Offline Goal-Conditioned RL with Latent States as Actions

Anonymous Authors[1]

Abstract

In the same way that unsupervised pre-training has become the bedrock for computer vision and NLP, goal-conditioned RL might provide a similar strategy for making use of vast quantities of unlabeled (reward-free) data. However, building effective algorithms for goal-conditioned RL, ones that can learn directly from offline data, is challenging because it is hard to accurately estimate the exact state value of reaching faraway goals. Nonetheless, goal-reaching problems exhibit structure – reaching a distant goal entails visiting some closer states (or representations thereof) first. Importantly, it is easier to assess the effect of actions on getting to these closer states. Based on this idea, we propose a hierarchical algorithm for goal-conditioned RL from offline data. Using one action-free value function, we learn two policies that allow us to exploit this structure: a high-level policy that predicts (a representation of) a waypoint, and a low-level policy that predicts the action for reaching this waypoint. Through analysis and didactic examples, we show how this hierarchical decomposition makes our method robust to noise in the estimated value function. We then apply our method to offline goal-reaching benchmarks, showing that our method can solve long-horizon tasks that stymie prior methods, can scale to high-dimensional image observations, and can readily make use of action-free data.

1. Introduction

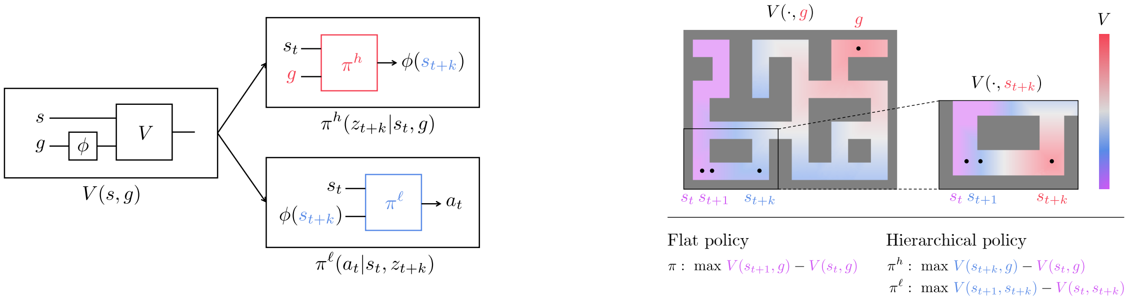
Many of the most successful machine learning systems for computer vision (Chen et al., 2020; He et al., 2022) and NLP (Devlin et al., 2019; Brown et al., 2020) leverage large amounts of unlabeled or weakly-labeled data. In the reinforcement learning (RL) setting, offline goal-conditioned RL provides a way of making use of similar quantities of unlabeled data; the offline setting (Lange et al., 2012; Levine et al., 2020) means that we can learn from passively observed data, and the goal-conditioned setting (Kaelbling, 1993; Schaul et al., 2015) means that we can learn from reward-free data (no need for reward labels). However, goal-conditioned RL poses major challenges. First, learning an accurate goal-conditioned value function for any state and goal pairs is challenging when considering very broad and long-horizon goal-reaching tasks. This often results in a noisy value function and thus potentially an erroneous policy. Second, while the offline setting unlocks the potential for using previously collected data, it is not straightforward to incorporate vast quantities of existing action-free, video data into standard RL methods. In this work, we aim to address these challenges by developing an effective offline, goal-conditioned RL method that can also readily make use of action-free data.

One straightforward approach to offline goal-conditioned RL is to first train a goal-conditioned value function and then learn a policy that leads to states with high values. However, the learned value function can often provide poor signals for selecting actions to reach distant goals. Intuitively, the value function depends on how far away the goal will be after taking a particular action. However, when the goal is far away, the optimal action may be only slightly better than suboptimal actions; for example, a move in the wrong direction can simply be corrected at the next time step, leading to a small relative increase in distance. Thus, when the value function is learned imperfectly and has small errors, these errors can drown out the signal for distant goals, potentially leading to an erroneous policy. This issue is further exacerbated with the offline RL setting, as erroneous predictions from the value function are not corrected when those actions are taken and their consequences observed.

To learn from noisy or inaccurate value functions, we will separate policy extraction into two levels. We first train a goal-conditioned value function from offline data with implicit Q-learning (IQL) (Kostrikov et al., 2022) and then we extract two-level policies from it. Our high-level policy produces intermediate waypoint states as temporally extended actions. Because predicting high-dimensional states can be challenging, we will propose a method that only requires the high-level policy to product *representations* of the way-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML). Do not distribute.



(a) **Three components of HIQL.**

(b) **Hierarchical policies get clearer learning signals.**

Figure 1. We train a value function parameterized as $V(s, \phi(g))$ and use $\phi(g)$ as a representation function. The high-level policy predicts the waypoint representation $z_{t+k} = \phi(s_{t+k})$, and the low-level policy takes the waypoint representation as input to produce actions. Both policies are extracted from the *same* value function.

points, with the representations learned end-to-end from the value function. Our low-level policy takes these waypoint representations as input and produces actions to reach the waypoint (Figure 1a). Although we extract both policies from the *same* value function, this hierarchical decomposition enables the value function to provide clearer learning signals for both policies (Figure 1b). For the high-level policy, the value difference between various waypoints is much larger than that between different low-level actions. For the low-level policy, the value difference between actions becomes relatively larger because the low-level policy only needs to reach nearby waypoints. Importantly, the value function and high-level policy do not require action labels, so this hierarchical scheme provides a way to leverage a potentially large amount of passive, action-free data. Training the low-level policy does require some data labeled with actions.

To summarize, our main contribution in this paper is to propose **Hierarchical Implicit Q-Learning** (**HIQL**), a hierarchical method for offline goal-conditioned RL. HIQL extracts all the necessary components—a representation function, a high-level policy, and a low-level policy—from a single goal-conditioned value function. Through our experiments on four types of state-based and pixel-based offline goal-conditioned RL benchmarks, we demonstrate that HIQL significantly outperforms previous offline goal-conditioned RL methods, especially in complex, long-horizon tasks.

2. Related Work

Our method draws on concepts from offline RL (Lange et al., 2012; Levine et al., 2020), goal-conditioned RL (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017), and hierarchical RL (Sutton et al., 1999; Stolle & Precup, 2002; Bacon et al., 2017; Machado et al., 2017; Wulfmeier et al., 2021; Salter et al., 2022), providing a way to effectively train general-purpose goal-conditioned policies from previously collected offline data. Prior work on goal-conditioned RL has introduced algorithms based on a variety of techniques, such as hindsight relabeling (Andrychowicz et al., 2017; Pong et al., 2018; Fang et al., 2019; Levy et al., 2019; Li et al., 2020; Chebotar et al., 2021; Yang et al., 2022), contrastive learning (Eysenbach et al., 2021; Zhang et al., 2022; Eysenbach et al., 2022), and state-occupancy matching (Ma et al., 2022; Durugkar et al., 2021).

However, directly solving goal-reaching tasks is often challenging in complex, long-horizon environments (Nachum et al., 2018; Levy et al., 2019; Gupta et al., 2019). To address this issue, several goal-conditioned RL methods have been proposed based on hierarchical RL (Schmidhuber, 1991; Dayan & Hinton, 1992; Kulkarni et al., 2016; Vezhnevets et al., 2017; Nachum et al., 2018; 2019; Levy et al., 2019; Zhang et al., 2020; Chane-Sane et al., 2021) or graph-based subgoal planning (Savinov et al., 2018; Eysenbach et al., 2019; Huang et al., 2019; Nasiriany et al., 2019; Zhang et al., 2021; Hoang et al., 2021; Kim et al., 2021; 2023). Like these prior methods, our method will use higher-level subgoals in a hierarchical policy structure, but we will focus on solving goal-reaching tasks from *offline* data. We use a value-based offline RL algorithm (Kostrikov et al., 2022) to compute the shortest distances between any pairs of states in the dataset, which allows us to simply *extract* the hierarchical policies in a decoupled manner with no need for complex graph-based planning procedures.

Our method is most closely related to previous works on hierarchical offline skill extraction and hierarchical offline (goal-conditioned) RL. Offline skill extraction methods (Krishnan et al., 2017; Pertsch et al., 2020; Ajay et al., 2021; Shi et al., 2022; Jiang et al., 2023; Rosete-Beas et al., 2022) encode trajectory segments into a latent skill space, and learn to combine these skills to solve downstream tasks. The primary challenge in this setting is deciding how trajectories should be decomposed hierarchically, which can be sidestepped in our goal-conditioned setting since subgoals provide a natural decomposition. Amongst goal-conditioned approaches, hierarchical imitation learning (Lynch et al., 2019; Gupta et al., 2019) jointly learns waypoints and low-level controllers from optimal demonstrations. These methods have two drawbacks: they predict waypoints in the raw observation space, and they require expert trajectories; our observation is that a value function can alleviate both challenges, as it provides a way to use sub-optimal data and stitch across trajectories, as well as providing a latent goal representation in which waypoints may be predicted. Another class of methods plans through a graph or model to generate subgoals (Shah et al., 2021; Fang et al., 2022a;b; Li et al., 2022); our method simply extracts all levels of the hierarchy from a single unified value function, avoiding the high computational overhead of planning. Finally, our method is similar to POR (Xu et al., 2022), which predicts the immediate next state as a waypoint; this can be seen as one extreme of our method without representations, although we show that more long-horizon waypoint prediction can be advantageous both in theory and practice.

3. Preliminaries

Problem setting. We consider the problem of offline goal-conditioned RL, defined by a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, p, r)$ and a dataset $\mathcal{D}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\mu \in \mathcal{P}(\mathcal{S})$ denotes an initial state distribution, $p \in \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ denotes a transition dynamics distribution, and $r(s, g)$ denotes a goal-conditioned reward function. The dataset $\mathcal{D}$ consists of trajectories $\tau = (s_0, a_0, s_1, a_1, \dots, s_T)$. In some experiments, we assume that we have an additional action-free dataset $\mathcal{D}_\mathcal{S}$ that consists of state-only trajectories $\tau_s = (s_0, s_1, \dots, s_T)$. Unlike some prior work (Andrychowicz et al., 2017; Nachum et al., 2018; Huang et al., 2019; Zhang et al., 2021; Kim et al., 2023), we assume that the goal space $\mathcal{G}$ is the same as the state space (*i.e.*, $\mathcal{G} = \mathcal{S}$). Our goal is to learn from $\mathcal{D} \cup \mathcal{D}_\mathcal{S}$ an optimal goal-conditioned policy $\pi(a|s, g)$ that maximizes $J(\pi) = \mathbb{E}_{g \sim p(g), \tau \sim p^\pi(\tau)}[\sum_{t=0}^{T} \gamma^t r(s_t, g)]$ with $p^\pi(\tau) = \mu(s_0) \prod_{t=0}^{T-1} \pi(a_t \mid s_t, g) p(s_{t+1} \mid s_t, a_t)$, where γ is a discount factor and $p(g)$ is a goal distribution.

Implicit Q-learning (IQL). One of the main challenges with offline RL is to prevent exploitation of out-of-distribution actions (Levine et al., 2020), as we cannot correct erroneous policies and values via environment interactions, unlike in online RL. To tackle this issue, Kostrikov et al. (2022) proposed implicit Q-learning (IQL), which avoids querying out-of-sample actions by converting the $\max$ operator in the Bellman optimal equation into expectile regression. Specifically, IQL trains an action-value function $Q_{\theta_Q}(s, a)$ and a state-value function $V_{\theta_V}(s)$ with

$$\mathcal{L}_V(\theta_V) = \mathbb{E}[L_2^\tau(Q_{\bar{\theta}_Q}(s, a) - V_{\theta_V}(s))], \tag{1}$$

$$\mathcal{L}_Q(\theta_Q) = \mathbb{E}[(r_{\text{task}}(s, a) + \gamma V_{\theta_V}(s') - Q_{\theta_Q}(s, a))^2], \tag{2}$$

where $r_{\text{task}}(s, a)$ denotes the task reward function, $\bar{\theta}_Q$ denotes the parameters of the target Q network (Mnih et al., 2013), and L_2^τ is the expectile loss with a parameter $\tau \in [0.5, 1)$: $L_2^\tau(x) = |\tau - \mathbb{1}(x < 0)|x^2$. Intuitively, expectile regression can be interpreted as an asymmetric square loss that penalizes positive values more than negative ones. As a result, when τ tends to 1, $V_{\theta_V}(s)$ gets closer to $\max_a Q_{\bar{\theta}_Q}(s, a)$ (Equation (1)). Thus, we can use the value function to estimate the TD target $(r_{\text{task}}(s, a) + \gamma \max_{a'} Q_{\bar{\theta}_Q}(s', a'))$ as $(r_{\text{task}}(s, a) + \gamma V_{\theta_V}(s'))$ without having to sample actions a'.

After training the value function with Equations (1) and (2), IQL extracts the policy with advantage-weighted regression (AWR) (Peters & Schaal, 2007; Neumann & Peters, 2008; Peters et al., 2010; Peng et al., 2019; Nair et al., 2020; Wang et al., 2020):

$$J_\pi(\theta_\pi) = \mathbb{E}[\exp(\beta \cdot (Q_{\bar{\theta}_Q}(s, a) - V_{\theta_V}(s))) \log \pi_{\theta_\pi}(a \mid s)], \tag{3}$$

where $\beta \in \mathbb{R}_0^+$ denotes an inverse temperature parameter. Intuitively, Equation (3) encourages the policy to select actions that lead to large Q values while not deviating far from the data collection policy (Peng et al., 2019).

Action-free goal-conditioned IQL. The original IQL method described above requires both reward and action labels in the offline data to train the value functions by Equations (1) and (2). However, in real-world scenarios, offline data might not contain task information or action labels, as in the case of task-agnostic demonstrations or videos. As such, we focus on the setting of offline goal-conditioned RL, which does not require task rewards, and provides us with a way to incorporate state-only trajectories into value learning. We can use the following action-free variant (Xu et al., 2022; Ghosh et al., 2023) of IQL to learn an offline goal-conditioned value function $V_{\theta_V}(s, g)$:

$$\mathcal{L}_V(\theta_V) = \mathbb{E}[L_2^\tau(r(s, g) + \gamma V_{\bar{\theta}_V}(s', g) - V_{\theta_V}(s, g))]. \tag{4}$$

Unlike Equations (1) and (2), this objective does not require actions when fitting the value function, as it directly takes backups from the values of the next states.

Action-labeled data is only needed when extracting the policy. With the goal-conditioned value function learned by Equation (4), we can extract the policy with the following variant of AWR:

$$J_\pi(\theta_\pi) = \mathbb{E}[\exp(\beta \cdot A(s, a, g)) \log \pi_{\theta_\pi}(a \mid s, g)], \tag{5}$$

where we approximate $A(s, a, g)$ as $\gamma V_{\theta_V}(s', g) + r(s, g) - V_{\theta_V}(s, g)$. Intuitively, Equation (5) encourages the policy to select the actions that lead to the states having high values. With this action-free variant of IQL, we can train an optimal goal-conditioned value function only using action-free data and extract the policy from action-labeled data that may be different from the passive dataset.

We note that this action-free variant of IQL is unbiased when the environment dynamics are deterministic (Ghosh et al., 2023), but it may overestimate values in stochastic environments. This deterministic environment assumption is inevitable for learning an unbiased value function solely from state trajectories. The reason is subtle but important: in stochastic environments, it is impossible to tell whether a good outcome was caused by taking a good action or because of noise in the environment. As a result, applying action-free IQL to stochastic environments will typically result in overestimating the value function, implicitly assuming that all noise is controllable. While we will build our method upon Equation (4) in this work for simplicity, in line with many prior works on offline RL that employ similar assumptions (Ghosh et al., 2021; Chen et al., 2021; Janner et al., 2021; 2022; Xu et al., 2022; Wang et al., 2023; Ghosh et al., 2023), we believe correctly handling stochastic environments with advanced techniques (*e.g.*, by identifying controllable parts of the environment (Yang et al., 2023; Villaflor et al., 2022)) is an interesting direction for future work.

4. Hierarchical policy structure for offline goal-conditioned RL

Goal-conditioned offline RL provides a general framework for learning flexible policies from data, but the goal-conditioned setting also presents an especially difficult multi-task learning problem for RL algorithms, particularly for long-horizon tasks where the goal is far away. In Section 4.1, we discuss some possible reasons for this difficulty, from the perspective of the "signal-to-noise" ratio in the learned goal-conditioned value function. We then propose hierarchical policy extraction as a solution (Section 4.2) and compare the performances of hierarchical and flat policies in a didactic environment, based on our theoretical analysis (Section 4.3).

4.1. Motivation: why non-hierarchical policies might struggle

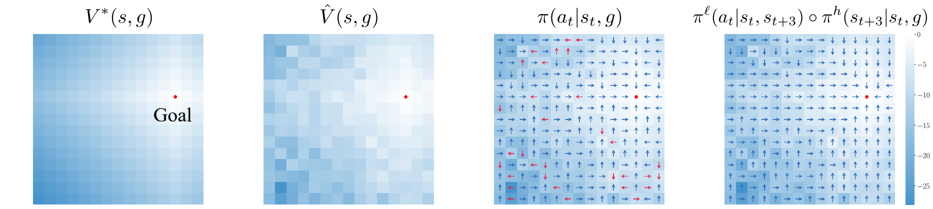
One common strategy in offline RL is to first fit a value function and then extract a policy that points in the direction of high values (Fujimoto et al., 2019; Peng et al., 2019; Wu et al., 2019; Kumar et al., 2019; Nair et al., 2020; Ghasemipour et al., 2021; Brandfonbrener et al., 2021; An et al., 2021; Kostrikov et al., 2022; Yang et al., 2022; Xu et al., 2022; Garg et al., 2023; Xu et al., 2023). This strategy can be directly applied to offline goal-conditioned RL by learning a goal-conditioned policy $\pi(a \mid s_t, g)$ that aims to maximize the learned goal-conditioned value function $V(s_{t+1}, g)$, as in Equation (5). However, when the goal g is far away from the state s, the learned goal-conditioned value function may not provide clear signals for the flat, non-hierarchical policy. There are two reasons for this failure. First, the differences between the values of different next states ($V(s_{t+1}, g)$) may be small, as incorrect primitive actions may be fixed in subsequent steps, causing only relatively minor costs. Second, these small differences can be further overshadowed by the noise present in the learned value function, especially when the goal is distant from the current state, in which case the magnitude of the goal-conditioned value (and thus its noise) is large. In other words, the "signal-to-noise" ratio in the very next values $V(s_{t+1}, g)$ can be small, not providing sufficiently clear learning signals for the flat policy. Figure 2 illustrates this problem. Figure 2a shows the ground-truth optimal value function $V^*(s, g)$ for a given goal at each state, which can guide the agent to reach the goal. However, when noise is present in the learned value function $\hat{V}(s, g)$ (Figure 2b), the flat policy $\pi(a \mid s, g)$ becomes erroneous, especially in states far from the goal (Figure 2c).

4.2. Our hierarchical policy structure

To address this issue, our main idea in this work, which we present fully in Section 5, is to separate policy extraction into two levels. Instead of directly learning a single, flat, goal-conditioned policy $\pi(a \mid s_t, g)$ that aims to maximize $V(s_{t+1}, g)$, we extract both a high-level policy $\pi^h(s_{t+k} \mid s_t, g)$ and a low-level policy $\pi^\ell(a \mid s_t, s_{t+k})$, each with its own maximization objective: $V(s_{t+k}, g)$ and $V(s_{t+1}, s_{t+k})$, respectively. Here, s_{t+k} can be viewed as the *waypoint* or subgoal. The high-level policy outputs intermediate waypoint states that are k steps away from s, while the low-level policy produces primitive actions to reach these waypoints. Although we extract both policies from the *same* learned value function in this way, this hierarchical scheme provides clearer learning signals for both policies. Intuitively, the high-level policy receives more reliable learning signals because different waypoints lead to more dissimilar values than primitive actions. The low-level policy also gets relatively clear signals since it queries the value function with only nearby states, for which the value function is relatively accurate (Figure 1b). As a result, the overall hierarchical policy can be more robust to the noise and thus can improve the accuracy (Figure 2d).

4.3. Didactic example: our hierarchical policy mitigates the signal-to-noise ratio challenge

To further understand the benefits of hierarchical policies, we study a toy example with one-dimensional state space (Figure 3). In this environment, the agent can move one unit to the left or right at each time step. The agent gets a reward of 0 when it reaches the goal; otherwise, it always gets -1. The optimal goal-conditioned value function is hence given as $V^*(s, g) = -|s - g|$. We assume that the noise in the



(a) Optimal value function (b) Noisy value function (c) Flat policy (d) Hierarchical policy

Figure 2. **Hierarchies allow us to better make use of noisy value estimates.** *(a)* In this gridworld environment, the optimal value function predicts higher values for states s that are closer to the goal g (•). *(b, c)* However, a noisy value function results in selecting incorrect actions (→). *(d)* Our method uses this *same* noisy value function to first predict an intermediate waypoint, and then select an action for reaching this waypoint. Actions selected in this way correctly lead to the goal.

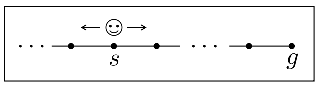


Figure 3. **1-D toy environment.**

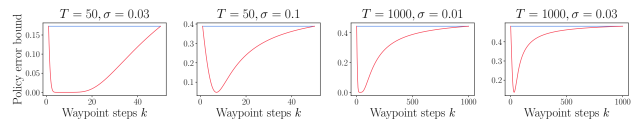


Figure 4. **Comparison of policy errors in flat vs. hierarchical policies in didactic environments.** The hierarchical policy, with an appropriate waypoint step, often yields significantly lower errors than the flat policy.

learned value function $\hat{V}(s,g)$ is proportional to the optimal value: *i.e.*, $\hat{V}(s,g) = V^*(s,g) + \sigma z_{s,g} V^*(s,g)$, where $z_{s,g}$ is sampled independently from the standard normal distribution and σ is its standard deviation. This indicates that as the goal becomes more distant, the noise generally increases, a trend we observed in our experiments (see Figure 6).

In this scenario, we compare the probabilities of choosing incorrect actions under the flat and hierarchical policies. We assume that the distance between s and g is T (*i.e.*, $g = s + T$ and $T > 1$). Both the flat policy and the low-level policy of the hierarchical approach consider the goal-conditioned values at $s \pm 1$. The high-level policy evaluates the values at $s \pm k$, using k-step away waypoints. For the hierarchical approach, we query both the high- and low-level policies at every step. Given these settings, we can bound the error probabilities of both approaches as follows:

Proposition 4.1. *In the environment described in Figure 3, the probability of the flat policy π selecting an incorrect action is given as $\mathcal{E}(\pi) = \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{T^2+1}}\right)$ and the probability of the hierarchical policy $\pi^\ell \circ \pi^h$ selecting an incorrect action is bounded as $\mathcal{E}(\pi^\ell \circ \pi^h) \leq \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{(T/k)^2+1}}\right) + \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{k^2+1}}\right)$, where Φ denotes the cumulative distribution function of the standard normal distribution, $\Phi(x) = \mathbb{P}[z \leq x] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} \mathrm{d}t$.*

The proof can be found in Appendix. We first note that each of the error terms in the hierarchical policy bound is always no larger than the error in the flat policy, implying that both the high- and low-level policies are more accurate than the flat policy. To compare the total errors, $\mathcal{E}(\pi)$ and $\mathcal{E}(\pi^\ell \circ \pi^h)$, we perform a numerical analysis. Figure 4 shows the hierarchical policy's error bound for varying waypoint steps in five different (T, σ) settings. The results indicate that the flat policy's error can be significantly reduced by employing a hierarchical policy with an appropriate choice of k, suggesting that splitting policy extraction into two levels can be beneficial.

5. Hierarchical Implicit Q-Learning (HIQL)

Based on the hierarchical policy structure in Section 4, we now present a practical algorithm, which we call **Hierarchical Implicit Q-Learning** (**HIQL**), to extract hierarchical policies that are robust to the noise present in the learned goal-conditioned value function. We first explain how to train a waypoint policy (Section 5.1) and then extend this policy to predict representations (learned via the value function), which will enable HIQL to scale to image-based environments (Section 5.2).

5.1. Hierarchical policy extraction

As motivated in Section 4.2, we split policy learning into two levels, with a high-level policy generating intermediate waypoints and a low-level policy producing primitive actions to reach the waypoints. In this way, the learned goal-conditioned value function can provide clearer signals for both policies, effectively reducing the total policy error. Our method, HIQL, extracts the hierarchical policies from the *same* value function learned by action-free IQL (Equation (4)) using AWR-style objectives. Specifically, HIQL trains both a high-level policy $\pi^h_{\theta_h}(s_{t+k} \mid s_t, g)$, which produces optimal k-step waypoints s_{t+k}, and a low-level policy $\pi^\ell_{\theta_\ell}(a \mid s_t, s_{t+k})$, which outputs primitive actions, with

$$J_{\pi^h}(\theta_h) = \mathbb{E}[\exp(\beta \cdot \tilde{A}^h(s_t, s_{t+k}, g)) \log \pi^h_{\theta_h}(s_{t+k} \mid s_t, g)], \tag{6}$$

$$J_{\pi^\ell}(\theta_\ell) = \mathbb{E}[\exp(\beta \cdot \tilde{A}^\ell(s_t, a_t, g)) \log \pi^\ell_{\theta_\ell}(a_t \mid s_t, s_{t+k})], \tag{7}$$

where β denotes the inverse temperature hyperparameter and we approximate $\tilde{A}^h(s_t, s_{t+k}, g)$ as $V_{\theta_V}(s_{t+k}, g) - V_{\theta_V}(s_t, g)$ and $\tilde{A}^\ell(s_t, a_t, s_{t+k})$ as $V_{\theta_V}(s_{t+1}, s_{t+k}) - V_{\theta_V}(s_t, s_{t+k})$. We do not include discount factors and rewards in these advantage estimates for simplicity, as they can be ignored or subsumed into the temperature β given our goal-sampling strategy described in Appendix (see Appendix for further discussion). Similarly to vanilla AWR (Equation (5)), our high-level objective (Equation (6)) performs a weighted regression over waypoints to reach the goal, and the low-level objective (Equation (7)) carries out a weighted regression over primitive actions to reach the waypoints.

We note that Equation (6) and Equation (7) are completely separated from one another, and only the low-level objective requires action labels. As a result, we can leverage action-free data for both the value function and high-level policy of HIQL, by further training them with a potentially large amount of additional passive data. Moreover, the low-level policy is relatively easy to learn compared to the other components, as it only needs to reach local waypoints without the need for learning the complete global structure. This enables HIQL to work well even with a limited amount of action information, as we will demonstrate in Section 6.4.

5.2. Representations for waypoints

In high-dimensional domains, such as pixel-based environments, directly predicting waypoint states can be prohibitive or infeasible for the high-level policy. To resolve this issue, we incorporate representation learning into HIQL, letting the high-level policy produce more compact *representations* of waypoints. While one can employ existing action-free representation learning methods (Seo et al., 2022; Nair et al., 2022; Ma et al., 2023; Ghosh et al., 2023) to learn state representations, HIQL simply uses an intermediate layer of the value function as a goal representation, which can be proven to be sufficient for control. Specifically, we parameterize the goal-conditioned value function $V(s, g)$ with $V(s, \phi(g))$, and use $\phi(g)$ as the representation of the goal. Using this representation, the high-level policy $\pi^h(z_{t+k} \mid s_{t+k}, g)$ produces $z_{t+k} = \phi(s_{t+k})$ instead of s_{t+k}, which the low-level policy $\pi^\ell(a \mid s_t, z_{t+k})$ takes as input to output actions (Figure 1a). In this way, we can simply learn compact goal representations that are sufficient for control with no separate training objectives or components. We provide the algorithm pseudocode and full implementation details in Appendix and present the sufficiency result below (see Appendix for the proof).

Proposition 5.1 (Goal representations from the value function are sufficient for action selection)**.** *Let $V^*(s, g)$ be the value function for the optimal reward-maximizing policy $\pi^*(a \mid s, g)$ in a deterministic MDP. Let a representation function $\phi(g)$ be given. If this same value function can be represented in terms of goal representations $\phi(g)$, then the reward-maximizing policy can also be represented in terms of goal representations $\phi(g)$:*

$$\exists\, V_\phi(s, \phi(g)) \text{ s.t. } V_\phi(s, \phi(g)) = V^*(s, g) \text{ for all } s, g \implies$$
$$\exists\, \pi_\phi(a \mid s, \phi(g)) \text{ s.t. } \pi_\phi(a \mid s, \phi(g)) = \pi^*(a \mid s, g) \text{ for all } s, g.$$

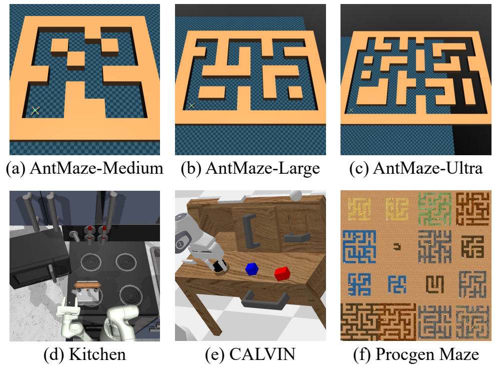


Figure 5. **Benchmark environments.**

6. Experiments

Our experiments will use six offline goal-conditioned tasks, aiming to answer the following questions:

1. How well does HIQL perform on a variety of goal-conditioned tasks, compared to prior methods?
2. Can HIQL solve image-based tasks, and are goal representations important for good performance?
3. Can HIQL utilize action-free data to accelerate learning?
4. Does HIQL mitigate policy errors caused by noisy and imperfect value functions in practice?

6.1. Experimental setup

We first describe our evaluation environments, shown in Figure 5. **AntMaze** (Todorov et al., 2012; Brockman et al., 2016) is a class of challenging long-horizon navigation tasks, where the goal is to control an 8-DoF Ant robot to reach a given goal location from the initial position. We use the four medium and large maze datasets from the original D4RL benchmark (Fu et al., 2020). While the large mazes already present a significant challenge for long-horizon reasoning, we also include two even larger mazes (AntMaze-Ultra) proposed by Jiang et al. (2023). **Kitchen** (Gupta et al., 2019) is a long-horizon manipulation domain, in which the goal is to complete four subtasks (*e.g.*, open the microwave or move the kettle) with a 9-DoF Franka robot. We employ two undirected datasets ('-partial' and '-mixed') from the D4RL benchmark (Fu et al., 2020). **CALVIN** (Mees et al., 2022), another long-horizon manipulation environment, also features four target subtasks similar to Kitchen. However, the dataset accompanying CALVIN (Shi et al., 2022) consists of a much larger number of task-agnostic trajectories

Table 1. **Evaluating HIQL on offline goal-conditioned RL.** HIQL mostly outperforms six baselines on a variety of benchmark tasks, including on different types of data. 'gc-' denotes goal-conditioned variants. We show the standard deviations across 8 random seeds and refer to Appendix for the full training curves. Baselines: GCBC (Ghosh et al., 2021), HGCBC (Gupta et al., 2019), IQL (Kostrikov et al., 2022), POR (Xu et al., 2022), TAP (Jiang et al., 2023), TT (Janner et al., 2021).

Task	**GCBC**	**HGCBC**	**IQL**	**POR**	**TAP**	**TT**	**HIQL (ours)**	**HIQL** (w/o repr.)
gc-antmaze-medium-diverse	67.3 ±10.1	71.6 ±8.9	63.5 ±14.6	74.8 ±11.9	85.0	**100.0**	86.8 ±4.6	89.9 ±3.5
gc-antmaze-medium-play	71.9 ±16.2	66.3 ±9.2	70.9 ±11.2	71.4 ±10.9	78.0	**93.3**	84.1 ±10.8	87.0 ±8.4
gc-antmaze-large-diverse	20.2 ±9.1	63.9 ±10.4	50.7 ±18.8	49.0 ±17.2	82.0	60.0	**88.2** ±5.3	87.3 ±3.7
gc-antmaze-large-play	23.1 ±15.6	64.7 ±14.5	56.5 ±14.4	63.2 ±16.1	74.0	66.7	**86.1** ±7.5	81.2 ±6.6
gc-antmaze-ultra-diverse	14.4 ±9.7	39.4 ±20.6	21.6 ±15.2	29.8 ±13.6	26.0	33.3	**52.9** ±17.4	52.6 ±8.7
gc-antmaze-ultra-play	20.7 ±9.7	38.2 ±18.1	29.8 ±12.4	31.0 ±19.4	22.0	20.0	39.2 ±14.8	**56.0** ±12.4
gc-kitchen-partial	38.5 ±11.8	32.0 ±16.7	39.2 ±13.5	18.4 ±14.3	-	-	**65.0** ±9.2	46.3 ±8.6
gc-kitchen-mixed	46.7 ±20.1	46.8 ±17.6	51.3 ±12.8	27.9 ±17.9	-	-	**67.7** ±6.8	36.8 ±20.1
gc-calvin	17.3 ±14.8	3.1 ±8.8	7.8 ±17.6	12.4 ±18.6	-	-	**43.8** ±39.5	23.4 ±27.1

from 34 different subtasks, which makes it challenging for the agent to learn relevant behaviors for the goal. **Procgen Maze** (Cobbe et al., 2020) is an image-based maze navigation environment. We train agents on an offline dataset consisting of 500 or 1000 different maze levels with a variety of sizes, colors, and difficulties, and test them on both the same and different sets of levels to evaluate their generalization capabilities. To make these benchmark environments goal-conditioned, during training, we replace the original rewards with a sparse goal-conditioned reward function, $r(s, g) = 0$ (if $s = g$), -1 (otherwise).

We compare the performance of HIQL with six previous behavioral cloning and offline RL methods. For behavioral cloning methods, we consider flat goal-conditioned behavioral cloning (GCBC) (Ding et al., 2019; Ghosh et al., 2021) and hierarchical goal-conditioned behavioral cloning (HGCBC) with two-level policies (Lynch et al., 2019; Gupta et al., 2019). For offline goal-conditioned RL methods, we evaluate a goal-conditioned variant of IQL (Kostrikov et al., 2022) (Section 3), which does not use hierarchy, and POR (Xu et al., 2022), which uses hierarchy but does not use temporal abstraction (*i.e.*, similar to $k = 1$ in HIQL) nor representation learning. In AntMaze, we additionally compare HIQL with two model-based approaches that studied this domain in prior work: Trajectory Transformer (TT) (Janner et al., 2021), which models entire trajectories with a Transformer (Vaswani et al., 2017), and TAP (Jiang et al., 2023), which encodes trajectory segments with VQ-VAE (van den Oord et al., 2017) and performs model-based planning over latent vectors in a hierarchical manner. We use the performance reported by Jiang et al. (2023) for comparisons with TT and TAP. In our experiments, we use 8 random seeds and represent 95% confidence intervals with shaded regions (in figures) or standard deviations (in tables), unless otherwise stated. We provide full details of environments and baselines in Appendix.

Table 2. **Evaluating HIQL on pixel-based Procgen Maze.** HIQL scales to high-dimensional pixel-based environments by using latent waypoint representations. HIQL achieves the best performance on both train and test maze levels. We refer to Appendix for the full training curves.

Task	**GCBC**	**HGCBC** (+ repr.)	**IQL**	**POR** (+ repr.)	**HIQL (ours)**
gc-procgen-500-train	16.8 ±2.8	14.3 ±4.1	72.5 ±10.0	75.8 ±12.1	**82.5** ±6.0
gc-procgen-500-test	14.5 ±5.0	11.2 ±3.7	49.5 ±9.8	53.8 ±14.5	**64.5** ±13.2
gc-procgen-1000-train	27.2 ±8.9	15.0 ±5.7	78.2 ±7.2	82.0 ±6.5	**87.0** ±13.9
gc-procgen-1000-test	12.0 ±5.9	14.5 ±5.0	60.0 ±10.6	69.8 ±7.4	**78.2** ±17.9

6.2. Results on state-based environments

We first evaluate HIQL in the five state-based environments (AntMaze-{Medium, Large, Ultra}, Kitchen, and CALVIN) using nine offline datasets. We periodically evaluate the performance of the learned policies by commanding them with the evaluation goal state g (*i.e.*, the benchmark task target position in AntMaze, or the state that corresponds to completing all four subtasks in Kitchen and CALVIN), and measuring the average return with respect to the original benchmark task reward function. We test two versions of HIQL (without and with representations) in state-based environments. Table 1 shows the results on the nine offline datasets, indicating that HIQL mostly achieves the best performance in our experiments. Notably, HIQL attains an 88% success rate on gc-antmaze-large-diverse and 53% on gc-antmaze-ultra-diverse, which is, to the best of our knowledge, better than any previously reported result on these datasets.[1] In manipulation domains, we find that having latent waypoint representations in HIQL is important for enabling good performance. In CALVIN, while other methods often fail to achieve any of the subtasks due to the high diversity in the data, HIQL completes approximately

[1] We note that we use goal-conditioned variants of AntMaze, which differ from the original tasks. These variants could potentially be more difficult as the policy needs to learn to reach *any* goal from any state, but they might also be potentially easier given that the states contain goal information. We note that the prior work (Jiang et al., 2023), from which we take the results of TT and TAP, also provides the goal information in the state. As the performance of IQL in Table 1 is similar to that of the original paper (Kostrikov et al., 2022), we believe these goal-conditioned variants have similar level of difficulties to the original ones.

Table 3. **HIQL can leverage passive, action-free data.** Since our method requires action information only for the low-level policy, which is relatively easier to learn, HIQL mostly achieves comparable performance with just 25% of action-labeled data, outperforming even baselines trained on full datasets.

Task	IQL (full)	POR (full)	HIQL (full)	HIQL (action-limited)
gc-antmaze-large-diverse	50.7 $\pm$18.8	49.0 $\pm$17.2	88.2 $\pm$5.3	88.9 $\pm$6.4
gc-antmaze-ultra-diverse	21.6 $\pm$15.2	29.8 $\pm$13.6	52.9 $\pm$17.4	38.2 $\pm$15.4
gc-kitchen-mixed	51.3 $\pm$12.8	27.9 $\pm$17.9	67.7 $\pm$6.8	59.1 $\pm$9.6
gc-calvin	7.8 $\pm$17.6	12.4 $\pm$18.6	43.8 $\pm$39.5	35.8 $\pm$30.7
gc-procgen-500-train	72.5 $\pm$10.0	75.8 $\pm$12.1	82.5 $\pm$6.0	77.0 $\pm$12.5
gc-procgen-500-test	49.5 $\pm$9.8	53.8 $\pm$14.5	64.5 $\pm$13.2	65.5 $\pm$16.4

two subtasks on average.

6.3. Results on pixel-based environments

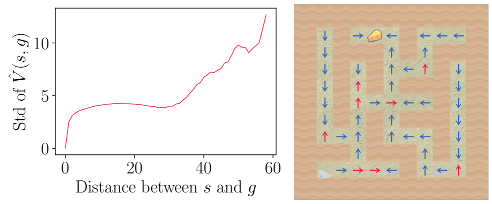
Next, to verify whether HIQL can scale to high-dimensional environments using goal representations, we evaluate our method on the Procgen Maze environment with $64 \times 64 \times 3$ image observations. We train HIQL and previous approaches using an offline dataset collected from either 500 or 1000 maze levels with varying difficulties, and assess them on both the training and test sets consisting of challenging levels (see Appendix for full details). We use a sparse goal-conditioned reward function as in previous experiments. For the prior hierarchical approaches that generate raw states (HGCBC and POR), we apply HIQL's representation learning scheme to enable them to handle the high-dimensional observation space. Table 2 presents the results, showing that our hierarchical policy extraction scheme, combined with representation learning, improves performance in these image-based environments as well. Notably, HIQL has larger gaps compared to the previous methods on the test sets. This is likely because the high-level policy can generalize better than the flat policy, as it can focus on the long-term direction toward the goal rather than the maze's detailed layout.

6.4. Results with action-free data

As mentioned in Section 5.1, one of the advantages of HIQL is its ability to leverage a potentially large amount of passive (action-free) data. To empirically verify this capability, we train HIQL on action-limited datasets, where we provide action labels for just 25% of the trajectories and use state-only trajectories for the remaining 75%. Table 3 shows the results from six different tasks, demonstrating that HIQL, even with a limited amount of action information, can mostly maintain its original performance. Notably, action-limited HIQL still outperforms previous offline RL methods (IQL and POR) trained with the full action-labeled data. We believe this is because HIQL learns a majority of the knowledge through hierarchical waypoint prediction from state-only trajectories.

6.5. Analysis

Does HIQL mitigate policy errors caused by noisy value functions in practice? To empirically verify whether our two-level policy architecture is more robust to errors in the learned value function (*i.e.*, the "signal-to-noise" ratio argument in Section 4), we compare the policy accuracies of IQL (flat policy), POR (hierarchy without temporal abstraction), and HIQL (ours) in Procgen Maze, by evaluating the ratio at which the ground-truth actions match the learned actions. We also measure the noisiness (*i.e.*, standard deviation) of the learned value function with respect to the ground-truth distance between the state and the goal. Figure 6 shows the results. We first observe that the noise in the value function generally becomes larger as the state-goal distance increases. Consequently, HIQL achieves the best policy accuracy, especially for distant goals ($\mathrm{dist}(s, g) \geq 50$), as its hierarchical policy extraction scheme provides the policies with clearer learning signals (Section 4.2). We refer to the supplementary materials for further analyses, including **waypoint visualizations** and an **ablation study** on waypoint steps and design choices for representations.



Policy accuracy metric	IQL	POR (+ repr.)	HIQL (ours)
All goals (train)	61.9 $\pm$1.9	64.5 $\pm$2.7	**66.6** $\pm$2.1 (+2.1)
All goals (test)	60.3 $\pm$2.8	63.6 $\pm$3.2	**68.0** $\pm$4.1 (+4.4)
Distant goals (train)	49.3 $\pm$3.3	48.1 $\pm$4.5	**56.8** $\pm$8.9 (+7.5)
Distant goals (test)	47.5 $\pm$8.6	47.2 $\pm$3.7	**59.9** $\pm$10.4 (+12.4)

Figure 6. **Value and policy errors in Procgen Maze**: *(top left)* As the distance between the state and the goal increases, the learned value function becomes noisier. *(top right)* We measure the accuracies of learned policies. *(bottom)* Thanks to our hierarchical policy extraction scheme (Section 4.2), HIQL exhibits the best policy accuracy, especially when the goal is far away from the state. The blue numbers denote the accuracy differences between HIQL and the second-best methods.

7. Conclusion

We proposed HIQL as a simple yet effective hierarchical algorithm for offline goal-conditioned RL. While hierarchical RL methods tend to be complex, involving many different components and objectives, HIQL shows that it is possible to build a method where a single value function simultaneously drives the learning of the low-level policy, the high-level policy, and the representations in a relatively simple and easy-to-train framework. We showed that HIQL not only exhibits strong performance in various challenging goal-conditioned tasks, but also can leverage action-free data and enjoy the benefits of built-in representation learning for image-based tasks. Due to space constraints, we further discuss **limitations** and future work in Appendix.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Ba, J., Kiros, J. R., and Hinton, G. E. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

Brandfonbrener, D., Whitney, W. F., Ranganath, R., and Bruna, J. Offline rl without off-policy evaluation. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *ArXiv*, abs/1606.01540, 2016.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Chane-Sane, E., Schmid, C., and Laptev, I. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning (ICML)*, 2021.

Chebotar, Y., Hausman, K., Lu, Y., Xiao, T., Kalashnikov, D., Varley, J., Irpan, A., Eysenbach, B., Julian, R. C., Finn, C., and Levine, S. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In *International Conference on Machine Learning (ICML)*, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Dayan, P. and Hinton, G. E. Feudal reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 1992.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

Ding, Y., Florensa, C., Phielipp, M., and Abbeel, P. Goal-conditioned imitation learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Durugkar, I., Tec, M., Niekum, S., and Stone, P. Adversarial intrinsic motivation for reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018.

Eysenbach, B., Salakhutdinov, R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Eysenbach, B., Salakhutdinov, R., and Levine, S. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations (ICLR)*, 2021.

Eysenbach, B., Zhang, T., Salakhutdinov, R., and Levine, S. Contrastive learning as goal-conditioned reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Fang, K., Yin, P., Nair, A., and Levine, S. Planning to practice: Efficient online fine-tuning by composing goals in latent space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022a.

Fang, K., Yin, P., Nair, A., Walke, H., Yan, G., and Levine, S. Generalization with lossy affordances: Leveraging broad offline data for learning visuomotor tasks. In *Conference on Robot Learning (CoRL)*, 2022b.

Fang, M., Zhou, C., Shi, B., Gong, B., Xu, J., and Zhang, T. Dher: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations (ICLR)*, 2019.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

Garg, D., Hejna, J., Geist, M., and Ermon, S. Extreme q-learning: Maxent rl without entropy. In *International Conference on Learning Representations (ICLR)*, 2023.

Ghasemipour, S. K. S., Schuurmans, D., and Gu, S. S. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning (ICML)*, 2021.

Ghosh, D. dibyaghosh/jaxrl_m, 2023. URL https://github.com/dibyaghosh/jaxrl_m.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Ghosh, D., Bhateja, C., and Levine, S. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning (ICML)*, 2023.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

He, K., Chen, X., Xie, S., Li, Y., Doll'ar, P., and Girshick, R. B. Masked autoencoders are scalable vision learners. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2022.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *ArXiv*, abs/1606.08415, 2016.

Hoang, C., Sohn, S., Choi, J., Carvalho, W., and Lee, H. Successor feature landmarks for long-horizon goal-conditioned reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Hong, Z.-W., Yang, G., and Agrawal, P. Bilinear value networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Huang, Z., Liu, F., and Su, H. Mapping state space using landmarks for universal goal reaching. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Janner, M., Li, Q., and Levine, S. Reinforcement learning as one big sequence modeling problem. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, 2022.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktaschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. In *International Conference on Learning Representations (ICLR)*, 2023.

Kaelbling, L. P. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

Kim, J., Seo, Y., and Shin, J. Landmark-guided subgoal generation in hierarchical reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Kim, J., Seo, Y., Ahn, S., Son, K., and Shin, J. Imitating graph-based planning with goal-conditioned policies. In *International Conference on Learning Representations (ICLR)*, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on Robot Learning (CoRL)*, 2017.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. B. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Neural Information Processing Systems (NeurIPS)*, 2016.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Kumar, A., Agarwal, R., Ma, T., Courville, A. C., Tucker, G., and Levine, S. Dr3: Value-based deep reinforcement learning requires explicit regularization. In *International Conference on Learning Representations (ICLR)*, 2022.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pp. 45–73. Springer, 2012.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Levy, A., Konidaris, G. D., Platt, R. W., and Saenko, K. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations (ICLR)*, 2019.

Li, A. C., Pinto, L., and Abbeel, P. Generalized hindsight for reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Li, J., Tang, C., Tomizuka, M., and Zhan, W. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):10216–10223, 2022.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *Conference on Robot Learning (CoRL)*, 2019.

Ma, Y. J., Yan, J., Jayaraman, D., and Bastani, O. How far i'll go: Offline goal-conditioned reinforcement learning via f-advantage regression. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Ma, Y. J., Sodhani, S., Jayaraman, D., Bastani, O., Kumar, V., and Zhang, A. Vip: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations (ICLR)*, 2023.

Machado, M. C., Bellemare, M. G., and Bowling, M. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Nair, A., Dalal, M., Gupta, A., and Levine, S. Accelerating online reinforcement learning with offline datasets. *ArXiv*, abs/2006.09359, 2020.

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

Nasiriany, S., Pong, V. H., Lin, S., and Levine, S. Planning with goal-conditioned policies. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Neumann, G. and Peters, J. Fitted q-iteration by advantage weighted regression. In *Neural Information Processing Systems (NeurIPS)*, 2008.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *ArXiv*, abs/1910.00177, 2019.

Pertsch, K., Lee, Y., and Lim, J. J. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning (ICML)*, 2007.

Peters, J., Muelling, K., and Altun, Y. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

Pong, V. H., Gu, S. S., Dalal, M., and Levine, S. Temporal difference models: Model-free deep rl for model-based control. In *International Conference on Learning Representations (ICLR)*, 2018.

Rosete-Beas, E., Mees, O., Kalweit, G., Boedecker, J., and Burgard, W. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2022.

Salter, S., Wulfmeier, M., Tirumala, D., Heess, N. M. O., Riedmiller, M. A., Hadsell, R., and Rao, D. Mo2: Model-based offline options. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

Savinov, N., Dosovitskiy, A., and Koltun, V. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations (ICLR)*, 2018.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning (ICML)*, 2015.

Schmidhuber, J. Learning to generate sub-goals for action sequences. In *Artificial neural networks*, 1991.

Seo, Y., Lee, K., James, S., and Abbeel, P. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning (ICML)*, 2022.

Shah, D., Eysenbach, B., Kahn, G., Rhinehart, N., and Levine, S. Recon: Rapid exploration for open-world navigation with latent goal models. In *Conference on Robot Learning (CoRL)*, 2021.

Shi, L., Lim, J. J., and Lee, Y. Skill-based model-based reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2022.

Stolle, M. and Precup, D. Learning options in reinforcement learning. In *Symposium on Abstraction, Reformulation and Approximation*, 2002.

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

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N. M. O., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Villaflor, A. R., Huang, Z., Pande, S., Dolan, J. M., and Schneider, J. G. Addressing optimism bias in sequence modeling for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2022.

Wang, T., Torralba, A., Isola, P., and Zhang, A. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning (ICML)*, 2023.

Wang, Z., Novikov, A., Zolna, K., Springenberg, J. T., Reed, S. E., Shahriari, B., Siegel, N., Merel, J., Gulcehre, C., Heess, N. M. O., and de Freitas, N. Critic regularized regression. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *ArXiv*, abs/1911.11361, 2019.

Wulfmeier, M., Rao, D., Hafner, R., Lampe, T., Abdolmaleki, A., Hertweck, T., Neunert, M., Tirumala, D., Siegel, N., Heess, N. M. O., and Riedmiller, M. A. Data-efficient hindsight off-policy option learning. In *International Conference on Machine Learning (ICML)*, 2021.

Xu, H., Jiang, L., Li, J., and Zhan, X. A policy-guided imitation approach for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Xu, H., Jiang, L., Li, J., Yang, Z., Wang, Z., Chan, V., and Zhan, X. Offline rl with no ood actions: In-sample learning via implicit value regularization. In *International Conference on Learning Representations (ICLR)*, 2023.

Yang, M., Schuurmans, D., Abbeel, P., and Nachum, O. Dichotomy of control: Separating what you can control from what you cannot. In *International Conference on Learning Representations (ICLR)*, 2023.

Yang, R., Lu, Y., Li, W., Sun, H., Fang, M., Du, Y., Li, X., Han, L., and Zhang, C. Rethinking goal-conditioned supervised learning and its connection to offline rl. In *International Conference on Learning Representations (ICLR)*, 2022.

Zhang, L., Yang, G., and Stadie, B. C. World model as a graph: Learning latent landmarks for planning. In *International Conference on Machine Learning (ICML)*, 2021.

Zhang, T., Guo, S., Tan, T., Hu, X., and Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Zhang, T., Eysenbach, B., Salakhutdinov, R., Levine, S., and Gonzalez, J. C-planning: An automatic curriculum for learning goal-reaching tasks. In *International Conference on Learning Representations (ICLR)*, 2022.

Algorithm 1 Hierarchical Implicit Q-Learning (HIQL)

1: **Input**: offline dataset $\mathcal{D}$, action-free dataset $\mathcal{D}_\mathcal{S}$ (optional, $\mathcal{D}_\mathcal{S} = \mathcal{D}$ otherwise)
2: Initialize value function $V_{\theta_V}(s, \phi(g))$ with built-in representation $\phi(g)$, high-level policy $\pi^h_{\theta_h}(z_{t+k}|s_{t+k}, g)$, low-level policy $\pi^\ell_{\theta_\ell}(a|s_t, z_{t+k})$, learning rates $\lambda_V, \lambda_h, \lambda_\ell$
3: **while** not converged **do**
4: $\quad\theta_V \leftarrow \theta_V - \lambda_V \nabla_{\theta_V} \mathcal{L}_V(\theta_V)$ with $(s_t, s_{t+1}, g) \sim \mathcal{D}_\mathcal{S}$ # Train value function, Equation (4)
5: **end while**
6: **while** not converged **do**
7: $\quad\theta_h \leftarrow \theta_h + \lambda_h \nabla_{\theta_h} J_{\pi^h}(\theta_h)$ with $(s_t, s_{t+\tilde{k}}, g) \sim \mathcal{D}_\mathcal{S}$ # Extract high-level policy, Equation (6)
8: **end while**
9: **while** not converged **do**
10: $\quad\theta_\ell \leftarrow \theta_\ell + \lambda_\ell \nabla_{\theta_\ell} J_{\pi^\ell}(\theta_\ell)$ with $(s_t, a_t, s_{t+1}, \tilde{s}_{t+k}) \sim \mathcal{D}$ # Extract low-level policy, Equation (7)
11: **end while**

A. Limitations

One limitation of HIQL is that the objective for its action-free value function (Equation (4)) is unbiased only when the environment dynamics are deterministic. As discussed in Section 3, HIQL (and other prior methods that use action-free videos) may overestimate the value function in partially observed or stochastic settings. To mitigate the optimism bias of HIQL in stochastic environments, we believe disentangling controllable parts from uncontrollable parts of the environment can be one possible solution (Villaflor et al., 2022; Yang et al., 2023), which we leave for future work.

B. Training details

Goal distributions. We train our goal-conditioned value function, high-level policy, and low-level policy respectively with Equations (4), (6) and (7), using different goal-sampling distributions. For the value function (Equation (4)), we sample the goals from either random states, futures states, or the current state with probabilities of 0.3, 0.5, and 0.2, respectively, following Ghosh et al. (2023). We use $\mathrm{Geom}(1-\gamma)$ for the future state distribution and the uniform distribution over the offline dataset for sampling random states. For the hierarchical policies, we mostly follow the sampling strategy of Gupta et al. (2019). We first sample a trajectory $(s_0, s_1, \dots, s_t, \dots, s_T)$ from the dataset $\mathcal{D}_\mathcal{S}$ and a state s_t from the trajectory. For the high-level policy (Equation (6)), we either (*i*) sample g uniformly from the future states s_{t_g} $(t_g > t)$ in the trajectory and set the target waypoint to $s_{\min(t+k, t_g)}$ or (*ii*) sample g uniformly from the dataset and set the target waypoint to $s_{\min(t+k,T)}$. For the low-level policy (Equation (7)), we first sample a state s_t from $\mathcal{D}$, and set the input waypoint to $s_{\min(t+k,T)}$ in the same trajectory.

Advantage estimates. In principle, the advantage estimates for Equations (6) and (7) are respectively given as

$$A^h(s_t, s_{t+\tilde{k}}, g) = \gamma^{\tilde{k}} V_{\theta_V}(s_{t+\tilde{k}}, g) + \sum_{t'=t}^{\tilde{k}-1} r(s_{t'}, g) - V_{\theta_V}(s_t, g), \tag{8}$$

$$A^\ell(s_t, a_t, \tilde{s}_{t+k}) = \gamma V_{\theta_V}(s_{t+1}, \tilde{s}_{t+k}) + r(s_t, \tilde{s}_{t+k}) - V_{\theta_V}(s_t, \tilde{s}_{t+k}), \tag{9}$$

where we use the notations $\tilde{k}$ and $\tilde{s}_{t+k}$ to incorporate the edge cases discussed in the previous paragraph (*i.e.*, $\tilde{k} = \min(k, t_g - t)$ when we sample g from future states, $\tilde{k} = \min(k, T - t)$ when we sample g from random states, and $\tilde{s}_{t+k} = s_{\min(t+k,T)}$). Here, we note that $s_{t'} \neq g$ and $s_t \neq \tilde{s}_{t+k}$ always hold except for those edge cases. Thus, the reward terms in Equations (8) and (9) are mostly constants, as are the third terms (with respect to the policy inputs). As such, we practically ignore these terms for simplicity, and this simplification further enables us to subsume the discount factors in the first terms into the temperature hyperparameter β. We hence use the following simplified advantage estimates, which we empirically found to lead to almost identical performances in our experiments:

$$\tilde{A}^h(s_t, s_{t+\tilde{k}}, g) = V_{\theta_V}(s_{t+\tilde{k}}, g) - V_{\theta_V}(s_t, g), \tag{10}$$

$$\tilde{A}^\ell(s_t, a_t, \tilde{s}_{t+k}) = V_{\theta_V}(s_{t+1}, \tilde{s}_{t+k}) - V_{\theta_V}(s_t, \tilde{s}_{t+k}). \tag{11}$$

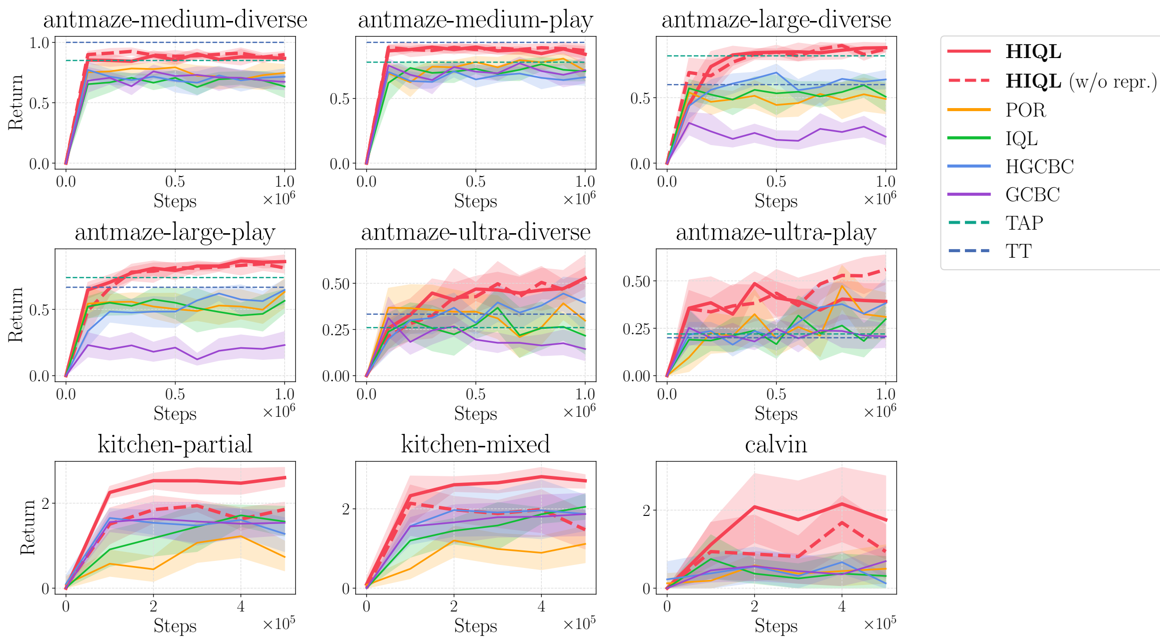


Figure 7. Training curves for the results with state-based environments (Table 1). Shaded regions denote the 95% confidence intervals across 8 random seeds.

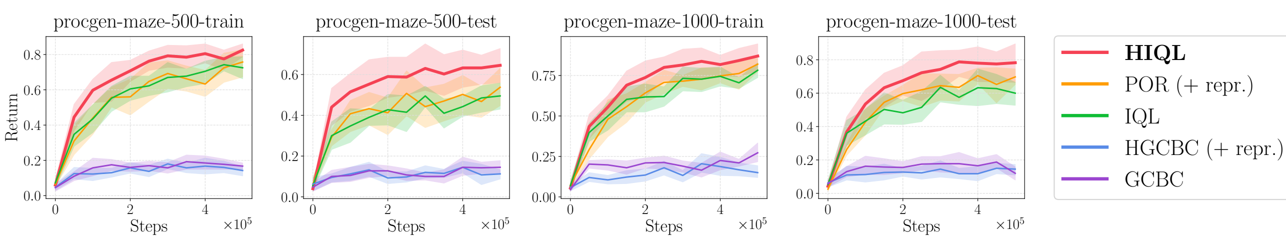


Figure 8. Training curves for the results with pixel-based Procgen Maze environments (Table 2). Shaded regions denote the 95% confidence intervals across 8 random seeds.

State representations. We model the output of the representation function $\phi(g)$ in $V(s, \phi(g))$ with a 10-dimensional latent vector and normalize the outputs of $\phi(g)$ (Kumar et al., 2022). Empirically, we found that concatenating s to the input (*i.e.*, using $\phi([g, s])$ instead of $\phi(g)$), similarly to Hong et al. (2022), improves performance in our experiments. While this might lose the sufficiency property of the representations (*i.e.*, Proposition 5.1), we found that the representations obtained in this way generally lead to better performance in practice, indicating that they still mostly preserve the goal information for control. We believe this is due to the imposed bottleneck on ϕ by constraining its effective dimensionality to 9 (by using normalized 10-dimensional vectors), which enforces ϕ to retain bits regarding g and to reference s only when necessary. Additionally, in pixel-based environments, we found that allowing gradient flows from the low-level policy loss (Equation (7)) to ϕ further improves performance. We ablate these choices and report the results in Appendix D.

We provide a pseudocode for HIQL in Algorithm 1. We note that the high- and low-level policies can be jointly trained with the value function as well.

C. Additional Plots

We include the training curves for Tables 1 to 3 in Figures 7 to 9, respectively. We include the full Rliable (Agarwal et al., 2021) plots in Figures 10 and 11.

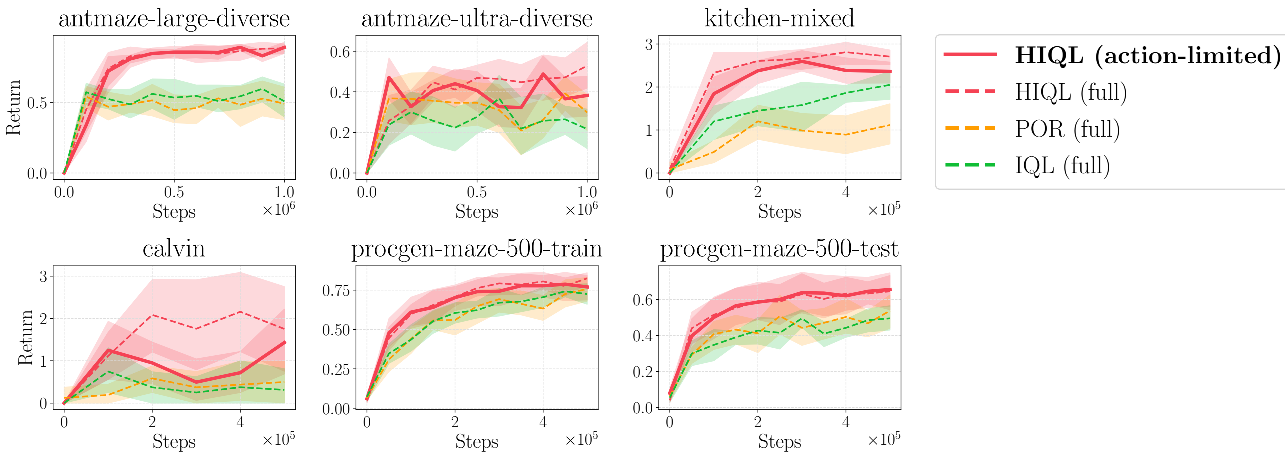


Figure 9. Training curves for the results with action-free data (Table 3). Shaded regions denote the 95% confidence intervals across 8 random seeds.

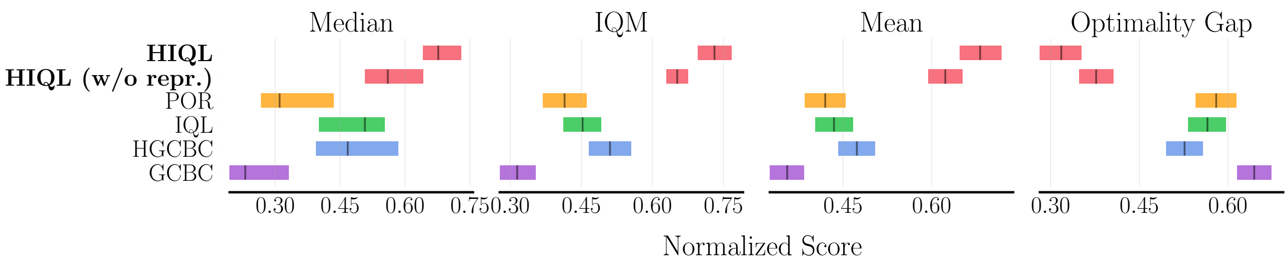


Figure 10. Rliable plots for state-based environments.

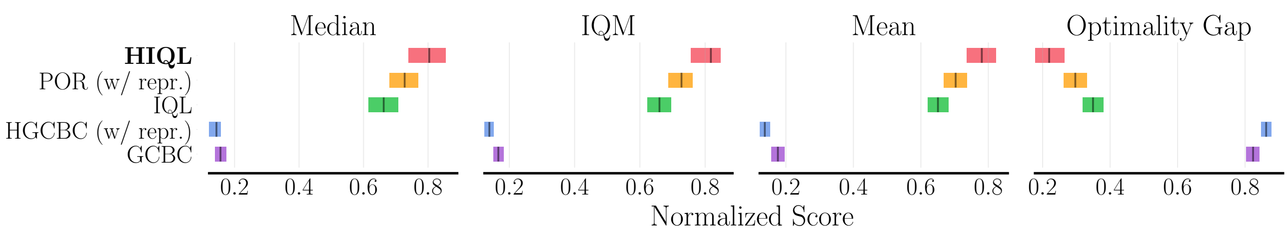


Figure 11. Rliable plots for pixel-based environments.

D. Ablation Study

Waypoint steps. To understand how the waypoint steps k affect performance, we evaluate HIQL with six different $k \in \{1, 5, 15, 25, 50, 100\}$ on AntMaze, Kitchen, and CALVIN. On AntMaze, we test both HIQL with and without representations (Section 5.2). Figure 12 shows the results, suggesting that HIQL generally achieves the best performance with k between 25 and 50. Also, HIQL still maintains reasonable performance even when k is not within this optimal range, unless k is too small.

Representation parameterizations. We evaluate four different choices of the representation function ϕ in HIQL: $\phi([g, s])$, $\phi(g - s)$, $\phi(g)$, and without ϕ. Figure 13 shows the results, indicating that passing g and s together to ϕ generally improves performance. We hypothesize that this is because ϕ, when given both g and s, can capture contextualized information about the goals (or waypoints) with respect to the current state, which is often easier to deal with for the low-level policy. For example, in AntMaze, the agent only needs to know the relative position of the waypoint with respect to the current position.

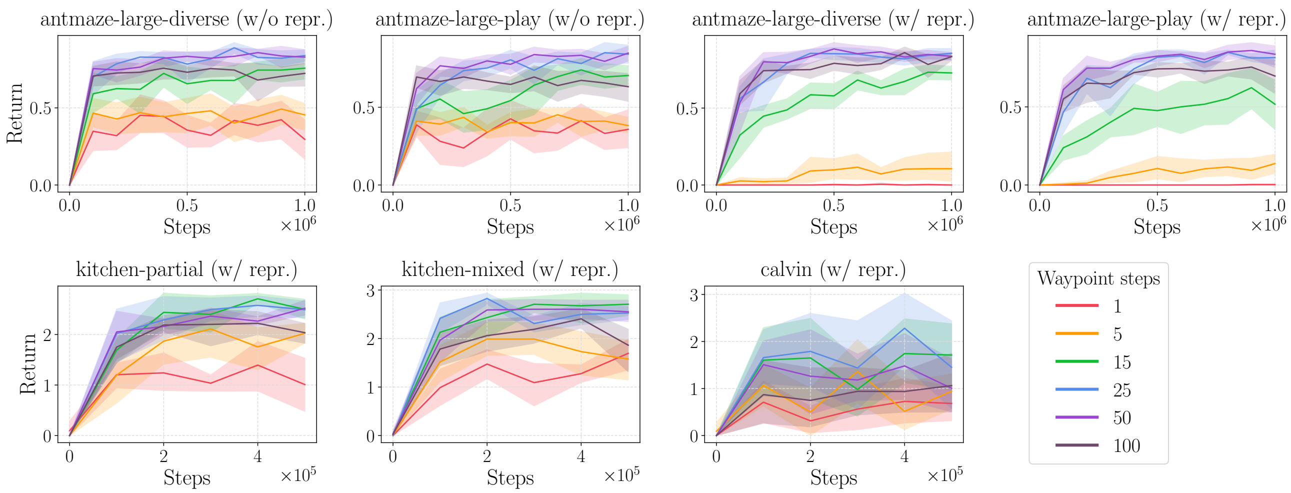


Figure 12. Ablation study of the waypoint steps k. HIQL generally achieves the best performances when k is between 25 and 50. Even when k is not within this range, HIQL mostly maintains reasonably good performance unless k is too small (*i.e.*, ≤ 5). Shaded regions denote the 95% confidence intervals across 8 random seeds.

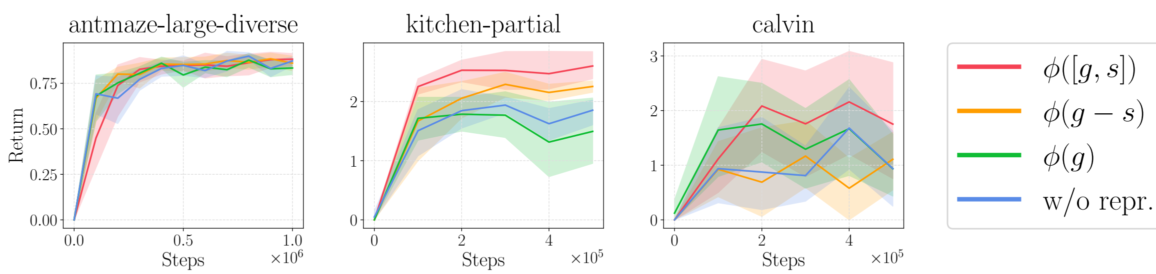


Figure 13. Ablation study of different parameterizations of the representation function. Passing s and g together to ϕ improves performance in general. Shaded regions denote the 95% confidence intervals across 8 random seeds.

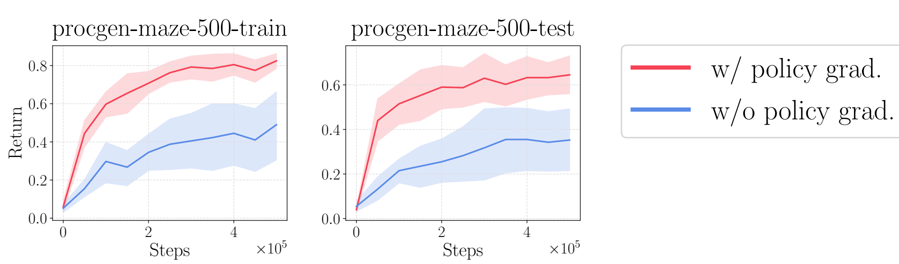


Figure 14. Ablation study of the auxiliary gradient flow from the low-level policy loss to ϕ on pixel-based ProcGen Maze. This auxiliary gradient flow helps maintain goal information in the representations. Shaded regions denote the 95% confidence intervals across 8 random seeds.

Auxiliary gradient flows for representations. We found that in Procgen Maze, allowing gradient flows from the low-level policy loss to the representation function enhances performance (Figure 14). We believe this is because the additional gradients from the policy loss further help maintain the information necessary for control. We also (informally) found that this additional gradient flow occasionally slightly improves performances in the other environments as well, but we do not enable this feature in the other domains to keep our method as simple as possible.

E. Implementation details

We implement HIQL based on JaxRL Minimal (Ghosh, 2023). Our implementation is available at the following anonymized repository: `https://github.com/hiqlauthors/hiql`. We run our experiments on an internal GPU cluster composed of TITAN RTX and A5000 GPUs. Each experiment on state-based environments takes no more than 8 hours and each experiment on pixel-based environments takes no more than 16 hours.

E.1. Environments

AntMaze (Todorov et al., 2012; Brockman et al., 2016) We use the antmaze-medium-diverse-v2, antmaze-medium-play-v2, antmaze-large-diverse-v2, and antmaze-large-play-v2 datasets from the D4RL benchmark (Fu et al., 2020). For AntMaze-Ultra, we use the antmaze-ultra-diverse-v0 and antmaze-ultra-play-v0 datasets proposed by Jiang et al. (2023). The maze in the AntMaze-Ultra task is twice the size of the largest maze in the original D4RL dataset. Each dataset consists of 999 length-1000 trajectories, in which the Ant agent navigates from an arbitrary start location to another goal location, which does not necessarily correspond to the target evaluation goal. At test time, to specify a goal g for the policy, we set the first two state dimensions (which correspond to the x-y coordinates) to the target goal given by the environment and the remaining proprioceptive state dimensions to those of the first observation in the dataset. At evaluation, the agent gets a reward of 1 when it reaches the goal.

Kitchen (Gupta et al., 2019). We use the kitchen-partial-v0 and kitchen-mixed-v0 datasets from the D4RL benchmark (Fu et al., 2020). Each dataset consists of 136950 transitions with varying trajectory lengths (approximately 227 steps per trajectory on average). In the kitchen-partial-v0 task, the goal is to achieve the four subtasks of opening the microwave, moving the kettle, turning on the light switch, and sliding the cabinet door. The dataset contains a small number of successful trajectories that achieve the four subtasks. In the kitchen-mixed-v0 task, the goal is to achieve the four subtasks of opening the microwave, moving the kettle, turning on the light switch, and turning on the bottom left burner. The dataset does not contain any successful demonstrations, only providing trajectories that achieve some subset of the four subtasks. At test time, to specify a goal g for the policy, we set the proprioceptive state dimensions to those of the first observation in the dataset and the other dimensions to the target kitchen configuration given by the environment. At evaluation, the agent gets a reward of 1 whenever it achieves a subtask.

CALVIN (Mees et al., 2022). We use the offline dataset provided by Shi et al. (2022), which is based on the teleoperated demonstrations from Mees et al. (2022). The task is to achieve the four subtasks of opening the drawer, turning on the lightbulb, sliding the door to the left, and turning on the LED. The dataset consists of 1204 length-499 trajectories. In each trajectory, the agent achieves some of the 34 subtasks in an arbitrary order, which makes the dataset highly task-agnostic (Shi et al., 2022). At test time, to specify a goal g for the policy, we set the proprioceptive state dimensions to those of the first observation in the dataset and the other dimensions to the target configuration. At evaluation, the agent gets a reward of 1 whenever it achieves a subtask.

Procgen Maze (Cobbe et al., 2020). We collect an offline dataset of goal-reaching behavior on the Procgen Maze suite. For each maze level, we pre-compute the optimal goal-reaching policy using an oracle, and collect a trajectory of 1000 transitions by commanding a goal, using the goal-reaching policy to reach this goal, then commanding a new goal and repeating henceforth. The procgen-maze-500 dataset consists of 500000 transitions collected over the first 500 levels and procgen-maze-1000 consists of 1000000 transitions over the first 1000 levels. At test time, we evaluate the agent on "challenging" levels that contain at least 20 leaf goal states (*i.e.*, states that have only one adjacent state in the maze). We use 50 such levels and goals for each evaluation, where they are randomly sampled either between Level 0 and Level 499 for the "-train" tasks or between Level 5000 and Level 5499 for the "-test" tasks. The agent gets a reward of 1 when it reaches the goal.

In Tables 1 to 3, we report the normalized scores with a multiplier of 100 (AntMaze and Procgen Maze) or 25 (Kitchen and CALVIN).

E.2. Hyperparameters

We present the hyperparameters used in our experiments in Table 4, where we mostly follow the network architectures and hyperparameters used by Ghosh et al. (2023). We use layer normalization (Ba et al., 2016) for all MLP layers. For

Table 4. Hyperparameters.

Hyperparameter	Value
# gradient steps	1000000 (AntMaze), 500000 (others)
Batch size	1024 (state-based), 256 (pixel-based)
Policy MLP dimensions	$(256, 256)$
Value MLP dimensions	$(512, 512, 512)$
Representation MLP dimensions (state-based)	$(512, 512, 512)$
Representation architecture (pixel-based)	Impala CNN (Espeholt et al., 2018)
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	0.0003
Target network smoothing coefficient	0.005

pixel-based environments, we use the Impala CNN architecture (Espeholt et al., 2018) to handle image inputs, mostly with 512-dimensional output features, but we use normalized 10-dimensional output features for the goal encoder of HIQL's value function to make them easily predictable by the high-level policy, as discussed in Appendix B. During training, we periodically evaluate the performance of the learned policy at every 100K (state-based) or 50K (pixel-based) steps, using 52[2] (state-based) or 50 (pixel-based) rollouts. At evaluation, we use the $\arg\max$ actions for environments with continuous action spaces and ϵ-greedy actions with $\epsilon = 0.05$ for environments with discrete action spaces (*i.e.*, Procgen Maze).

To ensure fair comparisons, we use the same architecture for HIQL and all the baselines implemented in this work (*i.e.*, GCBC, HGCBC, IQL, and POR). The discount factor γ is chosen from $\{0.99, 0.995\}$, the AWR temperature β from $\{1, 3, 10\}$, the IQL expectile τ from $\{0.7, 0.9\}$ for each method.

For HIQL, we set $(\gamma, \beta, \tau) = (0.99, 1, 0.7)$ across all environments. For IQL and POR, we use $(\gamma, \beta, \tau) = (0.99, 3, 0.9)$ (AntMaze-Medium and AntMaze-Large), $(\gamma, \beta, \tau) = (0.995, 1, 0.7)$ (AntMaze-Ultra), or $(\gamma, \beta, \tau) = (0.99, 1, 0.7)$ (others). For the waypoint steps k in HIQL, we use $k = 50$ (AntMaze-Ultra), $k = 3$ (Procgen Maze), or $k = 25$ (others). HGCBC uses the same waypoint steps as HIQL for each environment, with the exception of AntMaze-Ultra, where we find it performs slightly better with $k = 25$. In state-based environments, we sample goals for high-level or flat policies from either the future states in the same trajectory (with probability 0.7) or the random states in the dataset (with probability 0.3). We sample high-level goals only from the future states when training HGCBC or GCBC or when using Procgen Maze.

F. Proofs

F.1. Proof of Proposition 4.1

For simplicity, we assume that T/k is an integer and $k \leq T$.

Proof. Defining $z_1 := z_{1,T}$ and $z_2 := z_{-1,T}$, the probability of the flat policy π selecting an incorrect action can be computed as follows:

$$\begin{aligned}
\mathcal{E}(\pi) &= \mathbb{P}[\hat{V}(s+1, g) \leq \hat{V}(s-1, g)] && (12)\\
&= \mathbb{P}[\hat{V}(1, T) \leq \hat{V}(-1, T)] && (13)\\
&= \mathbb{P}[-(T-1)(1+\sigma z_1) \leq -(T+1)(1+\sigma z_2)] && (14)\\
&= \mathbb{P}[z_1\sigma(T-1) - z_2\sigma(T+1) \leq -2] && (15)\\
&= \mathbb{P}[z\sigma\sqrt{T^2+1} \leq -\sqrt{2}] && (16)\\
&= \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{T^2+1}}\right), && (17)
\end{aligned}$$

where z is a standard Gaussian random variable, and we use the fact that the sum of two independent Gaussian random

[2] This includes two additional rollouts for video logging.

variables with standard deviations of σ_1 and σ_2 follows a normal distribution with a standard deviation of $\sqrt{\sigma_1^2+\sigma_2^2}$.

Similarly, the probability of the hierarchical policy $\pi^\ell \circ \pi^h$ selecting an incorrect action is bounded using a union bound as

$$\begin{aligned}
\mathcal{E}(\pi^\ell \circ \pi^h) &\leq \mathcal{E}(\pi^h) + \mathcal{E}(\pi^\ell) && (18)\\
&= \mathbb{P}[\hat{V}(s+k,g) \leq \hat{V}(s-k,g)] + \mathbb{P}[\hat{V}(s+1,s+k) \leq \hat{V}(s-1,s+k)] && (19)\\
&= \mathbb{P}[\hat{V}(k,T) \leq \hat{V}(-k,T)] + \mathbb{P}[\hat{V}(1,k) \leq \hat{V}(-1,k)] && (20)\\
&= \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{(T/k)^2+1}}\right) + \Phi\left(-\frac{\sqrt{2}}{\sigma\sqrt{k^2+1}}\right). && (21)
\end{aligned}$$

□

F.2. Proof of Proposition 5.1

We first formally define some notations. For $s \in \mathcal{S}, a \in \mathcal{A}, g \in \mathcal{S}$, and a representation function $\phi : \mathcal{S} \to \mathcal{Z}$, we denote the goal-conditioned state-value function as $V(s,g)$, the action-value function as $Q(s,a,g)$, the parameterized state-value function as $V_\phi(s,z)$ with $z = \phi(g)$, and the parameterized action-value function as $Q_\phi(s,a,z)$. We assume that the environment dynamics are deterministic, and denote the deterministic transition kernel as $p(s,a) = s'$. Accordingly, we have $Q(s,a,g) = V(p(s,a),g) = V(s',g)$ and $Q_\phi(s,a,z) = V_\phi(p(s,a),z) = V_\phi(s',z)$. We denote the optimal value functions with the superscript "$*$", *e.g.*, $V^*(s,g)$. We assume that there exists a parameterized value function, which we denote $V^*_\phi(s,\phi(g))$, that is the same as the true optimal value function, *i.e.*, $V^*(s,g) = V^*_\phi(s,\phi(g))$ for all $s \in \mathcal{S}$ and $g \in \mathcal{S}$.

Proof. For π^*, we have

$$\begin{aligned}
\pi^*(a \mid s,g) &= \arg\max_{a\in\mathcal{A}} Q^*(s,a,g) && (22)\\
&= \arg\max_{s'\in\mathcal{N}_s} V^*(s',g) && (23)\\
&= \arg\max_{s'\in\mathcal{N}_s} V^*_\phi(s',z), && (24)
\end{aligned}$$

where $\mathcal{N}_s$ denotes the neighborhood sets of s, *i.e.*, $\mathcal{N}_s = \{s' \mid \exists a, p(s,a) = s'\}$. For π^*_ϕ, we have

$$\begin{aligned}
\pi^*_\phi(a \mid s,z) &= \arg\max_{a\in\mathcal{A}} Q^*_\phi(s,a,z) && (25)\\
&= \arg\max_{s'\in\mathcal{N}_s} V^*_\phi(s',z). && (26)
\end{aligned}$$

By comparing Equation (24) and Equation (26), we can see that they have the same $\arg\max$ action sets for all s and g. □

G. Waypoint Visualizations

We visualize learned waypoints in Figures 15 and 16 (videos are available at `https://sites.google.com/view/hiql/`). For AntMaze-Large, we train HIQL without representations and plot the x-y coordinates of waypoints. For Procgen Maze, we train HIQL with 10-dimensional representations and find the maze positions that have the closest representations (with respect to the Euclidean distance) to the waypoints produced by the high-level policy. The results show that HIQL learns appropriate k-step waypoints that lead to the target goal.

Figure 15. Waypoint visualization in AntMaze-Large. The red circles denote the target goal and the blue circles denote the learned waypoints. Videos are available at https://sites.google.com/view/hiql/.

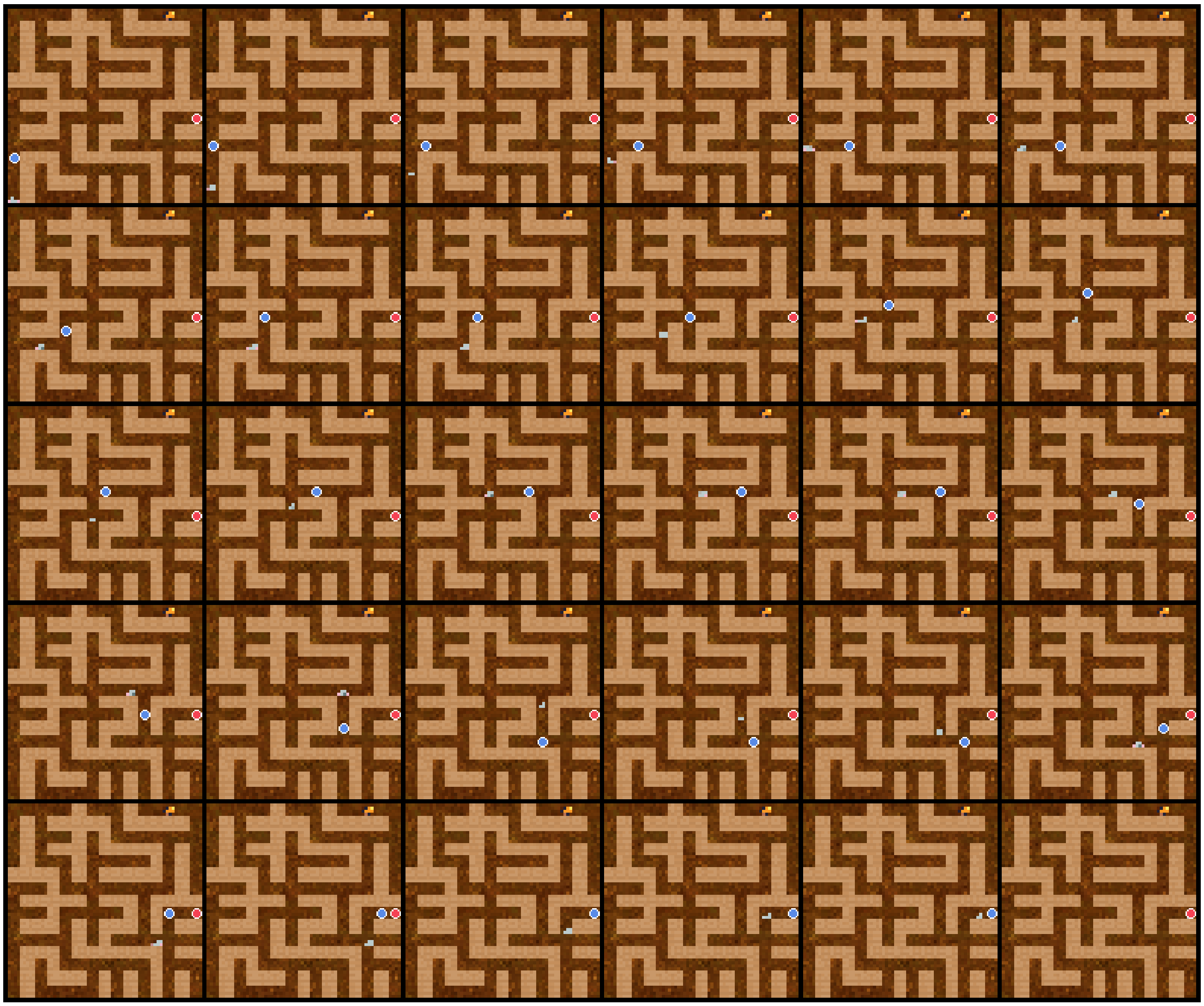


Figure 16. Waypoint visualization in Procgen Maze. The red circles denote the target goal, the blue circles denote the learned waypoints, and the white blobs denote the agent. Videos are available at https://sites.google.com/view/hiql/.