may predict infeasible subgoals for the current low-level policy, thus hindering final-goal reachability. To address this, we propose Hierarchical Contrastive Control (HCC), a constrained optimization approach that regularizes the future-state density objective with a lower-level optimality constraint that guides the high-level policy toward feasible subgoal predictions for the current low-level policy. We perform experiments on sparse-reward robotic tasks, showing that HCC achieves over 55% success rates, outperforms state-of-the-art baselines and yields feasible subgoals, establishing hierarchical future-state density estimation as an effective learning signal for high-level policy optimization in jointly trained hierarchical control.

1 INTRODUCTION

Hierarchical Reinforcement Learning (HRL) provides a principled framework for solving long-horizon decision-making problems by decomposing them into temporally extended subproblems (Sutton et al., 1999; Dayan & Hinton, 1993; Vezhnevets et al., 2017; Klissarov et al., 2017; Harb et al., 2018). In goal-conditioned hierarchical control, the high-level policy predicts subgoals, while the low-level policy predicts primitive actions to achieve them. However, jointly trained HRL suffers from training instability due to the changing low-level policy, which renders the high-level transition dynamics and rewards non-stationary (Nachum et al., 2018; Levy et al., 2018). As a result, the temporal-difference targets used to train the high-level policy change throughout training, rendering hierarchical policy optimization unstable. Existing approaches to stabilize HRL often rely on computationally expensive off-policy corrections (Andrychowicz et al., 2017) or privileged information such as preferences (Singh et al., 2026) that may be unavailable in complex robotics tasks.

To stabilize high-level learning under non-stationary high-level dynamics and rewards, we propose future-state density estimation (Eysenbach et al., 2021b; 2022) as an alternative to higher-level temporal-difference bootstrapping. Specifically, we optimize the high-level policy by maximizing the future-state density assigned to the final task goal. This provides a direct objective for predicting subgoals that increase the likelihood of reaching the final task goal. However, future-state density alone is insufficient in jointly trained hierarchical control because the high-level policy may predict subgoals that are infeasible for the current low-level policy. Therefore, high-level optimization must predict subgoals that both maximize the future-state density assigned to the final task goal and are feasible for the current low-level policy.

To address this, we propose **Hierarchical Contrastive Control (HCC)**, which optimizes the high-level policy using future-state density while incorporating a lower-level optimality constraint. The

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

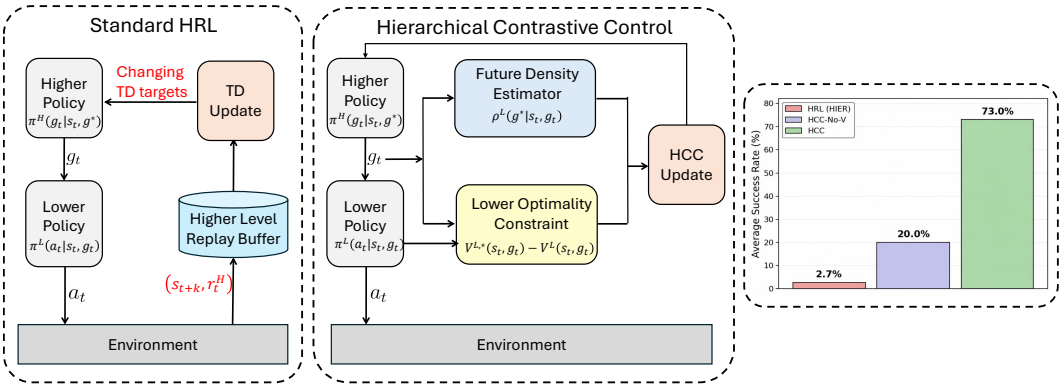


Figure 1: **Overview of Hierarchical Contrastive Control (HCC). Left:** In standard jointly trained HRL, the high-level policy is optimized using temporal-difference (TD) updates. As the low-level policy evolves, the induced high-level transition dynamics and rewards change, causing the TD targets used for high-level learning to change throughout training. **Middle:** HCC replaces higher-level TD bootstrapping with future-state density estimation, which optimizes subgoals according to the density assigned to the final task goal. Because future-state density alone can favor subgoals that are infeasible for the current low-level policy, HCC additionally incorporates a lower-level optimality constraint to guide feasible subgoal prediction. **Right:** Average final success rate across Pick and Place, Hollow, and Franka Kitchen. HCC-No-V substantially outperforms standard HRL, showing the benefit of replacing higher-level TD learning with future-state density optimization, while full HCC further improves performance by incorporating the lower-level optimality constraint.

future-state density term favors subgoals that contribute to reaching the final task goal, while the lower-level optimality term accounts for the ability of the current low-level policy to achieve the predicted subgoal. In practice, HCC estimates future-state density using binary classification and stabilizes this estimate using twin Polyak-averaged target classifiers as the low-level policy evolves. Figure 1 contrasts standard jointly trained HRL, where the evolving low-level policy changes the temporal-difference targets used for high-level learning, with HCC, which replaces higher-level temporal-difference bootstrapping with future-state density optimization and incorporates lower-level optimality to guide feasible subgoal prediction. The figure further shows that future-state density optimization alone improves over standard HRL, while incorporating lower-level optimality yields a substantially larger improvement in average final success rate across the three evaluated environments.

Our main contributions are as follows:

• **Future-state density optimization.** We formulate high-level policy optimization in jointly trained HRL using future-state density estimation, providing an alternative to higher-level temporal-difference bootstrapping under non-stationary high-level dynamics and rewards.

• **Lower-level optimality constraint.** We identify that future-state density alone can favor subgoals that are infeasible for the current low-level policy, and propose HCC, which incorporates a lower-level optimality constraint to guide feasible subgoal prediction. We further stabilize the practical future-state density estimator using twin Polyak-averaged target classifiers.

• **Empirical evaluation.** We evaluate HCC on three sparse-reward robotic tasks, where it consistently outperforms the evaluated hierarchical and non-hierarchical baselines. Our analysis further shows that accounting for lower-level optimality improves both task performance and subgoal feasibility.

2 Related Work

Hierarchical Reinforcement Learning. HRL decomposes long-horizon decision-making into temporally extended subproblems, where a high-level policy predicts subgoals or options that are executed by a low-level policy (Sutton et al., 1999; Barto & Mahadevan, 2003; Parr & Russell, 1998; Dietterich, 1999). Foundational approaches include the options framework (Sutton et al., 1999) and feudal architectures (Dayan & Hinton, 1993; Vezhnevets et al., 2017). Modern methods either learn the hierarchy jointly or employ fixed action primitives to improve temporal abstraction and sample efficiency. In jointly trained HRL, however, improvements in the low-level policy change the effective high-level transition dynamics and rewards, making high-level temporal-difference learning non-stationary (Nachum et al., 2018). HIRO (Nachum et al., 2018) addresses this through hindsight

relabeling of high-level transitions, while methods based on fixed action primitives avoid this source of non-stationarity by keeping the low-level behaviors fixed.

Other approaches reduce this dependence through additional supervision. PEAR (Singh & Namboodiri, 2025) leverages expert demonstrations to stabilize hierarchical learning, while DIPPER (Singh et al., 2026) optimizes the high-level policy using preference supervision over subgoal sequences. In contrast, HCC does not use expert demonstrations or preferences to supervise the high-level policy; the single demonstration used in our experiments is only an auxiliary signal for low-level learning and is applied consistently across all baselines. HCC instead replaces high-level temporal-difference bootstrapping with future-state density estimation under the current low-level policy.

Contrastive Learning for Goal-Conditioned Control. Contrastive learning has been widely used for representation learning and goal-conditioned control (Laskin et al., 2020). C-learning (Eysenbach et al., 2021b) formulates goal-conditioned control as future-state density-ratio estimation, training a classifier to distinguish future states induced by the policy from states sampled from a marginal distribution. Subsequent approaches similarly use contrastive objectives to optimize goal-conditioned policies (Eysenbach et al., 2022). These methods consider the single-level (flat) setting, where the optimized policy directly predicts primitive actions and thereby directly induces the corresponding future-state distribution. In jointly-trained hierarchical control, however, the high-level policy behavior depends on the changing lower level policy. As the low-level policy evolves, its ability to achieve a predicted subgoal also changes throughout training. Consequently, a subgoal that maximizes the future-state density objective may not be feasible for the current low-level policy, making a direct application of C-learning to hierarchical control non-trivial. HCC addresses this hierarchy-specific challenge through a constrained formulation that regularizes the future-state density objective with a lower-level optimality constraint, enabling the high-level policy to account for final-goal reachability while predicting feasible subgoals for the current low-level policy.

3 Problem Formulation

Goal-conditioned hierarchical control. We consider a two-level hierarchical MDP (Dayan & Hinton, 1993). Let $\mathcal{S}$ denote the state space, $\mathcal{A}^L$ the primitive action space, $\mathcal{G} \subseteq \mathcal{S}$ the goal and subgoal space, $\mathcal{P}$ the environment transition function, $r : \mathcal{S} \times \mathcal{A}^L \times \mathcal{G} \to \mathbb{R}$ the environment task reward, and $\gamma \in [0, 1)$ the discount factor. For a fixed low-level policy π^L, the induced high-level MDP is defined as $\mathcal{M}^H = (\mathcal{S}, \mathcal{G}, \mathcal{P}^{H,\pi^L}, r^{H,\pi^L}, \gamma)$, where $\mathcal{P}^{H,\pi^L}$ and r^{H,π^L} denote the high-level transition model and reward induced by π^L, respectively. The lower-level MDP is defined as $\mathcal{M}^L = (\mathcal{S}, \mathcal{A}^L, \mathcal{P}, r^L, \gamma)$, where r^L denotes the intrinsic reward used to train the low-level policy.

Given a final task goal $g^\star \in \mathcal{G}$, the high-level policy π^H predicts a subgoal $g_t \sim \pi^H(\cdot \mid s_t, g^\star)$ every k environment steps, where k is a fixed temporal abstraction interval. Conditioned on g_t, the low-level policy π^L predicts primitive actions $a_i \sim \pi^L(\cdot \mid s_i, g_t)$ for $i = t, \dots, t + k - 1$ to achieve the predicted subgoal. The low-level policy receives the intrinsic reward $r^L(s_i, a_i, g_t) = -\|s_{i+1} - g_t\|_2$, while the high-level policy is optimized using the environment task reward $r(s_i, a_i, g^\star)$ accumulated over the corresponding k-step interval.

Challenge in jointly trained HRL. A central challenge in jointly trained HRL is that the induced high-level transition dynamics and rewards depend on the evolving low-level policy (Nachum et al., 2018). For a predicted subgoal g_t at state s_t, let $\mathcal{P}^{H,\pi^L}(s_{t+k} \mid s_t, g_t)$ denote the distribution over the state reached after k primitive actions under π^L, and let $r^{H,\pi^L}(s_t, g_t; g^\star)$ denote the corresponding accumulated task reward. As π^L improves during joint training, both $\mathcal{P}^{H,\pi^L}$ and r^{H,π^L} change, so the same state–subgoal pair can induce different high-level transitions and rewards at different stages of training. Consequently, the temporal-difference targets used to train the high-level policy also change as the low-level policy evolves, which can destabilize high-level learning.

Limitation of directly applying contrastive future density estimation in HRL. The above limitation raises the question of whether the high-level policy can be optimized without relying on higher-level temporal-difference bootstrapping over changing transition dynamics and rewards. We investigate future-state density estimation (Eysenbach et al., 2021b) as an alternative learning signal for high-level policy optimization. Specifically, let $p_f^{\pi^L}(\cdot \mid s, g)$ denote the discounted future-state density induced after predicting subgoal g at state s under the current low-level policy π^L. When this density is evaluated at the final task goal $g^\star$, $p_f^{\pi^L}(g^\star \mid s, g)$ provides a learning signal for how a pre-

dicted subgoal contributes to final-goal reachability without requiring higher-level temporal-difference bootstrapping.

However, directly applying future-state density estimation to hierarchical goal-conditioned control is non-trivial. Optimizing the high-level policy using $p_f^{\pi^L}(g^\star \mid s, g)$ encourages it to predict subgoals that increase the likelihood of achieving the final goal $g^\star$. However, the high-level policy may predict subgoals that are infeasible for the evolving low-level policy π^L, thereby hindering the achievement of the final goal $g^\star$. Our key insight is that high-level policy optimization should account for two distinct requirements: (i) whether a predicted subgoal contributes to achieving the final goal $g^\star$, and (ii) whether that subgoal can be achieved by the current low-level policy.

4 METHODOLOGY

In this section, we introduce Hierarchical Contrastive Control (HCC), which optimizes the high-level policy using future-state density estimation while incorporating a lower-level optimality constraint to guide feasible subgoal prediction. We first formulate the future-state density objective. We then incorporate the lower-level optimality constraint to obtain the HCC objective, and finally derive a practical approximation with twin-classifier stabilization.

4.1 HIGHER LEVEL POLICY OPTIMIZATION VIA FUTURE STATE DENSITY ESTIMATION

We first formulate the high-level objective for predicting efficient subgoals that achieve the final task goal $g^\star$. Let $\rho^L(\cdot \mid s_t, g_t)$ denote the discounted future-state density induced after predicting subgoal g_t at state s_t under the current low-level policy π^L. Specifically,

$$\rho^L(s_f \mid s_t, g_t) = \sum_{\Gamma=1}^{\infty} (1-\gamma)\gamma^{\Gamma-1} p_\Gamma(s_f \mid s_t, g_t), \tag{1}$$

where $\Gamma \geq 1$ denotes the number of high-level steps into the future, and $p_\Gamma(s_f \mid s_t, g_t)$ denotes the conditional density of reaching state s_f after Γ high-level steps, given that subgoal g_t is predicted at state s_t. At each high-level step, π^H predicts a subgoal and the low-level policy π^L then predicts primitive actions for the next k environment steps. Therefore, Γ high-level steps correspond to $\Gamma * k$ environment steps, and the associated future state is $s_{t+\Gamma*k}$. The geometric weighting $(1-\gamma)\gamma^{\Gamma-1}$ assigns larger weight to states reached at earlier high-level steps.

For the final task goal $g^\star$, $\rho^L(g^\star \mid s_t, g_t)$ denotes the future-state density assigned to $g^\star$ after predicting subgoal g_t. We therefore optimize the high-level policy using

$$J_F(\pi^H; \pi^L) = \mathbb{E}_{s_t \sim \mathcal{D}^H,\, g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \big[\log \rho^L(g^\star \mid s_t, g_t) \big], \tag{2}$$

where maximizing equation 2 trains the high-level policy to predict subgoals whose induced future-state distributions assign high density to the final task goal.

4.2 HIERARCHICAL CONTRASTIVE CONTROL

The future-state objective in equation 2 encourages the high-level policy to predict subgoals that contribute to achieving the final task goal, but it does not ensure that those subgoals are feasible for the current low-level policy. To address this, we formulate HCC as a constrained optimization approach that incorporates lower-level optimality into high-level policy optimization. For a state–subgoal pair (s_t, g_t), we denote the value under the current low-level policy by $V^L(s_t, g_t)$ and the corresponding optimal low-level value by $V^{L,\star}(s_t, g_t)$. Specifically,

$$V^L(s_t, g_t) = \mathbb{E}_{\pi^L, \mathcal{P}} \left[\sum_{j=0}^{k-1} \gamma^j r^L(s_{t+j}, a_{t+j}, g_t) \mid s_t, g_t \right], \qquad V^{L,\star}(s_t, g_t) = \max_{\pi^L} V^L(s_t, g_t). \tag{3}$$

By definition, $V^{L,\star}(s_t, g_t) \geq V^L(s_t, g_t)$ for every state–subgoal pair. Therefore, lower-level optimality can be imposed through the constraint

$$\mathbb{E}_{s_t \sim \mathcal{D}^H,\, g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \big[V^{L,\star}(s_t, g_t) - V^L(s_t, g_t) \big] \leq 0. \tag{4}$$

Since the quantity inside the expectation is always non-negative, equation 4 can only be satisfied when $V^L(s_t, g_t) = V^{L,\star}(s_t, g_t)$ for the state–subgoal pairs induced by the high-level policy. Thus, satisfying the constraint enforces lower-level optimality on the predicted subgoals.

We additionally include entropy regularization with coefficient $\alpha > 0$ for the high-level policy. Combining this constraint with the future-state objective in equation 2 gives the HCC formulation

$$\begin{aligned} \max_{\pi^H, \pi^L} \quad & \mathbb{E}_{s_t \sim \mathcal{D}^H,\, g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \big[\log \rho^L(g^\star \mid s_t, g_t) + \alpha \mathcal{H}\big(\pi^H(\cdot \mid s_t, g^\star)\big) \big] \\ \text{s.t.} \quad & \mathbb{E}_{s_t \sim \mathcal{D}^H,\, g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \big[V^{L,\star}(s_t, g_t) - V^L(s_t, g_t) \big] \leq 0. \end{aligned} \tag{5}$$

The corresponding Lagrangian objective is

$$\begin{aligned} J_{\text{HCC}}(\pi^H, \pi^L) = \mathbb{E}_{s_t \sim \mathcal{D}^H,\, g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \big[& \underbrace{\log \rho^L(g^\star \mid s_t, g_t)}_{\text{future-state density}} + \underbrace{\eta\big(V^L(s_t, g_t) - V^{L,\star}(s_t, g_t)\big)}_{\text{lower-level optimality}} \\ & + \underbrace{\alpha \mathcal{H}(\pi^H(\cdot \mid s_t, g^\star))}_{\text{entropy regularization}} \big]. \end{aligned} \tag{6}$$

The future-state density term encourages the high-level policy to predict subgoals that increase the density assigned to the final task goal, while the lower-level optimality term encourages the high-level policy to predict feasible subgoals. An alternative interpretation of equation 6 as regularizing the high-level policy toward a lower-level-optimality-induced subgoal prior is provided in Appendix A.2.

4.3 Practical HCC Algorithm

Approximating lower-level optimality. The HCC objective in equation 6 depends on the optimal low-level value function $V^{L,\star}$, which is unavailable during training. We approximate the optimal low-level value using the value function V^L_m obtained after m low-level optimization updates. We use $m = 5$ in all experiments.

$$V^{L,\star}(s_t, g_t) \approx V^L_m(s_t, g_t). \tag{7}$$

Future-state density estimation. To estimate the future-state density in equation 6, we follow prior work on C-Learning (Eysenbach et al., 2021b;a) and learn a binary future-state classifier from trajectories stored in the replay buffer $\mathcal{D}^H$. For each state–subgoal pair (s_t, g_t), we sample a future offset $\Gamma \sim \text{Geom}(1-\gamma)$ and use $s_f = s_{t+\Gamma * k}$ as a positive future-state sample, while a random state $s_r \sim p_{\mathcal{D}^H}$ is used as a negative sample. The classifier $C_\psi(c \mid s_t, g_t, s')$ is trained using

$$\begin{aligned} \mathcal{L}_C(\psi) = & - \mathbb{E}_{(s_t, g_t) \sim \mathcal{D}^H,\, s_f \sim \rho^L(\cdot \mid s_t, g_t)} \big[\log C_\psi(c = 1 \mid s_t, g_t, s_f) \big] \\ & - \mathbb{E}_{(s_t, g_t) \sim \mathcal{D}^H,\, s_r \sim p_{\mathcal{D}^H}} \big[\log \left(1 - C_\psi(c = 1 \mid s_t, g_t, s_r)\right) \big]. \end{aligned} \tag{8}$$

Following C-Learning (Eysenbach et al., 2021b;a), the Bayes-optimal classifier is monotonic in the future-state density ratio. Therefore, for a fixed final task goal $g^\star$, we use the classifier score as a surrogate for the future-state density term in equation 6:

$$J^C_F(\pi^H) = \mathbb{E}_{s_t \sim \mathcal{D}^H,\, g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \big[\log C_\psi(c = 1 \mid s_t, g_t, g^\star) \big]. \tag{9}$$

The corresponding density-ratio derivation and its connection to the future-state objective in equation 2 are provided in Appendix A.3.

Twin-classifier stabilization. To stabilize this classifier-based future-state objective, HCC maintains two online classifiers C_{ψ_1} and C_{ψ_2}, each trained using equation 8, together with corresponding target classifiers $C_{\bar{\psi}_1}$ and $C_{\bar{\psi}_2}$. The target parameters are updated using Polyak averaging (Polyak & Juditsky, 1992):

$$\bar{\psi}_i \leftarrow (1 - \tau)\bar{\psi}_i + \tau \psi_i, \qquad i \in \{1, 2\}, \tag{10}$$

where $\tau \in (0, 1]$ controls the target-update rate, with larger values corresponding to faster target updates. We use the minimum of the two target-classifier scores to obtain a conservative surrogate for the future-state objective:

$$T(s_t, g_t, g^\star) = \min_{i \in \{1,2\}} \log C_{\bar{\psi}_i}(c = 1 \mid s_t, g_t, g^\star). \tag{11}$$

Practical HCC objective. Substituting the low-level value approximation in equation 7 and the twin-classifier surrogate in equation 11 into equation 6 gives the practical HCC objective

$$\widehat{J}_{\mathrm{HCC}}(\pi^H, \pi^L) = \mathbb{E}_{s_t \sim \mathcal{D}^H,\, g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \Big[\underbrace{T(s_t, g_t, g^\star)}_{\text{future-state density}} + \underbrace{\eta\big(V^L(s_t, g_t) - V_m^L(s_t, g_t)\big)}_{\text{lower-level optimality}} + \underbrace{\alpha \mathcal{H}\big(\pi^H(\cdot \mid s_t, g^\star)\big)}_{\text{entropy regularization}} \Big]. \tag{12}$$

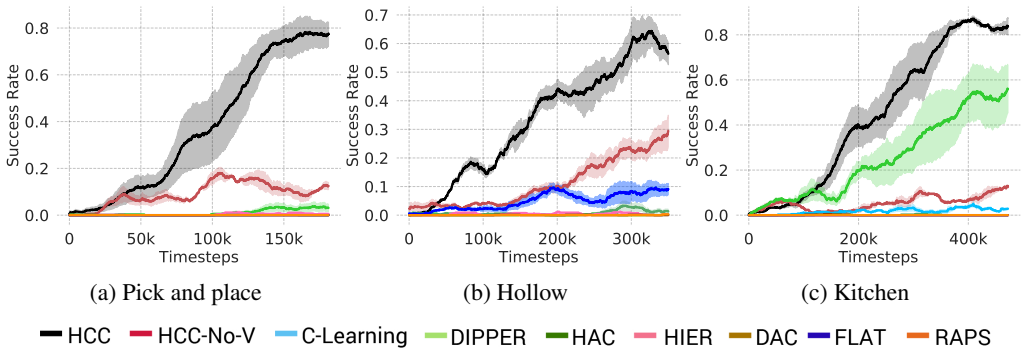
Low-level policy optimization. From equation 12, the low-level policy is optimized to maximize $V^L(s_t, g_t)$. We implement this using Soft Actor-Critic (SAC) (Haarnoja et al., 2018) with the intrinsic reward $r^L(s_i, a_i, g_t) = -\|s_{i+1} - g_t\|_2$. Algorithm 1 summarizes the complete training procedure.

5 Experiments

In our empirical analysis, we aim to answer the following questions:

- **RQ1.** How does HCC compare against existing hierarchical and non-hierarchical methods?
- **RQ2.** Does the lower-level optimality constraint improve task performance and subgoal feasibility?
- **RQ3.** How do HCC's stabilization mechanisms contribute to learning stability and performance?

Task and Environment Details. We evaluate HCC on three continuous-control robotic manipulation environments: (i) Pick and Place, requiring object interaction and transport; (ii) Hollow, requiring the agent to pick up a hollow block and place it on a table such that a vertical pole passes through its center; and (iii) Franka Kitchen (Gupta et al., 2019), a long-horizon task requiring the sequential execution of multi-step subtasks (e.g., opening a microwave, then activating a stove). All environments employ a sparse reward structure, where the agent receives a binary reward exclusively upon achieving the final task goal. The implementation details and hyperparameters are provided in Appendix A.5. To accelerate the learning of low-level primitives across all evaluated methods, we incorporate a single human demonstration via an additional imitation learning objective. We consistently apply this condition across all baselines to ensure fair comparisons. We additionally evaluate HCC without demonstration supervision and find that it continues to learn across all three environments, showing that the demonstration improves sample efficiency but is not required for HCC to learn; results are provided in Appendix A.4.1.



(a) Pick and place (b) Hollow (c) Kitchen

Figure 2: **Success Rate plots.** This figure provides success rate plots to compare HCC against various baselines across three sparse-reward robotic manipulation tasks: Pick and place, hollow and franka kitchen. The solid lines represent the mean and shaded regions denote the standard deviation across five random seeds. HCC-No-V denotes HCC without the lower-level optimality constraint.

RQ1. How does HCC compare against existing hierarchical and non-hierarchical methods?

We compare HCC against a suite of hierarchical and non-hierarchical baselines. Figure 2 reports the success rates averaged over five random seeds.

Comparison with Flat Contrastive Learning. To evaluate the necessity of the hierarchical formulation, we compare against C-Learning (CL) (Eysenbach et al., 2021b), a flat goal-conditioned method

that derives policies directly from future-state density estimates. Under sparse rewards, C-Learning struggles to solve long-horizon tasks. HCC substantially outperforms C-Learning, demonstrating the benefit of hierarchical decomposition when applying future-state density estimation to complex, multi-stage tasks.

Comparison with Hierarchical Baselines.

DIPPER baseline. We compare against DIPPER (Singh et al., 2026), an HRL method that leverages preference-based learning to stabilize HRL. HCC achieves superior performance compared to this baseline, demonstrating the effectiveness of future-state density estimation with lower-level optimality for hierarchical goal-conditioned learning.

HAC baseline. We also implement HAC (Levy et al., 2018), an HRL baseline that attempts to address shifting transition dynamics by simulating optimal low-level transitions via hindsight action transitions. Despite this mechanism, we find that HCC outperforms HAC in the complex manipulation tasks. This suggests that our formulation provides a more effective learning signal for high-level policy optimization than HAC's hindsight transitions.

RAPS baseline. We consider RAPS (Dalal et al., 2021), which optimizes a high-level policy over a fixed library of parameterized action primitives. While such primitives can provide effective temporal abstraction when suitable low-level behaviors are available, constructing an appropriate primitive library can be difficult for complex manipulation tasks. HCC outperforms this baseline without requiring engineered priors, demonstrating its broad applicability.

HIER baseline. We evaluate HIER, a standard HRL baseline applying SAC (Haarnoja et al., 2018) at both hierarchy levels. HIER exhibits poor performance across the evaluated environments, highlighting the difficulty of higher-level temporal-difference learning when the induced high-level transition dynamics and rewards change as the low-level policy evolves.

Comparison with Non-Hierarchical Baselines.

We include **DAC** (Kostrikov et al., 2018), which uses demonstrations, and **FLAT**, a standard SAC baseline. These baselines fail to make significant progress on the robotic manipulation tasks, further demonstrating the importance of hierarchical abstraction for these long-horizon problems.

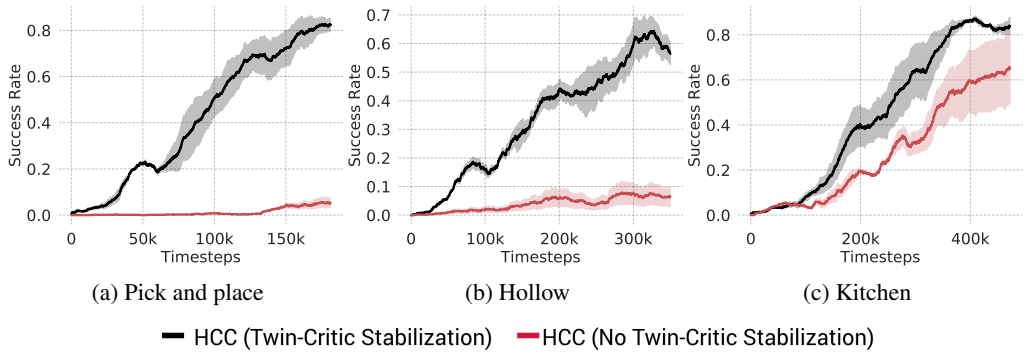
RQ2. Does the lower-level optimality constraint improve task performance and subgoal feasibility?

Here, we show that directly applying future-state density estimation to hierarchical control is non-trivial. The future-state objective encourages the high-level policy to predict subgoals that increase the state density assigned to the final task goal, but it does not account for whether those subgoals are feasible for the current low-level policy. This distinction is specific to the hierarchical setting, where the predicted subgoal is executed by a separately learned low-level policy whose behavior changes throughout training.

To isolate the importance of this requirement, we evaluate HCC-No-V, which removes the lower-level optimality constraint from HCC while retaining the hierarchical future-state density objective. As shown in Figure 2, HCC consistently outperforms HCC-No-V across the evaluated tasks, which shows that optimizing the future-state objective alone is insufficient: high-level policy optimization must also account for the ability of the current low-level policy to achieve the predicted subgoal. The lower-level optimality constraint in HCC explicitly incorporates this requirement.

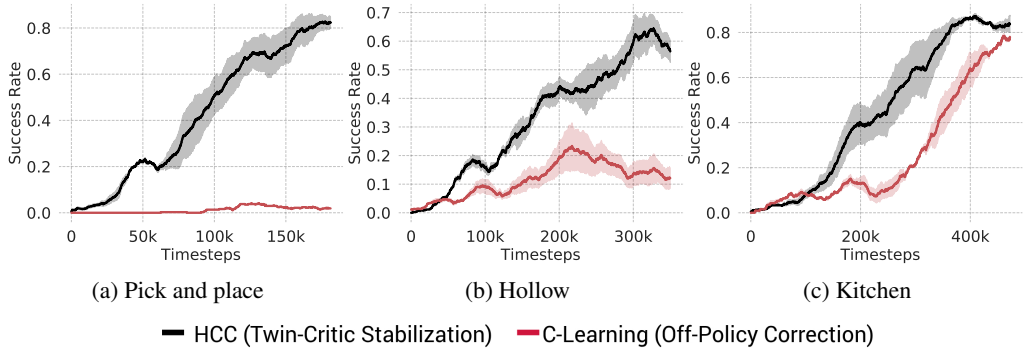
Subgoal Distance Metric. To further demonstrate that HCC predicts feasible subgoals, we measure the average euclidean distance between each subgoal predicted by the high-level policy and the state subsequently reached by the low-level policy across multiple training stages (Initial, Middle, Final). A smaller distance value indicates that the lower level policy is able to achieve the predicted subgoal. Appendix A.4.2 Figure 5 shows that HCC consistently achieves smaller predicted-to-achieved distance values than the hierarchical baselines across tasks.

Lower Q-Function Metric. We measure the average low-level Q-value evaluated at the subgoals specified by the high-level policy. Higher Q-values indicate that the low-level policy achieves high expected lower level rewards for the assigned subgoals, which naturally occurs if the subgoals are reachable. Appendix A.4.2 Figure 6 shows that HCC maintains higher low-level Q-values than HIER, RAPS, and HAC. Together with the performance gap between HCC and HCC-No-V, these results provide empirical evidence that ensuring feasible subgoal generation leads to high-level predictions that are better aligned with the current low-level policy.



(a) Pick and place (b) Hollow (c) Kitchen

Figure 3: **Twin-Classifier Stabilization Ablation.** We compare full HCC against a variant using a single classifier without target-network stabilization across Pick and Place, Hollow, and Franka Kitchen. The full twin-classifier formulation consistently achieves higher success rates, demonstrating the benefit of stabilizing the classifier-based future-state learning signal.



(a) Pick and place (b) Hollow (c) Kitchen

Figure 4: **Comparison with C-Learning's Off-Policy Correction.** We compare HCC's twin-classifier stabilization with the importance-sampling-based off-policy correction used in C-Learning (Eysenbach et al., 2021b). HCC's twin-classifier formulation achieves higher success rates across all tasks.

RQ3. How do HCC's stabilization mechanisms affect learning stability and performance?

We next evaluate the practical stabilization mechanisms introduced in Section 4.3. We first examine the behavior of the HCC objective without explicit classifier stabilization, and then evaluate the twin-classifier formulation, compare it with the off-policy correction used in C-Learning, and analyze the effect of Polyak-averaged target classifiers.

Inherent Stability of the Contrastive Formulation. To isolate the stability benefits of our core objective from the proposed stabilization mechanisms, we evaluate the performance of HCC without the twin-classifier and Polyak averaging. Appendix Figure 14 compares full HCC, this ablated HCC variant (no stabilization), and the standard HIER baseline across tasks. We observe that even in the absence of explicit stabilization mechanisms, the pure contrastive HCC formulation consistently outperforms the HIER baseline. This performance gap provides empirical evidence that the future-state density objective provides a more stable high-level learning signal than standard temporal-difference bootstrapping in the jointly trained hierarchical setting. Furthermore, the performance gain from the ablated variant to full HCC demonstrates the additional benefit of the twin-classifier and Polyak-averaging mechanisms for stabilizing the classifier-based future-state objective.

Effect of Twin-Classifier Stabilization. To separately examine the twin-classifier design, we compare full HCC against a variant that uses a single classifier without target-network stabilization. As shown in Figure 3, the full twin-classifier formulation consistently achieves higher success rates across all three tasks. This result demonstrates the benefit of the full twin-classifier target design used to construct the conservative surrogate in equation 11.

Comparison with C-Learning Off-Policy Correction. C-Learning (Eysenbach et al., 2021b) addresses off-policy future-state estimation using an importance-sampling-based correction. To

evaluate whether this correction transfers effectively to the jointly trained hierarchical setting, we replace HCC's twin-classifier stabilization with the C-Learning off-policy correction while retaining the remaining HCC formulation. Figure 4 shows that HCC's twin-classifier formulation consistently achieves higher success rates across the evaluated tasks. This result indicates that the proposed stabilization provides a more effective future-state learning signal than directly applying the original C-Learning correction when the low-level policy is learned concurrently.

Effect of Polyak Averaging. To assess the impact of slow target updates, we perform an ablation in Appendix A.4.3 Figure 7 where we remove the Polyak-averaging stabilization, effectively updating the target classifiers instantaneously. We observe that this modification leads to sub-optimal performance in the Pick and Place and Kitchen tasks, confirming the benefit of slowly changing classifier targets in these environments. Surprisingly, however, we find that fast updates actually benefit the agent in the Hollow task. We hypothesize that because Hollow is an extremely sparse-reward environment where positive signal is rare, the dampening effect of Polyak averaging may excessively slow down the incorporation of newly observed positive future-state samples. This suggests that there exists a trade-off between maintaining stable classifier targets and rapidly incorporating rare positive future-state information in extremely sparse-reward environments.

Additional Ablations and Sensitivity Analysis.

Robustness to Varying Goal Distributions. A critical capability for goal-conditioned HRL is the ability to generalize across diverse and changing goals. To evaluate the robustness of our approach under such conditions, we conduct an experiment on the Pick and Place task where the final goal positions are randomly generated across episodes. Appendix A.4.4 Figure 8 illustrates the success rates of HCC compared to the baselines in this setting. Despite the increased task complexity, HCC maintains strong performance and consistently outperforms all baselines. This demonstrates that HCC robustly generalizes across multiple goal configurations.

Sensitivity Analysis of Hyperparameters. We perform a comprehensive sensitivity analysis to evaluate the robustness of HCC across its five primary hyperparameters: lower-level optimality weight (η), discount factor (γ), subgoal interval (k), Polyak averaging rate (τ), and imitation weight (λ), and provide the figures in Appendix A.4.5. We first analyze the effect of the lower-level optimality weight η in equation 12. Appendix A.4.5 Figure 9 compares different values of η across all three environments, complementing the HCC-No-V ablation in RQ2 by examining how the relative contribution of the lower-level optimality term affects task performance. We next analyze the effect of varying the discount factor γ on the success rates (Appendix A.4.5 Figure 10). We also evaluate the subgoal interval k using task-appropriate ranges reflecting differing task horizons (Appendix A.4.5 Figure 11). An optimal k balances temporal abstraction: small k reduces the benefits of hierarchy, while large k makes subgoals harder for the low-level policy to reach. Finally, we vary the Polyak averaging rate τ and the imitation weight λ across all three tasks (Appendix A.4.5 Figures 12 and 13). τ balances the stability and learning speed of the target classifiers, while λ modulates the strength of the imitation objective against standard RL. Overall, our results show that HCC maintains reasonable stability and consistent performance across these hyperparameter ranges.

6 Discussion

Limitations and Future Work. In our formulation, applying HCC to extremely high-dimensional subgoal spaces, such as pixel-based subgoals, remains challenging because accurately estimating future-state densities through binary classification becomes increasingly difficult in such spaces. Further, HCC introduces additional overhead compared with standard HRL because high-level policy optimization additionally requires learning and maintaining the future-state classifiers. An interesting direction for future work is therefore to develop more computationally efficient future-state estimators that retain the benefits of HCC while reducing this additional training cost.

Conclusion. In this work, we introduced Hierarchical Contrastive Control (HCC), which optimizes the high-level policy using future-state density estimation together with a lower-level optimality constraint for feasible subgoal prediction. Across sparse-reward robotic tasks, HCC consistently improves over the evaluated hierarchical and non-hierarchical baselines, while our ablations demonstrate the importance of lower-level optimality and classifier stabilization. These results support future-state density estimation as an effective alternative to higher-level temporal-difference learning in jointly trained hierarchical control.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, code debugging, and figure preparation. We additionally used generative AI tools to assist with drafting and editing the manuscript, identifying relevant literature, and improving the organization and readability of the paper. All AI-assisted experimental analyses, figures, citations, and written content were reviewed and verified by the authors. The authors take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies learning methods for hierarchical robotic control in benchmark manipulation environments. The experiments do not involve human subjects or sensitive personal data. A single human-provided demonstration is used as an auxiliary signal for low-level policy learning and is applied consistently across the evaluated methods. While hierarchical control methods may eventually be applied to real-world robotic systems, such deployment would require additional task-specific safety constraints and validation that are outside the scope of this work.

REPRODUCIBILITY STATEMENT

We provide the complete HCC formulation and training procedure in Section 4 and Algorithm 1. The implementation details, environment configurations, baseline settings, and hyperparameters are provided in Appendix A.5. Additional derivations, ablations, and analyses are included in the appendix. We intend to release the complete implementation and evaluation code upon publication.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13:341–379, 2003.

Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34:21847–21859, 2021.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 5, pp. 271–278, 1993.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *CoRR*, cs.LG/9905014, 1999. URL https://arxiv.org/abs/cs/9905014.

Ben Eysenbach, Sergey Levine, and Russ R Salakhutdinov. Replacing rewards with examples: Example-based policy search via recursive classification. *Advances in Neural Information Processing Systems*, 34:11541–11552, 2021a.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021b.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

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

Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learnings options end-to-end for continuous action tasks. *arXiv preprint arXiv:1712.00004*, 2017.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pp. 5639–5650. PMLR, 2020.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations*, 2018.

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

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In M. Jordan, M. Kearns, and S. Solla (eds.), *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Utsav Singh and Vinay P. Namboodiri. PEAR: primitive enabled adaptive relabeling for boosting hierarchical reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=0nJEgNpb4l.

Utsav Singh, Souradip Chakraborty, Wesley Suttle, Brian Sadler, Derrik Asher, Anit Kumar Sahu, Mubarak Shah, Vinay Purushothaman Namboodiri, and Amrit Bedi. Direct preference optimization for primitive-enabled hierarchical rl: A bilevel approach. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 1808–1828, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/03f99ca79b87c513d0b502e737a41a41-Paper-Conference.pdf.

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

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

CONTENTS

1 Introduction **1**

2 Related Work **2**

3 Problem Formulation **3**

4 Methodology **4**

4.1 Higher level policy optimization via future state density estimation 4

4.2 Hierarchical Contrastive Control 4

4.3 Practical HCC Algorithm 5

5 Experiments **6**

6 Discussion **9**

A Appendix **12**

A.1 HCC Algorithm 12

A.2 HCC as KL-Regularized High-Level Policy Optimization 13

A.3 Future-State Density Estimation via Classification 14

A.4 Additional Experiments 15

A.4.1 Effect of Demonstration Supervision 15

A.4.2 Subgoal Feasibility and Training Stability 15

A.4.3 Effect of Polyak Averaging 17

A.4.4 Robustness to Varying Goal Distributions 17

A.4.5 Additional Ablation Experiments 18

A.5 Implementation Details 19

A.6 Environment Specifications 20

A.7 Qualitative visualizations 21

A APPENDIX

A.1 HCC ALGORITHM

Here we provide the algorithm for HCC.

Algorithm 1 Hierarchical Contrastive Control (HCC)

Initialize: High-level policy π^H, low-level policy π^L, and low-level SAC critic.
Initialize: Online classifiers C_{ψ_1}, C_{ψ_2} and target classifiers $C_{\bar{\psi}_1}, C_{\bar{\psi}_2}$ with $\bar{\psi}_i \leftarrow \psi_i$.
Initialize: Replay buffers $\mathcal{D}^L, \mathcal{D}^H$.
for each episode **do**
  Sample final goal $g^\star$ and initial state s_0.
  for $t = 0, k, 2k, \dots$ **do**
    Sample subgoal $g_t \sim \pi^H(\cdot \mid s_t, g^\star)$.
    for $j = 0, \dots, k-1$ **do**
      Sample action $a_{t+j} \sim \pi^L(\cdot \mid s_{t+j}, g_t)$.
      Execute a_{t+j} and observe s_{t+j+1}.
      Store $(s_{t+j}, a_{t+j}, s_{t+j+1}, g_t)$ in $\mathcal{D}^L$.
    Store $(s_t, g_t, s_{t+k}, g^\star)$ in $\mathcal{D}^H$.

    Low-level update:
    Cache the current low-level value $V^L(s, g)$ for sampled state–subgoal pairs.
    for $\ell = 1, \dots, m$ **do**
      Update π^L and its critic using SAC on $\mathcal{D}^L$ with intrinsic reward r^L.
    Let V^L_m denote the resulting low-level value estimate.

    Future-state density estimation:
    Sample (s, g) from $\mathcal{D}^H$ and $\Gamma \sim \mathrm{Geom}(1-\gamma)$.
    Set positive sample $s_f = s_{t+\Gamma * k}$ from the same trajectory and negative state $s_r \sim p_{\mathcal{D}^H}$.
    Update C_{ψ_1} and C_{ψ_2} by minimizing $\mathcal{L}_C$ in equation 8.
    for $i \in \{1, 2\}$ **do**
      $\bar{\psi}_i \leftarrow (1-\tau)\bar{\psi}_i + \tau\psi_i$.
    Compute the conservative future-state surrogate $T(s, g, g^\star)$ using equation 11.

    High-level update:
    Update π^H to maximize the practical HCC objective $\widehat{J}_{\mathrm{HCC}}$ in equation 12.

A.2 HCC as KL-Regularized High-Level Policy Optimization

We provide an alternative interpretation of HCC as KL-regularized high-level policy optimization. For a fixed low-level policy π^L, consider the objective

$$J_{\mathrm{KL}}(\pi^H) = \mathbb{E}_{s_t \sim \mathcal{D}}\Big[\mathbb{E}_{g_t \sim \pi^H(\cdot \mid s_t, g^\star)}\big[\log \rho^L(g^\star \mid s_t, g_t)\big] - \alpha D_{\mathrm{KL}}\left(\pi^H(\cdot \mid s_t, g^\star) \,\|\, \pi_{\mathrm{prior}}(\cdot \mid s_t)\right)\Big], \tag{13}$$

where π_{prior} is a prior over subgoals induced by the current low-level policy. We define this prior as

$$\pi_{\mathrm{prior}}(g_t \mid s_t) = \frac{1}{Z(s_t)} \exp\left(\frac{\eta}{\alpha}\left[V^L(s_t, g_t) - V^{L,\star}(s_t, g_t)\right]\right), \tag{14}$$

where

$$Z(s_t) = \int_{\mathcal{G}} \exp\left(\frac{\eta}{\alpha}\left[V^L(s_t, g) - V^{L,\star}(s_t, g)\right]\right) dg \tag{15}$$

is the normalization constant. Since $V^L(s_t, g_t) - V^{L,\star}(s_t, g_t) \le 0$, the prior assigns greater probability to subgoals for which the current low-level policy has a smaller optimality gap.

To establish the connection with HCC, expanding the KL term in equation 13 gives

$$\begin{aligned} &- \alpha D_{\mathrm{KL}}\left(\pi^H(\cdot \mid s_t, g^\star) \,\|\, \pi_{\mathrm{prior}}(\cdot \mid s_t)\right) \\ &= \alpha \mathcal{H}\left(\pi^H(\cdot \mid s_t, g^\star)\right) + \alpha \mathbb{E}_{g_t \sim \pi^H(\cdot \mid s_t, g^\star)}\left[\log \pi_{\mathrm{prior}}(g_t \mid s_t)\right]. \end{aligned} \tag{16}$$

From equation 14,

$$\alpha \log \pi_{\text{prior}}(g_t \mid s_t) = \eta \left[V^L(s_t, g_t) - V^{L,\star}(s_t, g_t) \right] - \alpha \log Z(s_t). \tag{17}$$

Substituting equation 17 into equation 16 yields

$$\begin{aligned} &- \alpha D_{\text{KL}} \left(\pi^H(\cdot \mid s_t, g^\star) \,\|\, \pi_{\text{prior}}(\cdot \mid s_t) \right) \\ &= \eta \mathbb{E}_{g_t \sim \pi^H(\cdot \mid s_t, g^\star)} \left[V^L(s_t, g_t) - V^{L,\star}(s_t, g_t) \right] + \alpha \mathcal{H} \left(\pi^H(\cdot \mid s_t, g^\star) \right) - \alpha \log Z(s_t). \end{aligned} \tag{18}$$

Therefore,

$$J_{\text{KL}}(\pi^H) = J_{\text{HCC}}(\pi^H; \pi^L) - \alpha \mathbb{E}_{s_t \sim \mathcal{D}} \big[\log Z(s_t) \big], \tag{19}$$

where J_{HCC} denotes the high-level optimization objective in equation 6. For a fixed low-level policy π^L, $Z(s_t)$ is independent of π^H. Hence, maximizing equation 13 with respect to π^H is equivalent to maximizing equation 6.

This interpretation also gives the corresponding optimal high-level policy. For a fixed state s_t and low-level policy π^L, equation 13 has the solution

$$\pi^{H,\star}(g_t \mid s_t, g^\star) = \frac{\pi_{\text{prior}}(g_t \mid s_t) \rho^L(g^\star \mid s_t, g_t)^{1/\alpha}}{\displaystyle\int_{\mathcal{G}} \pi_{\text{prior}}(g \mid s_t) \rho^L(g^\star \mid s_t, g)^{1/\alpha} dg}. \tag{20}$$

Equivalently,

$$\pi^{H,\star}(g_t \mid s_t, g^\star) \propto \rho^L(g^\star \mid s_t, g_t)^{1/\alpha} \exp\left(\frac{\eta}{\alpha} \left[V^L(s_t, g_t) - V^{L,\star}(s_t, g_t) \right] \right). \tag{21}$$

Thus, HCC balances two signals: the future-state density term favors subgoals that contribute to final-goal reachability, while the low-level-induced prior favors subgoals with smaller low-level optimality gaps. Consequently, the high-level policy is regularized toward subgoals that contribute to the final task goal while being better aligned with the current low-level policy.

A.3 Future-State Density Estimation via Classification

We provide the density-ratio derivation underlying the classifier-based future-state objective used in equation 9. This follows the standard C-Learning formulation (Eysenbach et al., 2021b;a).

For a fixed state–subgoal pair (s_t, g_t), positive samples are drawn from the future-state density

$$s' \sim \rho^L(\cdot \mid s_t, g_t), \tag{22}$$

while negative samples are drawn independently from the replay-buffer marginal

$$s' \sim p_{\mathcal{D}}(\cdot). \tag{23}$$

Under the equal class weighting used in equation 8, Bayes' rule gives the optimal binary classifier

$$C^\star(c = 1 \mid s_t, g_t, s') = \frac{\rho^L(s' \mid s_t, g_t)}{\rho^L(s' \mid s_t, g_t) + p_{\mathcal{D}}(s')}. \tag{24}$$

Consequently,

$$1 - C^\star(c = 1 \mid s_t, g_t, s') = \frac{p_{\mathcal{D}}(s')}{\rho^L(s' \mid s_t, g_t) + p_{\mathcal{D}}(s')}. \tag{25}$$

Taking the ratio gives

$$\frac{C^\star(c = 1 \mid s_t, g_t, s')}{1 - C^\star(c = 1 \mid s_t, g_t, s')} = \frac{\rho^L(s' \mid s_t, g_t)}{p_{\mathcal{D}}(s')}. \tag{26}$$

Equivalently,

$$\rho^L(s' \mid s_t, g_t) = \frac{C^\star(c = 1 \mid s_t, g_t, s')}{1 - C^\star(c = 1 \mid s_t, g_t, s')} p_{\mathcal{D}}(s'). \tag{27}$$

Evaluating equation 27 at the final task goal $s' = g^\star$ and taking logarithms gives

$$\log \rho^L(g^\star \mid s_t, g_t) = \log \frac{C^\star(c = 1 \mid s_t, g_t, g^\star)}{1 - C^\star(c = 1 \mid s_t, g_t, g^\star)} + \log p_{\mathcal{D}}(g^\star). \tag{28}$$

For a fixed final task goal $g^\star$, the marginal term $\log p_{\mathcal{D}}(g^\star)$ does not depend on the predicted subgoal g_t. Therefore,

$$\arg\max_{g_t} \log \rho^L(g^\star \mid s_t, g_t) = \arg\max_{g_t} \log \frac{C^\star(c = 1 \mid s_t, g_t, g^\star)}{1 - C^\star(c = 1 \mid s_t, g_t, g^\star)}. \tag{29}$$

Since both $C/(1 - C)$ and $\log C$ are strictly increasing functions of $C \in (0, 1)$,

$$\arg\max_{g_t} \log \frac{C^\star(c = 1 \mid s_t, g_t, g^\star)}{1 - C^\star(c = 1 \mid s_t, g_t, g^\star)} = \arg\max_{g_t} \log C^\star(c = 1 \mid s_t, g_t, g^\star). \tag{30}$$

Thus, the classifier objective

$$J_F^C(\pi^H) = \mathbb{E}_{\substack{s_t \sim \mathcal{D}, \\ g_t \sim \pi^H(\cdot \mid s_t, g^\star)}} [\log C_\psi(c = 1 \mid s_t, g_t, g^\star)] \tag{31}$$

provides a monotonic surrogate for the future-state density objective in equation 2. In practice, HCC replaces this single-classifier score with the conservative twin-classifier surrogate in equation 11.

A.4 Additional Experiments

A.4.1 Effect of Demonstration Supervision

To determine whether HCC's gains are driven by the single demonstration rather than by the proposed high-level learning objective, we remove the imitation-learning term entirely and train both HCC and HIER using reinforcement learning alone. Table 1 reports the final success rates using the same training budget as the corresponding experiments in the main paper.

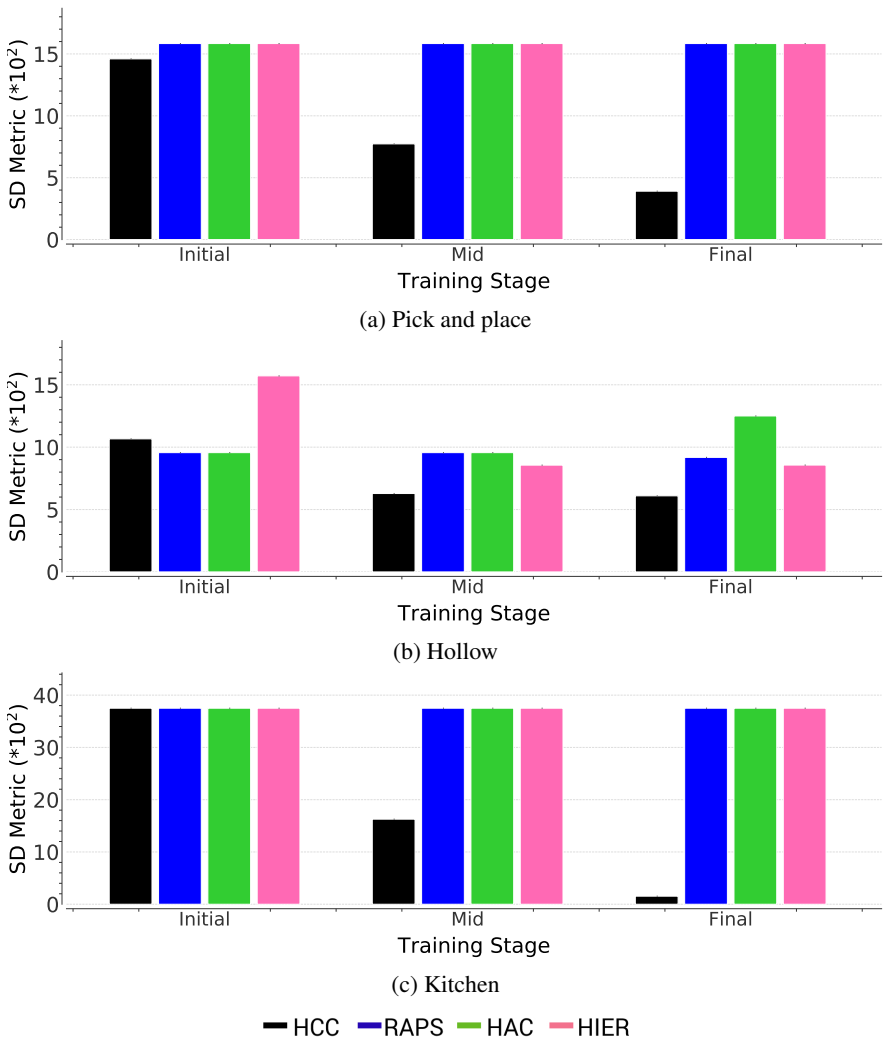
Table 1: **Effect of demonstration supervision.** Success rates of HCC and HIER with and without the single demonstration used for auxiliary low-level imitation learning. Results are averaged over five random seeds using the same training budget.

Method	Pick & Place	Hollow	Kitchen
HCC (with 1 demonstration)	**0.81**	**0.67**	**0.82**
HIER (with 1 demonstration)	0.05	0.10	0.11
HCC (no demonstration)	**0.55**	**0.41**	**0.51**
HIER (no demonstration)	0.00	0.00	0.00

HCC continues to learn without demonstration supervision and substantially outperforms HIER across all three environments, whereas HIER remains near zero without the demonstration. The demonstration therefore improves sample efficiency but does not account for HCC's advantage over the TD-based hierarchical baseline.

A.4.2 Subgoal Feasibility and Training Stability

To quantitatively verify that HCC stabilizes the hierarchical learning process and avoids generating unfeasible subgoals, we evaluate the Subgoal Distance and Lower Q-Function metrics across multiple training stages. As shown in Figures 5 and 6, HCC consistently generates feasible, high-return subgoals that induce optimal low-level behavior, successfully stabilizing the coupled optimization process compared to standard baselines.

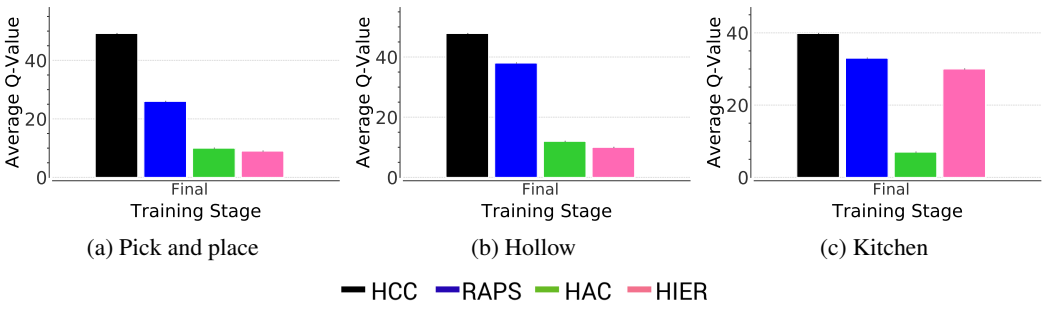


(a) Pick and place

(b) Hollow

(c) Kitchen

Figure 5: **Subgoal Distance Metric Comparison.** This figure tracks the Euclidean distance between the subgoals predicted by the high-level policy and the actual States achieved by the low-level policy across different training stages (Initial: early training, Mid: intermediate training, Final: convergence). HCC consistently yields minimal distance between the predicted and achieved subgoals throughout the optimization process, demonstrating that it generates structurally achievable subgoals that induce stable low-level policy execution.



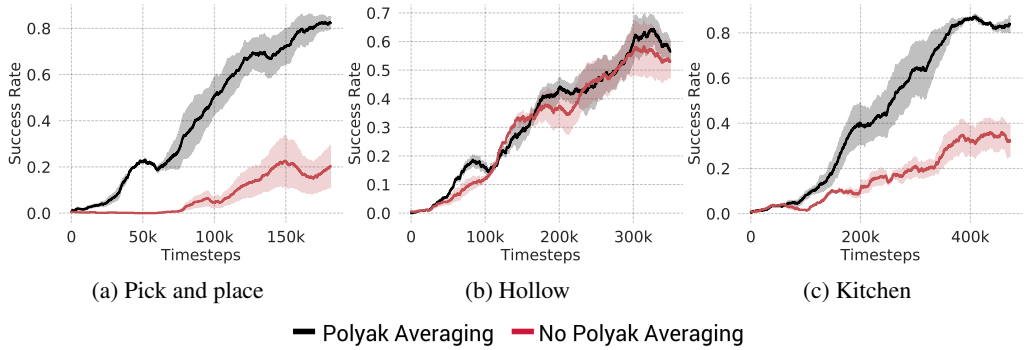
(a) Pick and place (b) Hollow (c) Kitchen

Figure 6: **Lower Q-Function Metric Comparison.** This figure compares the average low-level Q-function values evaluated at the subgoals predicted by the high-level policy. Higher values indicate that the low-level policy predicts a high probability of success for the assigned subgoals, implying near-optimal low-level execution and a stable learning signal for the high-level policy. HCC consistently sustains higher Q-values than HIER, RAPS, and HAC across all environments, empirically confirming that HCC effectively stabilizes hierarchical training against shifting transition dynamics. The plotted values include a fixed positive constant shift for visualization purposes.

A.4.3 Effect of Polyak Averaging

We further study the role of Polyak averaging in stabilizing the target classifiers used by HCC. In this ablation, we remove Polyak averaging by setting $\tau = 1$, which corresponds to instantaneous target updates. Figure 7 compares this variant with the default HCC configuration across all three environments.

We observe that removing Polyak averaging leads to worse performance on Pick and Place and Kitchen, indicating that slowly updated target classifiers improve stability in these more complex manipulation tasks. Interestingly, instantaneous updates perform well on Hollow. We hypothesize that because Hollow is an extremely sparse-reward task, faster target updates may help propagate the rare positive learning signal more quickly. Overall, these results suggest a trade-off between stability and signal propagation speed, while confirming that Polyak averaging is generally beneficial for HCC.

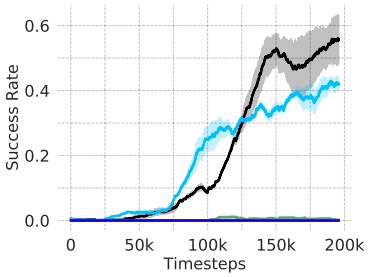


(a) Pick and place (b) Hollow (c) Kitchen

Figure 7: **Polyak-averaging ablation.** We investigate the effect of removing Polyak averaging ($\tau = 1$, instant updates) on performance. While fast updates degrade stability in complex manipulation tasks like Pick and Place and Kitchen, they surprisingly perform well in the extremely sparse Hollow task. This suggests a trade-off: smooth updates are critical for stability in complex tasks, while faster updates may aid signal propagation in environments with extremely sparse feedback.

A.4.4 Robustness to Varying Goal Distributions

To evaluate the generalization capabilities of our approach, we test HCC on a multi-goal variant of the Pick and place task where final goal positions are randomly generated across episodes. As shown in Figure 8, HCC maintains strong performance and consistently outperforms all baselines, demonstrating its robustness to diverse goal configurations.



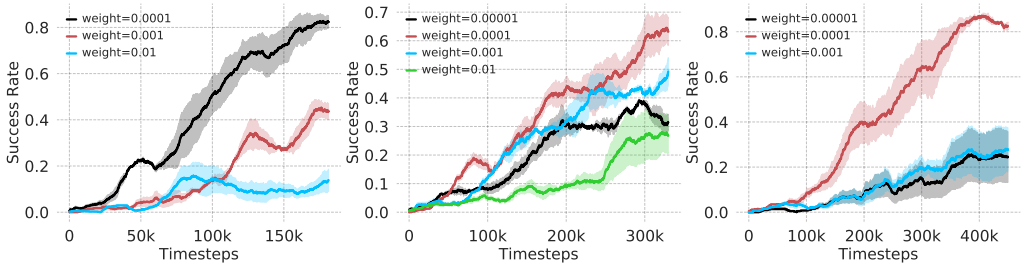
(a) Pick and place



Figure 8: **Pick and Place Performance with Multiple Goals.** This figure illustrates the success rates for pick and place task, where the solid lines represent the mean, and the shaded areas denote the standard deviation across 5 different seeds. In this task, we randomly generate the final goal positions. HCC still demonstrates strong performance and is able to consistently outperform the baselines.

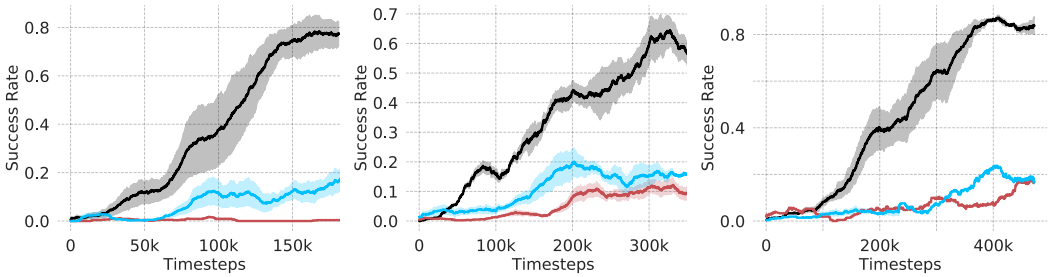
A.4.5 Additional Ablation Experiments

Here, we provide additional ablations to analyze the effect of varying lower-level optimality weight η, discount factor (γ), subgoal interval (k), Polyak averaging rate (τ), and imitation weight (λ) hyperparameters.



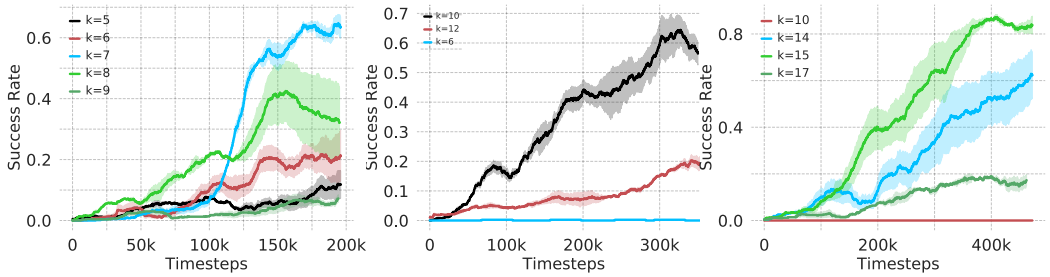
(a) Pick and place (b) Hollow (c) Kitchen

Figure 9: **Effect of the Lower-Level Optimality Weight.** Success rates for different values of the lower-level optimality weight η in equation 12 across Pick and Place, Hollow, and Franka Kitchen. Solid lines represent the mean and shaded regions denote the standard deviation across five random seeds.



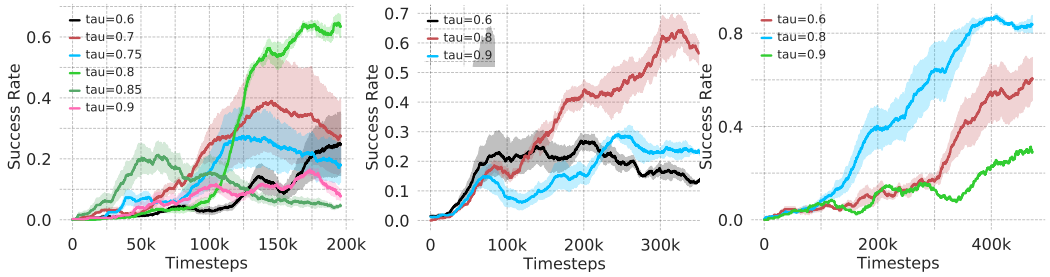
(a) Pick and place (b) Hollow (c) Kitchen

Figure 10: **Discount Factor (γ) Ablation.** This figure illustrates the success rates for various discount factor γ values across three sparse-reward robotic manipulation tasks, where the solid lines represent the mean, and the shaded areas denote the standard deviation across 5 different seeds.



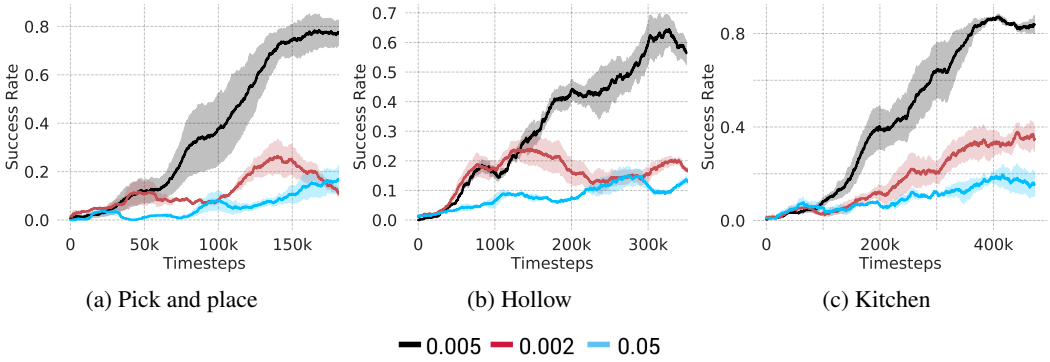
(a) Pick and place (b) Hollow (c) Kitchen

Figure 11: k **Ablation.** This figure illustrates the success rates for various k values across three sparse-reward robotic manipulation tasks, where the solid lines represent the mean, and the shaded areas denote the standard deviation across 5 different seeds.



(a) Pick and place (b) Hollow (c) Kitchen

Figure 12: **Polyak Averaging** τ **Ablation.** This figure illustrates the success rates for various τ values across three sparse-reward robotic manipulation tasks, where the solid lines represent the mean, and the shaded areas denote the standard deviation across 5 different seeds.

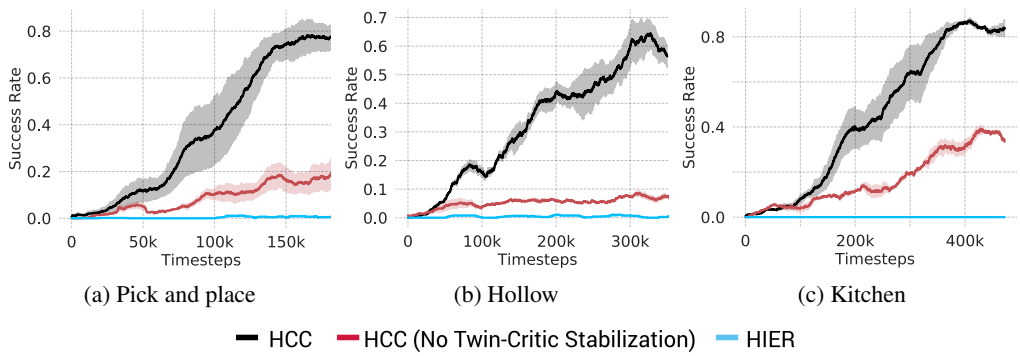


(a) Pick and place (b) Hollow (c) Kitchen

Figure 13: **Imitation Weight** λ **Ablation.** This figure illustrates the success rates for various λ values across three sparse-reward robotic manipulation tasks, where the solid lines represent the mean, and the shaded areas denote the standard deviation across 5 different seeds.

A.5 Implementation Details

All experiments were conducted using computing nodes configured with Intel Core i7 processors, 48GB of RAM, and Nvidia GeForce GTX 1080 GPUs. We track and report the total timesteps required for convergence in each experiment. For the network architecture, both the actor and critic models utilize three-layer fully connected neural networks with a width of 512 neurons per layer.



(a) Pick and place (b) Hollow (c) Kitchen

Figure 14: **Inherent Stability of the Contrastive Formulation.** In this figure, we compare the performance of full HCC (Black), HCC without any stabilization (no twin-critic and no Polyak averaging), and the standard HRL baseline (HIER). Even without these stabilization mechanisms, HCC formulation (Red) still outperforms HIER (Blue) on all environments. This empirically demonstrates that reframing high-level control as future-State density estimation is more robust to shifting transition dynamics than standard TD bootstrapping. The subsequent performance gain to full HCC confirms that the twin-classifier and Polyak averaging mechanisms effectively resolve the secondary moving-target problem associated with the classifier's shifting target distribution.

The experimental suite includes three distinct domains. In the **pick-and-place** task requires the same 7-DoF arm to locate a square block, grasp it, and transport it to a specified goal. In the **hollow** environment, the agent has to pick a hollow block on the table and place the grasped block into a pole such that the pole goes through the hollow block. Finally, the **kitchen** task involves a 9-DoF Franka robot executing a complex manipulation sequence, specifically opening a microwave door, and turning on the stove.

To ensure rigorous benchmarking, we standardize hyperparameters across all baselines. Common settings include the optimizer configuration, SAC implementation specifics, and network depth/width. The specific hyperparameters used are listed below:

Table 2: Hyperparameter settings for the hierarchical policy networks.

Parameter	**Value**	**Description**
`activation`	tanh	Activation for hierarchical policies
`layers`	3	Number of layers (critic/actor)
`hidden`	512	Neurons per hidden layer
`Q_lr`	0.001	Critic learning rate
`pi_lr`	0.001	Actor learning rate
`buffer_size`	10^7	Experience replay buffer capacity
`clip_obs`	200	Observation clipping threshold
`n_cycles`	1	Cycles per epoch
`n_batches`	10	Training batches per cycle
`batch_size`	1024	Training batch size
`random_eps`	0.2	Random action probability
`alpha` (α)	0.05	Entropy weight (SAC)
`noise_eps`	0.05	Gaussian action noise std
`norm_eps`	0.01	Epsilon for observation norm
`norm_clip`	5	Normalized observation clip
`adam_beta1`	0.9	Adam optimizer β_1
`adam_beta2`	0.999	Adam optimizer β_2

A.6 Environment Specifications

This section outlines the State space definitions and dynamics for the evaluated tasks.

Pick-and-Place and Hollow Tasks. For manipulation tasks involving object interaction, the State is defined as $[dx, o, q, e]$. Here, dx represents the gripper position, o is the object's table position, q is the relative vector between the block and gripper, and e encodes the linear and angular velocities for both entities. Dimensions for dx, o, q, and e are 3, 3, 3, and 11, respectively.

Input to the high-level policy is the concatenation $[dx, o, q, e, g]$, while the low-level policy observes $[dx, o, q, e, s_g]$. The primitive action a is a 4-dimensional vector controlling position offsets and gripper State. During training episodes, the block is initialized at random positions on the table, while the goal is generated at a fixed height above the workspace.

Franka Kitchen. We utilize the standard Franka Kitchen environment as defined in D4RL Fu et al. (2020). In this domain, the agent controls a Franka robot to sequentially solve multi-stage subtasks to achieve a terminal configuration.

A.7 Qualitative visualizations

We provide qualitative visualizations for all the environments:

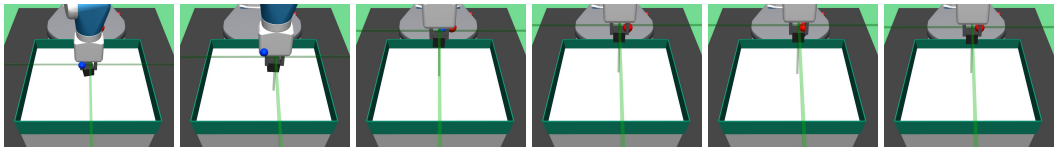


Figure 15: **Successful visualization**: The visualization is a successful attempt at performing pick and place task.

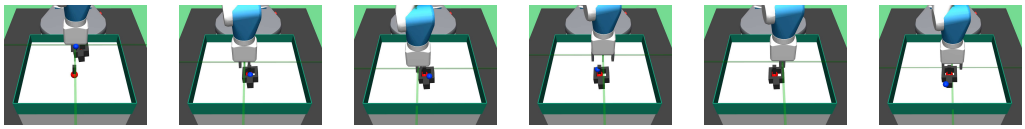


Figure 16: **Hollow task visualization**: The visualization is a successful attempt at performing hollow task

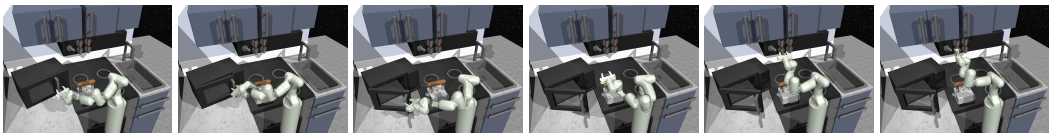


Figure 17: **Successful visualization**: The visualization is a successful attempt at performing kitchen task.