Abstraction for Offline Goal-Conditioned Reinforcement Learning

Clarisse Wibault
FLAIR, MLRG
University of Oxford
clarisse.wibault@magd.ox.ac.uk

Alexander D. Goldie
FLAIR
University of Oxford

Antonio Léon Villares
FLAIR
University of Oxford

Maike A. Osborne
MLRG
University of Oxford

Jakob N. Foerster
FLAIR
University of Oxford

Abstract

Markov Decision Processes (MDPs) often exhibit significant redundancy due to symmetries and shared structure across state-goal pairs in real-world Goal-Conditioned Reinforcement Learning (GCRL). While hierarchical policies have been motivated for horizon reduction via *temporal* abstraction in offline GCRL, we demonstrate that hierarchy also enables *absolute* abstraction. By introducing *relativised* options as well as *distinct representations* for different levels of the hierarchy, we demonstrate how an agent can reuse experience across similar contexts of the state-space. Based on this framework, we introduce two simple algorithms for learning relativised options and abstracting from the absolute frame of reference. Our experiments show that such inductive biases significantly improve performance in offline GCRL.

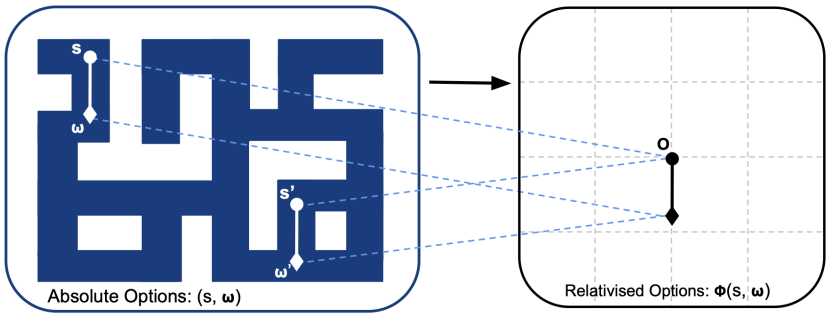

Figure 1: **Abstractive RL (ARL).** By learning relativised options, ARL enables the reuse of experience across similar contexts of the state-space.

1 Introduction

Offline Goal-Conditioned Reinforcement Learning (GCRL) [1–5] provides a principled framework for training a general-purpose agent to solve complex long-horizon tasks from static datasets. However, in practice, existing methods have struggled to learn effective policies from offline data, partly due to imperfect dataset coverage of the state-action space [3, 6]. Furthermore, since common offline

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning.

RL algorithms [7–9] regularise actions towards those close to the dataset distribution to mitigate issues such as distribution shift [3, 6], an agent may fail to recover an optimal policy if a dataset only contains low-quality actions in certain regions of the state space. This is a key challenge in offline RL [10] and leads to difficulties in value estimation, policy extraction, and policy generalisation [11].

Recent work suggests that horizon reduction is essential for scaling offline RL [12], motivating the use of hierarchical policies, or options [13], for temporal abstraction [14]. We extend this perspective by arguing that hierarchy offers an additional advantage: *absolute* abstraction. By using *absolute* abstraction, which we define as using *relativised* options and *distinct representations* at different levels of the hierarchy, an agent abstracts away from the absolute frame of reference and can reuse experience across similar contexts of the state-space.

To illustrate this point, consider an agent undertaking a locomotion task. While the dataset may only contain a limited number of suboptimal demonstrations of the full task, many of the constituent relativised options (subtasks such as simply moving forward or navigating a corner) might be well represented across demonstrations with different goals. By defining options relative to the agent's local context (e.g. *navigate to the corner directly ahead* rather than *navigate to corner A*) and decoupling low-level actions (i.e motor-actuation) from redundant high-level information, the agent can leverage many more subtask examples to learn a policy.

In principle, relativised options [15] exploit redundancy and symmetry in MDPs to allow behaviours to generalise across states. However, in practice, implementations remain limited to toy examples due to the inherent difficulty of identifying MDP homomorphisms or learning such relative representations. In this work, we introduce *Abstractive Reinforcement Learning* (ARL), a general framework that learns relativised options via action similarity. Based on this, it defines high-level similarity and low-level similarity respectively as state-goal pairs inducing similar options and state-option pairs inducing similar immediate actions.

We propose two simple algorithms that comply with the ARL framework: the first can be applied generally, simply using action similarity to implicitly learn relativised options; the second introduces a representational inductive bias for the low-level MDP by explicitly enforcing translational invariance, improving generalisation in certain high-dimensional manipulation tasks. Our experiments demonstrate that such relativised options and inductive biases result in better performance in high-dimensional offline GCRL.

Contributions. Concisely, this work addresses the following question:

Can hierarchy enable more robust policy extraction in regions where the dataset suffers from low-quality transitions?

Our contributions are two-fold. We first motivate hierarchy in offline RL through abstraction from the absolute frame of reference. We show how relativised options and distinct representations at different levels of the hierarchy can enable data reuse, bounding the maximum error in expected return compared to a flat policy. Secondly, we introduce two simple algorithms that learn relativised options and abstract from the absolute frame of reference. These algorithms outperform both flat policies and hierarchical ones that are anchored in the absolute state-space in high-dimensional tasks, without introducing additional hyperparameters.

Explicitly, for an RL practitioner, our work demonstrates that: (i) options should be learned via *action similarity* rather than value similarity i.e. jointly optimised with the low-level policy; (ii) we necessitate two value functions to decouple the high-level from the low-level decision process; and (iii) that imposing translation invariance on the low-level MDP can improve policy generalisation in high-dimensional manipulation tasks. For an RL researcher, our work opens up new avenues, such as methods to learn relativised options via action chunking [16, 17], or incorporating more flexible inductive biases using ideas from Geometric Deep Learning (GDL) [18, 19].

2 Preliminaries

In Offline [3, 20] Goal-Conditioned Reinforcement Learning (GCRL) [1] an agent seeks to learn a universal policy [2] from a fixed dataset, enabling it to reach arbitrary goal states in the smallest number of timesteps [5].

2.1 Problem Setting

We consider a standard Markov Decision Process (MDP) [21] defined by the tuple $\mathcal{M} := (p_{s_0}, \mathcal{S}, \mathcal{G}, \mathcal{A}, \mathcal{T}, \beta_g, \gamma)$, where p_{s_0} is the initial state distribution, $\mathcal{S}$, $\mathcal{G}$ and $\mathcal{A}$ respectively denote the state, goal and action space, $\mathcal{T}$ is the transition function, β_g is a goal-conditioned pseudo-termination function, and $\gamma < 1$ is the discount factor. At the beginning of the episode, a state s_0 is sampled from p_{s_0}. A goal state g is uniformly sampled from the goal space $\mathcal{G}$, and is fixed for the entire episode. The goal space may be defined over all or a subset of the state dimensions. At each timestep $t \geq 0$, an agent takes an action a_t conditioned on its current state s_t and goal state g, and transitions to a new state $s_{t+1} = \mathcal{T}(\cdot \mid s_t, a_t)$. Episodes terminate according to the goal-conditioned pseudo-termination function [22] $\beta_g : \mathcal{S} \to \{0, 1\}$, where $\beta_g(s) = 1$ if and only if the goal has been reached. Following Andrychowicz et al. [23], we focus on the problem of sparse and binary rewards, which is motivated in robotics, for example. The agent receives a reward of -1 on all steps, and a reward of 0 upon reaching the goal $r_t = -\mathbb{1}\{\beta_g(s_t) = 0\}$. The aim of the agent is to learn a universal policy [2] conditioned on its state and the goal $\pi : \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$ that maximises the sum of discounted returns $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ from a fixed dataset $\mathcal{D}$. The dataset contains N state-action trajectories of length H, collected using an arbitrary policy in a task-agnostic manner. Following Park et al. [24], we adopt the fully observed deterministic MDP framework for simplicity, though we additionally evaluate in a stochastic environment.

2.2 Offline Reinforcement Learning

Offline RL is commonly formulated as a two-step procedure: first, value learning, and subsequently policy extraction from this value function. A major challenge in offline RL is extracting an optimal policy from suboptimal data. This difficulty is exacerbated by distribution shift [3, 10], where the learned policy induces state-action pairs that are poorly supported by the dataset, leading to unreliable value estimates.

To address this issue, model-free algorithms typically incorporate some form of conservatism during value learning, regularise policy extraction towards the dataset distribution via behaviour-cloning, or combine both strategies. The value function might be learned with a distribution-constrained objective [25, 7], to minimise uncertainty [26] or encourage structured representations [27]. Similarly for policy extraction, the agent is regularised by using an additional behaviour-cloning loss term [28], weighted behaviour-cloning [8], or rejection-sampling [29] of a behaviour-cloned policy. In all cases:

Due to their underlying bias towards the dataset distribution, flat offline RL algorithms struggle in regions where the dataset suffers from low-quality transitions.

2.3 Options

Options [13] provide a framework for decision making at different levels of temporal abstraction. In the original options framework, each option $\omega \sim \Omega$ represents temporally extended behaviour, executed until termination according to a learned or predefined termination condition [30]. In contrast, hierarchical approaches in Offline GCRL typically resample options at every timestep [24, 12, 31], effectively removing temporal commitment and avoiding the need to specify a termination function:

$$\pi(a \mid s, g) = \pi_l(a \mid s, \omega) \quad \omega \sim \pi_h(\cdot \mid s, g),$$

where π_h is the high-level policy that samples an option conditioned on the current state and goal, and π_l is a low-level policy that samples an action conditioned on the same state and that sampled option. Such approaches still differ from a flat policy, since the low-level policy is conditioned on the option rather than the goal. Since local gradient information can be uninformative or misleading when optimising for distant goals, a key benefit of this hierarchical inductive bias is that it reduces the effective horizon for both the high-level and low-level MDP [24, 12].

3 Related Work

Extensive literature motivates hierarchy for temporal abstraction and policy horizon reduction [13, 32, 33], but apart from Nachum et al. [34], [35] and Levy et al. [36], very little work explicitly motivates hierarchy via representation learning and data reuse. While original work used a set of

hardcoded options [13], since then options have been learned by identifying bottleneck states (e.g. [37, 38]), or by jointly learning the options with the low-level policy [30]. Most work defines options in the original state-space, but, for example, Vezhnevets et al. [14] learn options in an embedding space. Ravindran and Barto [15] introduce the concept of relativised options, but non-hardcoded implementations of relativised options have remained scarce. We remark that options can be defined in a latent space, but still be anchored to the absolute frame of reference. All of these works study hierarchical RL in the online setting, while we focus on the offline setting.

Offline GCRL naturally lends itself to hierarchical formulations, where high-level policies specify intermediate goals, and low-level policies learn intermediate goal-reaching behaviours [5, 11, 12, 24, 31, 39]. In hierarchical offline GCRL, options have again generally taken the form of absolute options in the original state-space (e.g. [12, 31]). Our approach is most related to HIQL [24], which learns latent options, but differs in two fundamental ways. First, unlike HIQL, which uses one value function, we use two distinct value functions, enabling different representations at different levels of the hierarchy (Section 4.3). Second, rather than basing option embeddings on value similarity, we base them on action similarity (Section 4.2). This changes the organisation of the latent space, since two state-waypoints might induce similar values but differ entirely in their low-level actions.

The theory of state abstraction identifies conditions under which information can be compressed while maintaining policy optimality at different levels of abstraction [40]. Prior work in State Representation Learning (SRL) has explored homomorphisms, bisimulation metrics, and contrastive objectives to map together semantically similar states (e.g. [33, 41, 42]). However, these methods often require auxiliary loss terms that necessitate hyperparameter tuning and can lead to training instability. We refer to Echchahed and Castro [43] for an overview. Furthermore, none of these methods explicitly address hierarchical goal-conditioned representation learning.

4 Abstractive Reinforcement Learning (ARL)

In this section, we introduce Abstractive Reinforcement Learning (ARL), a framework for learning abstractions in offline hierarchical GCRL to improve robustness in regions where the dataset suffers from low-quality transitions. Based on this framework, we introduce two simple algorithms: the first learns relativised options via action similarity, while the second additionally imposes translational invariance on the low-level MDP. Together, these algorithms demonstrate how (i) relativised options and (ii) representational inductive biases can improve generalisation in offline GCRL.

4.1 Objective

In principle, our aim is to learn abstractions that enable reuse of experience across similar contexts. By learning options that group together state-waypoint pairs, which, under an optimal policy, induce similar action sequences, options $\omega \in \Omega$ are relative rather than anchored to an absolute frame of reference. This naturally induces two hierarchical notions of similarity: a high-level similarity, where similar state-goal pairs induce similar options, and a low-level similarity, where similar state-option pairs induce similar immediate actions. Consequently, we can define high-level embeddings $\phi_h(s, g)$ and low-level embeddings $\phi_l(s, \omega)$ that abstract away information irrelevant to their respective levels:

$$\pi(a \mid s, g) = \pi_l(a \mid \phi_l(s, \omega)) \quad \omega \sim \pi_h(\cdot \mid \phi_h(s, g)). \tag{1}$$

To allow the high-level and low-level decision processes to operate on different representations and at different temporal abstractions (i.e. different discount factors), such an approach necessitates two distinct value functions. In the following Motivation Box, we provide an intuition on how such abstractions can mitigate failures in regions where the dataset suffers from low-quality transitions by analysing the maximum error in offline RL for a finite-state, finite-action MDP.

Motivation Box: Bounding the Maximum Error

We consider learning an optimal policy in a finite-state, finite-action MDP from a fixed dataset $\mathcal{D}$ of size N. We refer to Appendix B for a complete derivation and full definitions. We assume at most two possible next states for each state-action pair, which is realistic given our deterministic transition assumption (Section 2). For a flat goal-conditioned policy, the Probably Approximately

Correct (PAC) Learning error ϵ in expected return is given by [44–46]:

$$\epsilon \propto \sqrt{\frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}| \cdot \kappa}{(1-\gamma)^3 N}}, \quad \kappa = \sup_{s \in \mathcal{S}, a \in \mathcal{A}, g \in \mathcal{S}} \frac{d^{\pi^\star}(s,a,g)}{d^{\pi^{BC}}(s,a,g)},$$

with constant probability of $1-\delta$. Here, the proportionality constant depends on δ. κ is the concentrability coefficient that accounts for the distribution shift in data collected by the offline behaviour cloning policy π^{BC}, and the data that would have been collected under the optimal policy $\pi^\star$. $d^\pi(s,a,g)$ is the discounted occupancy measure (stationary distribution) of the policy π. Intuitively, if the dataset does not include the state-action pairs required to learn the optimal policy, κ (and hence the error ϵ) will be large.

We build on the work of Robert et al. [47] and Li et al. [40] to show that, by using a hierarchical policy with absolute abstraction, the maximum error is bounded by:

$$\epsilon^{\text{hierarchy, rep}} \propto \sqrt{\frac{|\mathcal{C}_h||\Omega| \cdot \kappa_h^{\text{rep}}}{(1-\gamma^n)^3 N}} + \sqrt{\frac{|\mathcal{C}_l|\mathcal{A}|n^3 \cdot \kappa_l^{\text{rep}}}{N}}.$$

This reparameterisation reduces the error bound in four ways:

1. **Horizon reduction**: the high-level policy faces an effective discount factor of γ^n (with $n \geq 1$) rather than γ, reducing the denominator's sensitivity: $\frac{1}{(1-\gamma^n)^3} \leq \frac{1}{(1-\gamma)^3}$[a].
2. **Cardinality Reduction**: by mapping (s,g) and (s,ω) pairs into equivalence classes $c_h \in \mathcal{C}_h$ and $c_l \in \mathcal{C}_l$, the effective state-space is reduced: $|\mathcal{C}_h| \leq |\mathcal{S}||\mathcal{G}|$ and $|\mathcal{C}_l| \leq |\mathcal{S}||\Omega|$.
3. **Option Efficiency**: relativised options ensure that the option space is small and invariant to absolute position: $|\Omega^{\text{rel}}| \leq |\Omega^{\text{abs}}|$, where Ω^{abs} represents an option space anchored in an absolute frame of reference, and Ω^{rel} represents one in a relative frame of reference.
4. **Concentrability Improvement**: because the concentrability coefficients are now defined using reparameterised policies over the reparameterised latent spaces, the probability mass of the dataset is aggregated across similar contexts:

   $$\kappa_h^{\text{rep}} = \sup_{c_h \in \mathcal{C}_h, \omega \in \Omega} \frac{d^{\pi_h^\star}(c_h,\omega)}{d^{\pi_h^{BC}}(c_h,\omega)} \quad \text{and} \quad \kappa_l^{\text{rep}} = \sup_{c_l \in \mathcal{C}_l, a \in \mathcal{A}} \frac{d^{\pi_l^\star}(c_l,a)}{d^{\pi_l^{BC}}(c_l,a)},$$

   where

   $$d^{\pi_h}(c_h,\omega) = \sum_{(s,g) \in \phi_h^{-1}(c_h)} d_h^\pi(s,g,\omega) \quad \text{and} \quad d^{\pi_l}(c_l,a) = \sum_{(s,\omega) \in \phi_l^{-1}(c_l)} d_l^\pi(s,\omega,a).$$

   Even if the dataset has zero mass on a specific state-goal-option (s,g,ω) or state-option-action (s,ω,a), it likely has mass on the abstract-context-option (c_h,ω) or abstract-context-action (c_l,a). This reduces the likelihood of a support mismatch, where the optimal policy requires a state transition on which the dataset places zero mass. By aggregating similar contexts we now perform a ratio over sums such that

   $$\kappa_h^{\text{rep}} \leq \kappa_h \quad \text{and} \quad \kappa_l^{\text{rep}} \leq \kappa_l,$$

   where κ_h and κ_l are concentrability coefficients without using aggregating embeddings.

Consequently, for a fixed N and unlike a flat policy, such a reparameterisation could enable learning more optimal behaviour in regions that suffer from low-quality data.

[a] Note that while hierarchy introduces an additive error term for the low-level policy, this is typically dominated by the exponential reduction in the high-level error's horizon-dependent constant, especially in tasks where $\gamma \to 1$ [12].

4.2 Abstractive RL Implicitly Learning Relativised Options

Based on this objective, we propose a minimal amendment to HIQL [24] to encourage learning relativised options, which directly address the third point in the Motivation Box (Section 4.1). Rather

than learning option representations with the value function, we propose learning them via the low-level policy. Following HIQL, we also bound the option space to a hypersphere to introduce geometric regularisation. However, unlike HIQL, we learn representations via the low-level policy to push together state-waypoint pairs with similar low-level actions rather than state-waypoint pairs with similar values. Also unlike HIQL (and, as motivated in Section 4.1) we use two value functions rather than one.

Low-Level Value. Although ARL is agnostic to the choice of loss, we train the low-level value V_l and critic Q_l using Implicit Q Learning [7] to match our benchmark algorithms:

$$\mathcal{L}_{V_l} = \mathbb{E}_{(s,a)\sim\mathcal{D}, g_s\sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\ell^2_\tau\left(V_l(s,g_s) - \tilde{Q}_l(s,g_s,a)\right)\right] \tag{2}$$

$$\mathcal{L}_{Q_l} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}, g_s\sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\left(Q_l(s,g_s,a) - r(s,g_s) - \gamma_l V_l(s',g_s)\right)^2\right] \tag{3}$$

where, $\tilde{Q}_l$ denotes the target network, γ_l the low-level discount factor, ℓ^2_τ the expectile loss, and $g_s \in \mathcal{S}$ a waypoint to the goal.

Low-Level Policy. We learn the option embeddings ϕ_ω jointly with the low-level policy. As with the value function, ARL is agnostic to the choice of policy and policy extraction algorithm. In our implementations we use Advantage-Weighted Regression (AWR) [8]:

$$\mathcal{L}_{\pi_l,\phi_\omega} = -\mathbb{E}_{(s,a,s'\cdots g_s)\sim\mathcal{D}}\left[e^{\alpha_l A_l(s,a,s',g_s)}\log\pi_l\left(a \mid s, \hat{\phi}_\omega(s,g_s)\right)\right], \tag{4}$$

where $A_l(s,a,s',g_s) = Q_l(s,g_s,a) - V_l(s,g_s)$ represents the advantage associated with action a.

High-Level Value. To stabilise training and mitigate the issue of simultaneously learning the option representation, which can lead to training instability, we learn an action-free high-level value function, which can be learned directly from state trajectories without requiring explicit option labels. Note that, apart from being action-free, ARL is agnostic to the choice of high-level value learning and could be implemented with value horizon reduction such as TD-n or TRL [48]. In our implementations, we use one-step IVL [49, 50]:

$$\mathcal{L}_{V_h} = \mathbb{E}_{(s,a)\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\ell^2_\tau\left(V_h(s,g) - r(s,g) - \gamma\tilde{V}_h(s',g))\right)\right], \tag{5}$$

where $\tilde{V}_h$ represents the high-level target network. Although this biases the high-level value function towards being optimistic in stochastic environments, future work could incorporate a notion of reachability from the low-level value function.

High-Level Policy. Again, ARL is agnostic to the choice of high-level policy extraction. We hypothesise the high-level policy to be multi-modal, corresponding to distinct and equally optimal options, but note that choice of high-level policy is orthogonal to this work. A high-level Q-function could also be fitted to the high-level value function, which would enable use of Behaviour-Cloned Deep Deterministic Policy Gradient (DDPGBC) for policy extraction [28], for example; we include details in Appendix C. In our implementations, we use a Gaussian high-level policy, which we generally (see Appendix E) train using AWR:

$$\mathcal{L}_{\pi_h} = -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}_{\gamma}(\cdot|s)}\left[e^{\alpha_h A_h(s,\hat{\phi}_\omega(s,g_s),g_s,g)}\log\pi_h\left(\hat{\phi}_\omega(s,g_s) \mid s, g\right)\right], \tag{6}$$

where A_h represents the advantage associated with option $\hat{\phi}_\omega(s,g_s)$. We provide pseudocode in Algorithm 1 in Appendix C. Full experiment details, hyperparameters, sampling methods and seeds are found in Appendices C and E, and in our codebase.

4.3 Abstractive RL Explicitly Enforcing Translation Invariance

We now introduce a second algorithm, which explicitly imposes translation invariance on state-waypoint representations in the low-level decision process in order to learn from similar contexts across the state-space. We hypothesise that such an inductive bias could be useful in manipulation tasks, for example. We propose this algorithm as a proof of concept that using different representations at different levels of the hierarchy can improve generalisation in Offline RL.

We define *relativised states* as an unnormalised displacement vector:

$$v = g_s - s,$$

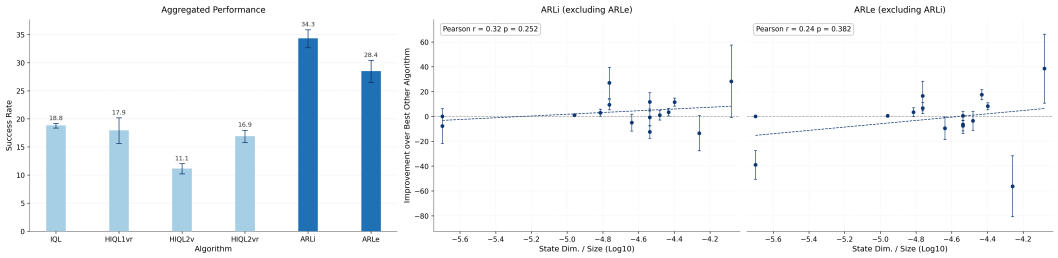

Figure 2: **Analysis.** Aggregate Performance across all tasks (**left**) and ARLi's (**middle**) and ARLe's (**right**) performance improvements over next-best performing algorithm against number of state dimensions per dataset sample. Bootstrapped 95% CI over 4 seeds and 20 evaluation runs.

resulting in a single vector that simultaneously encodes both the state and waypoint.

Since hard-coding representations can impose representational constraints, our approach exploits the two-step procedure of Offline RL as a compromise. To enforce experience reuse, we define the low-level value function in terms of relativised states: $V_l(g_s - s)$. Although this introduces a representational constraint (where state-waypoint pairs with distinct values may map to the same relative vector[1]), it explicitly collapses the state-waypoint space into a manifold of relative displacements, addressing the second and fourth points in the Motivation Box (Section 4.1). We remark that the issue of representational constraints could also be mitigated by computing the difference of encoded representations such that $v = \phi_{l_{g_s}}(g_s) - \phi_{l_s}(s)$, although we did not find this to be necessary to achieve superior performance in our experiments.

As in Section 4.2, to relativise options and address the third point in the Motivation Box, option-embeddings are learned with the low-level policy. However, now options are also explicitly relativised by defining them in terms of relativised states. We use soft-normalisation rather than length-normalisation to avoid numerical instability while introduce geometric regularisation and allow re-normalising of samples from the high-level policy [24] upon deployment:

$$o := \hat{\phi}_\omega(s, g_s) = \frac{\phi_\omega(g_s - s) \cdot \tanh ||\phi_\omega(g_s - s)||}{||\phi_\omega(g_s - s)||} \cdot \sqrt{d},$$

where d is the dimension of the embedding ϕ_ω, and $|| \cdot ||$ denotes the standard Euclidean norm. Soft normalisation means that the option space includes the space within the hypersphere and allows options to incorporate implicit temporal awareness as their magnitude scales linearly when displacement is small.

To satisfy local constraints, we still condition the low-level policy on the absolute state: $\pi_l(\cdot \mid s, \hat{\phi}_\omega(s, g_s))$. We provide pseudocode in Algorithm 2 in Appendix C. Full experiment details, hyperparameters, sampling methods and seeds are found in Appendices C and E and in our codebase.

5 Experiments

The goal of our experiments is simple: to test whether relativised options and distinct representations at different levels of the hierarchy can lead to better policy generalisation in offline GCRL.

Benchmark and Ablations. We perform all experiments on the standard OGBench datasets, focusing on the more challenging locomotion and manipulation environments (i.e. selecting *giant* over *medium* or *large*). For completeness, we also evaluate on a stochastic setting (*teleport*), despite its mismatch with our deterministic assumption (Section 2). We exclude visual environments as they introduce additional challenges related to high-dimensional perception that are orthogonal to this work. Unlike prior work [12, 48], we do not use oracle representations, which simplify option learning in locomotion, and discard proprioceptive information that might be useful in manipulation.

We benchmark ARL with implicitly learned relativised options (Section 4.2, **ARLi**), and explicitly enforced translation invariance (Section 4.3, **ARLe**), against the original version of HIQL [24]

[1]For instance, in a maze, a state s and waypoint g_s separated by a wall may map to the same relative vector as a pair in open space, yet induce vastly different value estimates.

(**HIQL1vr**), which uses a single value function and learns option representations via this value function. To isolate the effect of the relativised representation rather than any differences arising due to structure of the value function (ARL employs two value functions), we compare against variants of HIQL that also use two value functions. We include a variant with two value functions that does not include option representation learning (**HIQL2v**), and a variant with two value functions that learns option representations via the low-level value function (**HIQL2vr**), with the intention of mirroring HIQL1vr. Finally, we also compare against goal-conditioned IQL [7], the best-performing flat policy from Park et al. [5] (**IQL**). We refer to Appendix C and our codebase for full implementation details.

Since hyperparameter tuning is expensive and ARL, which is based on inductive biases, introduces no additional hyperparameters over HIQL, we simply adopt those tuned for HIQL [5, 12] (see Appendix E). The fact that ARL achieves strong performance under these hyperparameters attests to its efficacy. ARL is agnostic to the choice of policy class and value learning objective, so, to avoid related confounding factors, we use a Gaussian and one-step TD for all experiments.

Results. Our results (Table 1 and left of Figure 2) show that both variants of ARL outperform both the flat and absolute hierarchical policies, achieving a mean success rate that is at least 10 percentage points higher than the benchmarks when aggregating over all tasks. We highlight that this is without necessitating hyperparameter tuning.

Simply by learning relativised options based on action similarity rather than value similarity, ARLi consistently improves over the hierarchical benchmarks (excluding ARLe). In particular, it more than doubles the success rate in both the *humanoidmaze* environments, and almost doubles it for *scene-play-v0*.

ARLe performs especially well in the manipulation environments. Most notably, in *puzzle-4x4-play-v0*, an 83-dimensional manipulation task, ARLe achieves an 88% success rate, outperforming the next-best benchmark by 39 percentage points (excluding ARLi). We hypothesise that translational invariance is particularly beneficial in high-dimensional settings with underlying symmetries and sparse state-space coverage. To investigate this, we plot improvement over the next-best benchmark[2] against state dimensionality normalised by dataset size (right of Figure 2). Although correlation does not imply causation, and the observed correlations are weak and not statistically significant (amplified by a small number of environments[3]), both ARLi and ARLe exhibit positive improvement trends with increasing dimensional sparsity. In comparison, HIQL1vr and HIQL2vr, for example, show stronger negative trends with increasing sparsity (Figure 6 in Appendix D) under the same methodology. Intuitively, it makes sense that relativised options and experience reuse become increasingly important as sparsity increases.

To better understand the effect of imposing translational invariance, we visualise the low-level value functions for ARLe and HIQL2v in the *antmaze* locomotion environment (Figure 3). Due to explicitly collapsing equivalent relative states, ARLe learns a substantially smoother low-level value function. We now address potential questions.

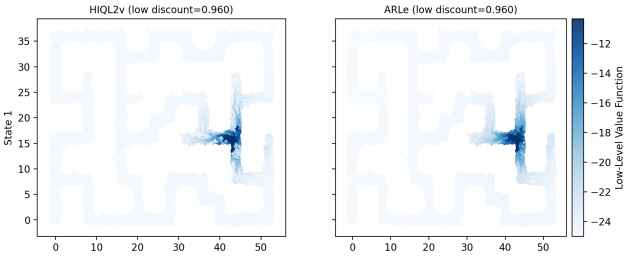

Figure 3: **Low-level** value function for **HIQL2v** (**left**) and **ARLe** (**right**) (task 4, *antmaze-giant-stitch-v0*).

Why not tune hyperparameters for ARL? Offline RL typically requires significant online tuning [52], which is expensive and would be unscalable for training a billion-parameter general-purpose agent [53, 54]. By using inductive biases rather than representational losses, we avoid introducing additional hyperparameters: ARL achieving significant performance gains under hyperparameters tuned for HIQL indicates robustness.

Why introduce ARLe if ARLi is so consistent? ARLe demonstrates how hierarchical structures and relativised options enable level-specific representations, leveraging this inductive bias to improve

[2]We try to mitigate confounding factors such as horizon length, absolute dataset size and policy expressivity (e.g. whether unimodal or multimodal) by plotting performance gains over the next-best algorithm rather than absolute success rate.

[3]Note that this is a common issue in Deep RL, and, like prior work [51], we emphasise that lack of statistically significant results does not demonstrate the absence of effect.

Table 1: **Results.** We report each method's average (binary) success rate (%) across the five test-time goals. Bootstrapped 95% CI over 4 seeds and 20 evaluation runs. Blue bold indicates the highest mean; black bold overlapping confidence intervals.

Task	Size	Dim.	IQL	HIQL1vr	HIQL2v	HIQL2vr	ARLi	ARLe
pointmaze-giant-navigate-v0	1M	2	0±0	**55±11**	21±7	**46±6**	**48±8**	16±2
pointmaze-giant-stitch-v0	1M	2	0±0	**0±0**	0±0	0±0	0±0	0±0
antmaze-giant-navigate-v0	1M	29	0±0	38±3	**40±3**	**48±6**	**48±3**	**42±4**
antmaze-giant-stitch-v0	1M	29	0±0	7±7	15±3	20±3	**32±7**	21±1
antmaze-teleport-stitch-v0	1M	29	**49±2**	29±4	43±3	40±6	36±5	41±4
humanoidmaze-giant-navigate-v0	4M	69	1±1	22±11	12±5	11±10	**49±6**	38±4
humanoidmaze-giant-stitch-v0	4M	69	0±0	4±3	0±0	0±0	**13±2**	**10±3**
cube-double-play-v0	1M	37	50±3	2±0	0±0	3±0	53±2	**67±3**
cube-triple-play-v0	3M	46	**11±2**	7±3	0±0	1±1	**14±2**	**15±3**
cube-quadruple-play-v0	5M	55	0±0	0±0	0±0	0±0	**1±0**	0±0
puzzle-3x3-play-v0	1M	55	**100±0**	27±8	0±0	24±3	**86±14**	44±25
puzzle-4x4-play-v0	1M	83	30±3	49±27	0±0	17±6	**78±11**	**88±6**
puzzle-4x5-play-v0	3M	99	**15±3**	**17±3**	0±0	**12±3**	**18±3**	**14±7**
puzzle-4x6-play-v0	5M	115	13±1	0±0	0±0	**18±2**	**14±7**	**9±9**
scene-play-v0	1M	40	13±2	12±3	12±7	13±1	**24±3**	**21±3**

performance by 10 percentage points over ARLi in two out of the four sparsest datasets (*puzzle-4x4-play-v0* and *cube-double-play-v0*).

Why does ARLe perform poorly in certain tasks? ARLe's performance depends on the alignment between its inductive bias and the environment's structure. Assuming local translational invariance can benefit high-dimensional manipulation through experience reuse but can be detrimental in dense, low-dimensional environments like *pointmaze*, since the representation is inherently lossy. We also hypothesise that mapping 2D relative displacements into a 10D latent hypersphere introduces representational noise. While future work could leverage GDL to learn more flexible symmetries, ARLe demonstrates that decoupling representations across the hierarchy enables a degree of experience reuse fundamentally inaccessible to flat or absolute-frame architectures. When the inductive bias is well-matched to the environment, it significantly enhances policy generalisation.

Why do all algorithms perform poorly in certain tasks? We hypothesise this to be due to other confounding factors such as: (i) poor high-level value learning and a lack of gradient in long horizon tasks (we include plots of the high-level value function in Figure 5 in Appendix D); (ii) the high-level policy being unimodal rather than multimodal; (iii) not training for enough gradient steps for large dataset sizes; (iv) not learning goal representations for the high-level policy. The aforementioned issues could be mitigated by combining ARL with value horizon reduction methods such as TD-n or TRL [48], learning a flow-policy [55] rather than a Gaussian (especially for the high-level policy), training for more steps (we run all experiments for 1M), and using, for example, Dual-Goal Representations [56] for the high-level decision process.

6 Conclusion

In this work we motivate hierarchy in offline RL through *absolute* abstraction. By learning relativised options and using distinct representations at different levels of the hierarchy, agents can reuse optimal experience across similar contexts of the state-space, enabling better performance in regions of the dataset only supported by low-quality data. Based on our framework, we introduce two simple algorithms for learning relativised options via action similarity and explicitly enforcing translational invariance on the low-level decision process. Our experiments demonstrate that such relativised options and inductive biases improve policy generalisation in high-dimensional offline GCRL. This proof of concept opens many avenues for future research, including imposing more flexible inductive biases, or leveraging action-chunking to learn relativised options over action sequences. We hope that this work motivates progress towards scalable offline RL.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgements

The authors would like to thank **Michael Matthews** and **Zilin Wang** for their helpful advice and comments on the project.

Clarisse Wibault is funded by EPSRC grant EP/W524311/1. **Alexander Goldie** is funded by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems EP/S024050/1. **Jakob Foerster** is partially funded by the UKRI grant EP/Y028481/1, as well as being supported by the JPMC and Amazon Research Awards.

References

[1] Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1094–1099, 1993.

[2] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, 2015.

[3] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems, November 2020. URL `http://arxiv.org/abs/2005.01643`. arXiv:2005.01643 [cs].

[4] Sergey Levine. Understanding the World Through Action, October 2021. URL `http://arxiv.org/abs/2110.12543`. arXiv:2110.12543 [cs].

[5] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking Offline Goal-Conditioned RL, February 2025. URL `http://arxiv.org/abs/2410.20092`. arXiv:2410.20092 [cs].

[6] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(8):10237–10257, August 2024. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2023.3250269. URL `http://arxiv.org/abs/2203.01387`. arXiv:2203.01387 [cs].

[7] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning, October 2021. URL `http://arxiv.org/abs/2110.06169`. arXiv:2110.06169 [cs].

[8] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning, October 2019. URL `http://arxiv.org/abs/1910.00177`. arXiv:1910.00177 [cs].

[9] Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the Minimalist Approach to Offline Reinforcement Learning, October 2023. URL `http://arxiv.org/abs/2305.09836`. arXiv:2305.09836 [cs].

[10] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of Real-World Reinforcement Learning, April 2019. URL `http://arxiv.org/abs/1904.12901`. arXiv:1904.12901 [cs].

[11] Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is Value Learning Really the Main Bottleneck in Offline RL?, October 2024. URL `http://arxiv.org/abs/2406.09329`. arXiv:2406.09329 [cs].

[12] Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon Reduction Makes RL Scalable, October 2025. URL http://arxiv.org/abs/2506.04168. arXiv:2506.04168 [cs].

[13] 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, August 1999. ISSN 00043702. doi: 10.1016/S0004-3702(99)00052-1. URL https://linkinghub.elsevier.com/retrieve/pii/S0004370299000521.

[14] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3540–3549, 2017.

[15] Balaraman Ravindran and Andrew G. Barto. Model minimization in hierarchical reinforcement learning.

[16] Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies, December 2025. URL http://arxiv.org/abs/2506.07339. arXiv:2506.07339 [cs].

[17] Kwanyoung Park, Seohong Park, Youngwoon Lee, and Sergey Levine. Scalable Offline Model-Based RL with Action Chunks, December 2025. URL http://arxiv.org/abs/2512.08108. arXiv:2512.08108 [cs].

[18] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[19] Arsh Tangri, Nichols Crawford Taylor, Haojie Huang, and Robert Platt. Equivariant goal conditioned contrastive reinforcement learning. 2025. doi: 10.48550/arXiv.2507.16139.

[20] Sascha Lange, Thomas Gabel, and Martin A. Riedmiller. Batch reinforcement learning. Springer, 2012.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

[22] Adam White, Joseph Modayil, and Richard S. Sutton. Scaling life-long off-policy learning. *CoRR*, abs/1206.6262, 2012. URL http://arxiv.org/abs/1206.6262.

[23] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay, February 2018. URL http://arxiv.org/abs/1707.01495. arXiv:1707.01495 [cs].

[24] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline Goal-Conditioned RL with Latent States as Actions, March 2024. URL http://arxiv.org/abs/2307.11949. arXiv:2307.11949 [cs].

[25] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning, August 2020. URL http://arxiv.org/abs/2006.04779. arXiv:2006.04779 [cs].

[26] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-Based Offline Reinforcement Learning with Diversified Q-Ensemble, October 2021. URL http://arxiv.org/abs/2110.01548. arXiv:2110.01548 [cs].

[27] Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive Learning as Goal-Conditioned Reinforcement Learning, February 2023. URL http://arxiv.org/abs/2206.07568. arXiv:2206.07568 [cs].

[28] Scott Fujimoto and Shixiang Shane Gu. A Minimalist Approach to Offline Reinforcement Learning, December 2021. URL http://arxiv.org/abs/2106.06860. arXiv:2106.06860 [cs].

[29] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online RL. *CoRR*, abs/2007.11091, 2020. URL https://arxiv.org/abs/2007.11091.

[30] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The Option-Critic Architecture, December 2016. URL http://arxiv.org/abs/1609.05140. arXiv:1609.05140 [cs].

[31] Seungho Baek, Taegeon Park, Jongchan Park, Seungjun Oh, and Yusung Kim. Graph-Assisted Stitching for Offline Hierarchical Reinforcement Learning, June 2025. URL http://arxiv.org/abs/2506.07744. arXiv:2506.07744 [cs] version: 1.

[32] Richard S Sutton, Doina Precup, and Satinder Singh. Intra-Option Learning about Temporally Abstract Actions.

[33] Balaraman Ravindran and Andrew G. Barto. Smdp homomorphisms: An algebraic approach to abstraction in semi-markov decision processes. In *Probabilistic Planning*, pages 1011–1016, 2003.

[34] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning, October 2018. URL http://arxiv.org/abs/1805.08296. arXiv:1805.08296 [cs].

[35] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-Optimal Representation Learning for Hierarchical Reinforcement Learning, January 2019. URL http://arxiv.org/abs/1810.01257. arXiv:1810.01257 [cs].

[36] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning Multi-Level Hierarchies with Hindsight, September 2019. URL http://arxiv.org/abs/1712.00948. arXiv:1712.00948 [cs].

[37] Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 361–368, 2001.

[38] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation (SARA)*, pages 212–223, 2002.

[39] Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning, 2022. URL https://arxiv.org/abs/2205.11790.

[40] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a Unified Theory of State Abstraction for MDPs.

[41] Norman Ferns, Prakash Panangaden, and Doina Precup. Metrics for Finite Markov Decision Processes, July 2012. URL http://arxiv.org/abs/1207.4114. arXiv:1207.4114 [cs].

[42] Jianda Chen and Sinno Jialin Pan. Learning Representations via a Robust Behavioral Metric for Deep Reinforcement Learning.

[43] Ayoub Echchahed and Pablo Samuel Castro. A Survey of State Representation Learning for Deep Reinforcement Learning, June 2025. URL http://arxiv.org/abs/2506.17518. arXiv:2506.17518 [cs].

[44] Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. Phd thesis, University College London, 2003.

[45] Remi Munos, Remi Munos, and Csaba Szepesvari. Finite-Time Bounds for Fitted Value Iteration.

[46] Tor Lattimore and Marcus Hutter. PAC Bounds for Discounted MDPs, February 2012. URL http://arxiv.org/abs/1202.3890. arXiv:1202.3890 [cs].

[47] Arnaud Robert, Ciara Pike-Burke, and A Aldo Faisal. Sample Complexity of Goal-Conditioned Hierarchical Reinforcement Learning.

[48] Seohong Park, Aditya Oberai, Pranav Atreya, and Sergey Levine. Transitive RL: Value Learning via Divide and Conquer, February 2026. URL http://arxiv.org/abs/2510.22512. arXiv:2510.22512 [cs].

[49] Dibya Ghosh, Chethan Bhateja, and Sergey Levine. Reinforcement Learning from Passive Data via Latent Intentions, April 2023. URL http://arxiv.org/abs/2304.04782. arXiv:2304.04782 [cs].

[50] Haoran Xu, Li Jiang, Jianxiong Li, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning, 2023. URL https://arxiv.org/abs/2210.08323.

[51] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

[52] Matthew Thomas Jackson, Uljad Berdica, Jarek Liesen, Shimon Whiteson, and Jakob Nicolaus Foerster. A Clean Slate for Offline Reinforcement Learning, April 2025. URL http://arxiv.org/abs/2504.11453. arXiv:2504.11453 [cs].

[53] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

[54] Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzciński, and Benjamin Eysenbach. 1000 Layer Networks for Self-Supervised RL: Scaling Depth Can Enable New Goal-Reaching Capabilities, February 2026. URL http://arxiv.org/abs/2503.14858. arXiv:2503.14858 [cs].

[55] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow Matching Guide and Code, December 2024. URL http://arxiv.org/abs/2412.06264. arXiv:2412.06264 [cs].

[56] Seohong Park, Deepinder Mann, and Sergey Levine. Dual Goal Representations, February 2026. URL http://arxiv.org/abs/2510.06714. arXiv:2510.06714 [cs].

[57] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[58] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[59] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

[60] Adam Villaflor, Zhe Huang, Swapnil Pande, John Dolan, and Jeff Schneider. Addressing optimism bias in sequence modeling for reinforcement learning, 2022. URL https://arxiv.org/abs/2207.10295.

A Sample Complexity in Online Goal-Conditioned RL

We provide some intuition into the choice of policy learning and representation using sample complexity in finite state and action space problems. We build on the works of [44–47, 40].

In online GCRL, the aim is to find the optimal policy with the smallest number of samples or online interactions, N, such that the error in optimal return is smaller than a fixed constant ϵ.

A.1 GCRL Sample Complexity

Consider the case of a discrete, discounted horizon MDP with a finite state space $\mathcal{S}$, action space $\mathcal{A}$ and discount factor $\gamma \in [0, 1)$. The Probably-Approximately Correct (PAC) Learning [44] upper-bound sample complexity to find an ϵ optimal policy reaching a unique goal-state optimally (assuming at most two possible next-states for each state/action pair) with constant probability of $1 - \delta$ is given by [46]:

$$N^{\text{infinite, single goal}} \propto \frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}.$$

Hence, the minimax sample complexity required to find an ϵ optimal policy reaching any given state (such that we have $|\mathcal{G}|$ unique goal-states) is given by:

$$N^{\text{infinite}} \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}. \tag{7}$$

Consider the case of a discrete finite horizon (of length H) MDP with a finite state space $\mathcal{S}$, action space $\mathcal{A}$ and discount factor $\gamma \in [0, 1)$. The minimax sample complexity to find an ϵ optimal policy reaching a unique goal-state optimally is given by:

$$N^{\text{finite}} \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|H^3}{\epsilon^2}.$$

A.2 GCRL Hierarchical Sample Complexity

Using a hierarchical policy, we can break the distant goal g from our current state s into options defined over an option space Ω. Hence, the high-level policy becomes

$$N^{\text{high}} \propto \frac{|\mathcal{S}||\mathcal{G}||\Omega|}{(1-\gamma^n)^3\epsilon^2},$$

where we have substituted the option-space to Equation 7, and use the environment's discount factor raised to a factor of n, assuming that the high-level policy acts, on average, every n steps.

The low-level policy has a sample complexity of

$$N^{\text{low}} \propto \frac{|\mathcal{S}||\Omega||\mathcal{A}|n^3}{\epsilon^2},$$

since $|\Omega|$ options can be executed. Hence, the overall sample complexity of the hierarchical policy $\pi(a \mid s, g) := \pi_l(a \mid s, \omega), \quad \omega \sim \pi_h(\cdot \mid s, g)$ is [47]:

$$N^{\text{hierarchy}} \propto \frac{|\mathcal{S}||\mathcal{G}||\Omega|}{(1-\gamma^n)^3\epsilon^2} + \frac{|\mathcal{S}||\Omega||\mathcal{A}|n^3}{\epsilon^2}. \tag{8}$$

Note, that, in the case of no temporal abstraction, when $n = 1$, we approximately recover the original sample complexity of a flat policy. Since then the option just becomes a single primitive action, Equation 8 becomes:

$$N^{\text{hierarchy}} \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2} + \frac{|\mathcal{S}||\mathcal{A}|^2}{\epsilon^2} \approx \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2},$$

where the approximation follows due to the dominating $\frac{1}{(1-\gamma)^3}$ factor in the first term, assuming long-horizon problems such that $\gamma \to 1$ and that the goal-space is larger or equal to the size of the action space $|\mathcal{G}| \geq |\mathcal{A}|$ [4].

[4] Even though this might not be the case, usually the goal-space is unknown, so we train the policy to reach any state within the state-space i.e. such that $\mathcal{G} = \mathcal{S}$. Assuming that $|\mathcal{S}| \gg |\mathcal{A}|$ is a standard assumption.

Issues arise under a misspecified option horizon n. In this case, the sample complexity of the hierarchical policy becomes worse than that of a flat policy, as the skill space must then account for every sequence of primitive actions over n steps, such that $|\Omega| = |\mathcal{A}|$. The sample complexity of the hierarchical policy becomes

$$N^{\text{hierarchy}} \propto \frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}|^n}{(1-\gamma^n)^3\epsilon^2} + \frac{|\mathcal{S}||\Omega||\mathcal{A}|^n n^3}{\epsilon^2},$$

which blows up due to the exponential factor multiplying the action space.

A.3 GCRL State Representation Sample Complexity

State representation sample complexity exploits symmetry in the state-goal space to a mapping $\phi(s, g) \to c$ such that $\phi(s, g) = \phi(s', g')$ if $\pi^\star(a \mid s, g) = \pi^\star(a \mid s', g')$. The sample complexity of learning an optimal policy (Equation 7) becomes:

$$N^{\text{rep}} \propto \frac{|\mathcal{C}||\mathcal{A}|}{(1-\gamma)^3\epsilon^2}. \tag{9}$$

Note that $|\mathcal{C}| \leq |\mathcal{S}||\mathcal{G}|$, with $|\mathcal{C}| = |\mathcal{S}||\mathcal{G}|$ if each (s, g) has a distinct optimal action.

B Error in Offline Goal-Conditioned RL

In offline GCRL, the aim is to bound the error ϵ given an offline dataset $\mathcal{D}$ of fixed size N.

Unlike in the online setting, where it is assumed that the agent can sample any state-action pair to learn the environment's dynamics, in offline RL, a concentrability coefficient κ is incorporated to account for the distribution shift in data collected by the policy π^{BC}, and the data that would have been collected induced under the optimal policy $\pi^\star$. Intuitively, if the dataset does not include the states required to learn the optimal policy, the algorithm may never learn that optimal policy. The concentrability coefficient κ is defined as:

$$\kappa = \sup_{s\in\mathcal{S}, a\in\mathcal{A}, g\in\mathcal{G}} \frac{d^{\pi^\star}(s, a, g)}{d^{\pi^{\text{BC}}}(s, a, g)},$$

where $d^\pi(s, a, g)$ is the discounted occupancy measure (i.e. stationary distribution) of the policy π:

$$d^\pi(s, a, g) = (1-\gamma)\mathbb{E}_{\tau\sim\pi, s_0, g_0\sim\text{Unif}(\mathcal{S})}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}(s_t = s, a_t = a, g_0 = g)\right],$$

and where the trajectory is induced by the MDP and following the policy $\pi(a \mid s, g)$. If the dataset is highly exploratory and covers the optimal paths well, κ will be small. If the dataset is narrow, or misses critical regions of the state-action space, κ will be large.

Rearranging Equation 7 and incorporating the concentrability coefficient, the offline bound for a goal-conditioned flat policy is given by:

$$\epsilon \propto \sqrt{\frac{|\mathcal{S}||\mathcal{G}||\mathcal{A}| \cdot \kappa}{(1-\gamma)^3 N}}.$$

B.1 GCRL Hierarchical Error

Since the size of the offline dataset is fixed, the only way to reduce the error ϵ is to use a hierarchical policy with distinct state-goal representations.

Using a hierarchical policy, the error term becomes:

$$\epsilon^{\text{hierarchy}} \propto \sqrt{\frac{|\mathcal{S}||\mathcal{G}||\Omega| \cdot \kappa_h}{(1-\gamma^n)^3 N}} + \sqrt{\frac{|\mathcal{S}||\Omega||\mathcal{A}| n^3 \cdot \kappa_l}{N}},$$

where the concentrability coefficients are defined as:

$$\kappa_h = \sup_{s\in\mathcal{S}, \omega\in\Omega, g\in\mathcal{S}} \frac{d^{\pi_h^\star}(s, \omega, g)}{d^{\pi_h^{\text{BC}}}(s, \omega, g)} \quad \text{and} \quad \kappa_l = \sup_{s\in\mathcal{S}, a\in\mathcal{A}, \omega\in\Omega} \frac{d^{\pi_l^\star}(s, a, \omega)}{d^{\pi_l^{\text{BC}}}(s, a, \omega)}.$$

Then, introducing two embeddings that group together state-goal pairs (s, g) requiring similar options ω such that $\phi_h(s, g) = \phi_h(s', g') = c_h$ if $\pi_h^\star(\cdot \mid s, g) = \pi_l^\star(\cdot \mid s', g')$ and $c_h \in \mathcal{C}_h$, and state-option (s, ω) requiring similar low-level actions $\phi_l(s, \omega) = \phi_l(s', \omega') = c_l$ if $\pi_l^\star(\cdot \mid s, \omega) = \pi_l^\star(\cdot \mid s', \omega')$ [5] and $c_l \in \mathcal{C}_l$, the error term is bounded by:

$$\epsilon^{\text{hierarchy, rep}} \propto \sqrt{\frac{|\mathcal{C}_h||\Omega| \cdot \kappa_h^{\text{rep}}}{(1-\gamma^n)^3 N}} + \sqrt{\frac{|\mathcal{C}_l||\mathcal{A}| n^3 \cdot \kappa_l^{\text{rep}}}{N}}.$$

By definition, $|\mathcal{C}_h||\Omega| \leq |\mathcal{S}||\mathcal{G}||\Omega|$ and $|\mathcal{C}_l||\mathcal{A}| \leq |\mathcal{S}||\Omega||\mathcal{A}|$.

Because the concentrability coefficients are now defined using the reparameterised policies over the reparameterised latent spaces $\mathcal{C}_h$ and $\mathcal{C}_l$, i.e.

$$\kappa_h^{\text{rep}} = \sup_{c_h \in \mathcal{C}_h, \omega \in \Omega} \frac{d^{\pi_h^\star}(c_h, \omega)}{d^{\pi_h^{\text{BC}}}(c_h, \omega)} \quad \text{and} \quad \kappa_l^{\text{rep}} = \sup_{c_l \in \mathcal{C}_l, a \in \mathcal{A}} \frac{d^{\pi_l^\star}(c_l, a)}{d^{\pi_l^{\text{BC}}}(c_l, a)},$$

the probability mass of the offline dataset is aggregated across state-goal or state-option pairs that are equivalent under these embeddings:

$$d^{\pi_h}(c_h, \omega) = \sum_{(s,g) \in \phi_h^{-1}(c_h)} d_h^\pi(s, g, \omega) \quad \text{and} \quad d^{\pi_l}(c_l, a) = \sum_{(s,\omega) \in \phi_l^{-1}(c_l)} d_l^\pi(s, \omega, a).$$

This reduces the likelihood of a support mismatch, where the optimal policy requires a state transition on which the dataset places zero mass. Hence,

$$\kappa_h^{\text{rep}} \leq \kappa_h \quad \text{and} \quad \kappa_l^{\text{rep}} \leq \kappa_l.$$

[5] Note that we should not use these same embeddings for the value or critic functions, since even though $\pi_l^\star(\cdot \mid s, \omega) = \pi_l^\star(\cdot \mid s', \omega')$, this does not instantly imply that $Q_h(s, g, \omega) = Q_h(s', g', \omega')$: even though the ordering over actions might be the same, there might be an offset such that $Q_h(s, g, \omega) = Q_h(s', g', \omega) + c(s, g, s', g') \quad \forall \quad \omega$. We refer the reader to Li et al. [40].

C Offline RL Algorithms

In the following section g_s is a waypoint to the goal, such that $g_s \in \mathcal{S}$. When sampled from the dataset $\mathcal{D}$, g_s is n steps ahead of the current state s. When sampling the waypoint g_s or goal g from $p^{\mathcal{D}}(\cdot \mid s, a)$, we sample either from a geometric distribution according to the specified discount factor, or a uniform distribution. Details are given in Appendix E.

C.1 Implicit Q-Learning (IQL)

The flat policy we benchmark is IQL [7], which trains a state-goal value function $V(s, g)$ and state-goal-action value function $Q(s, g, a)$ using the following losses:

$$\mathcal{L}_V = \mathbb{E}_{(s,a)\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\ell_\tau^2\left(V(s,g) - \tilde{Q}(s,g,a)\right)\right],$$

where $\tilde{Q}$ denotes the target network, and ℓ_τ^2 denotes the expectile loss $\ell_\tau^2(x) = |\tau - (x < 0)|x^2$, and

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,s')\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\left(Q(s,a,g) - r(s,g) - \gamma V(s',g)\right)^2.\right]$$

A Gaussian policy is then extracted using the following DDPGBC [28] loss:

$$\mathcal{L}_\pi^{\text{DDPGBC}} = -\mathbb{E}_{(s,a,s')\sim\mathcal{D}, g\sim p_\gamma^{\mathcal{D}}(\cdot|s,a)}\left[Q(s,g,\mu^\pi(s,g)) + \alpha \log \pi(a \mid s, g)\right], \tag{10}$$

which has been found to outperform AWR [11].

C.2 Hierarchical Implicit Q-Learning 1 Value Function with Representation Learning (HIQL1vr)

We benchmark against HIQL [24], which trains a single, action-free state-goal value function $V(s, g)$ using Implicit V-Learning (IVL) [5] and extracts hierarchical policies using AWR-like objectives. The parameterisation ϕ_ω for the low-level policy is learned with the value function $V(s, \hat{\phi}_\omega(s, g))$, where $\phi_\omega(s, g) \in \mathbb{R}^d$, and normalised such that $||\hat{\phi}_\omega(s, g)||_2^2 = d$, where d is the dimension of the embedding. The IVL loss is given by:

$$\mathcal{L}_{V,\phi_\omega} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\ell_\tau^2\left(V(s,\hat{\phi}_\omega(s,g)) - r(s,g) - \gamma\tilde{V}(s',\tilde{\hat{\phi}}_\omega(s',g))\right)\right],$$

where $\tilde{V}$ and $\tilde{\hat{\phi}}_o$ denote the target network and representations and ℓ_τ^2 the expectile loss, as before. The low-level and high-level policies are extracted as follows:

$$\mathcal{L}_{\pi_h} = -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g\sim p_\gamma^{\mathcal{D}}(\cdot|s,a))}\left[e^{\alpha_h(V(g_s,\hat{\phi}_\omega(g_s,g)) - V(s,\hat{\phi}_\omega(s,g))}\log \pi_h\left(\hat{\phi}_\omega(s,g_s) \mid s, g\right)\right],$$

$$\mathcal{L}_{\pi_l} = -\mathbb{E}_{(s,a,s'\cdots g_s)\sim\mathcal{D}}\left[e^{\alpha_l(V(s',\hat{\phi}_\omega(s',g_s)) - V(s,\hat{\phi}_\omega(s,g_s)))}\log \pi_l\left(a \mid s, \hat{\phi}_\omega(s,g_s)\right)\right].$$

Since Park et al. [11] found that DDPGBC is better at extracting a policy than AWR, and similarly to Park et al. [24]'s action-free value function, we then fit a sort of high-level, action-free Q function to the value function:

$$\mathcal{L}_{Q_h} = \mathbb{E}_{(s,\cdots,g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\left(Q_h(s,g,g_s) - V_h(g_s,g)\right)^2\right].$$

This is simply to allow some extrapolation during extraction of the high-level policy.

C.3 Hierarchical Implicit Q-Learning 2 Value Functions (HIQL2v)

Since ARL uses two value functions, we also train a second style of HIQL, where we use a low-level and high-level value function. The high-level value function is trained using IVL, while the low-level value function is trained using IQL:

$$\mathcal{L}_{V_h} = \mathbb{E}_{(s,a)\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\ell_\tau^2\left(V_h(s,g) - r(s,g) - \gamma\tilde{V}_h(s',g))\right)\right],$$

and

$$\mathcal{L}_{V_l} = \mathbb{E}_{(s,a)\sim\mathcal{D}, g_s\sim p_{\gamma_l}^{\mathcal{D}}(\cdot|s,a))}\left[\ell_\tau^2\left(V_l(s,g_s) - \tilde{Q}_l(s,g_s,a)\right)\right],$$

$$\mathcal{L}_{Q_l} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}, g_s\sim p_{\gamma_l}^{\mathcal{D}}(\cdot|s,a))}\left[\left(Q_l(s,g_s,a) - r(s,g_s) - \gamma_l V_l(s',g_s)\right)^2\right],$$

where $\tilde{Q}_l$ and $\tilde{V}_h$ denote the target networks, $\gamma_l = 1 - \frac{1}{n}$ denotes the low-level discount factor, and ℓ^2_τ denotes the expectile loss $\ell^2_\tau(x) = |\tau - (x < 0)|x^2$.

As for HIQL1vr, a high-level Q function is then learned only to allow DDPGBC high-level policy extraction:

$$\mathcal{L}_{Q_h} = \mathbb{E}_{(s,\cdots,g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\left(Q_h(s,g,g_s) - V_h(g_s,g)\right)^2\right].$$

Policies are then extracted using one of the following two loss functions for the low-level policy:

$$\mathcal{L}^{\text{AWR}}_{\pi_l} = -\mathbb{E}_{(s,a,s'\cdots g_s)\sim\mathcal{D}}\left[e^{\alpha_l(Q_l(s,g_s,a)) - V_l(s,g_s))}\log\pi_l\left(a \mid s, g_s\right)\right],$$
$$\mathcal{L}^{\text{DDPGBC}}_{\pi_l} = -\mathbb{E}_{(s,a,s',\cdots g_s)\sim\mathcal{D}}\left[Q_l(s,g_s,\mu^{\pi_l}(s,g_s)) + \alpha_l\log\pi_l(a \mid s, g_s)\right],$$

and one of the following two loss functions for the high-level policy:

$$\mathcal{L}^{\text{AWR}}_{\pi_h} = -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}_\gamma(\cdot|s)}\left[e^{\alpha_h(Q_h(s,g,g_s)) - V_h(s,g))}\log\pi_h\left(g_s \mid s, g\right)\right],$$
$$\mathcal{L}^{\text{DDPGBC}}_{\pi_h} = -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}_\gamma(\cdot|s)}\left[Q_h(s,g,\mu^{\pi_h}(s,g)) + \alpha_h\log\pi_h(g_s \mid s, g)\right].$$

C.4 Hierarchical Implicit Q-Learning 2 Value Functions with Representation Learning (HIQL2vr)

Since HIQL1vr uses representation learning for the options, HIQL2vr learns option representations with the low-level value function. As in HIQL1vr (Section C.2), the representation is length-normalised. The high-level value function is trained as before, but the low-level value function, low-level Q function and high-level Q-function are trained using the following losses:

$$\mathcal{L}_{V_l,\phi_\omega} = \mathbb{E}_{(s,a)\sim\mathcal{D}, g_s\sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\ell^2_\tau\left(V_l(s,\hat{\phi}_\omega(s,g_s)) - \tilde{Q}_l(s,g_s,a)\right)\right],$$
$$\mathcal{L}_{Q_l} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}, g_s\sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\left(Q_l(s,g_s,a) - r(s,g_s) - \gamma_l V_l(s',\hat{\phi}_\omega(s',g_s))\right)^2\right],$$
$$\mathcal{L}_{Q_h} = \mathbb{E}_{(s,\cdots,g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}(\cdot|s,a))}\left[\left(Q_h(s,g,\hat{\phi}_\omega(s,g_s)) - V_h(g_s,g)\right)^2\right].$$

The policies are extracted using one of the following two loss functions for the low-level policy,

$$\mathcal{L}^{\text{AWR}}_{\pi_l} = -\mathbb{E}_{(s,a,s'\cdots g_s)\sim\mathcal{D}}\left[e^{\alpha_l(Q_l(s,g_s,a)) - V_l(s,\hat{\phi}_\omega(s,g_s)))}\log\pi_l\left(a \mid s, \hat{\phi}_\omega(s,g_s)\right)\right],$$
$$\mathcal{L}^{\text{DDPGBC}}_{\pi_l} = -\mathbb{E}_{(s,a,s',\cdots g_s)\sim\mathcal{D}}\left[Q_l(s,g_s,\mu^{\pi_l}(s,\hat{\phi}_\omega(s,g_s))) + \alpha_l\log\pi_l(a \mid s, \hat{\phi}_\omega(s,g_s))\right],$$

and one of the following two loss functions for the high-level policy:

$$\mathcal{L}^{\text{AWR}}_{\pi_h} = -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}_\gamma(\cdot|s)}\left[e^{\alpha_h(Q_h(s,g,\hat{\phi}_\omega(s,g_s))) - V_h(s,g))}\log\pi_h\left(\hat{\phi}_\omega(s,g_s) \mid s, g\right)\right],$$
$$\mathcal{L}^{\text{DDPGBC}}_{\pi_h} = -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g\sim p^{\mathcal{D}}_\gamma(\cdot|s)}\left[Q_h(s,g,\mu^{\pi_h}(s,g)) + \alpha_h\log\pi_h(\hat{\phi}_\omega(s,g_s) \mid s, g)\right].$$

C.5 Abstractive Reinforcement Learning Implicitly Learning Relativised Options (ARLi)

As presented in the main body of the paper, this is a minimal amendment to HIQL2vr, but, to learn relativised options via action similarity, option representations are now learned with the low-level policy. Equations are identical to HIQL2vr, except for the following low-level value functions:

$$\mathcal{L}_{V_l} = \mathbb{E}_{(s,a)\sim\mathcal{D}, g_s\sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\ell^2_\tau\left(V_l(s,g_s)) - \tilde{Q}_l(s,g_s,a)\right)\right],$$
$$\mathcal{L}_{Q_l} = \mathbb{E}_{(s,a,s')\sim\mathcal{D}, g_s\sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\left(Q_l(s,g_s,a) - r(s,g_s) - \gamma_l V_l(s',g_s)\right)^2\right],$$

and low-level policy functions:

$$\mathcal{L}^{\text{AWR}}_{\pi_l,\phi_\omega} = -\mathbb{E}_{(s,a,s'\cdots g_s)\sim\mathcal{D}}\left[e^{\alpha_l(Q_l(s,g_s,a)) - V_l(s,g_s))}\log\pi_l\left(a \mid s, \hat{\phi}_\omega(s,g_s)\right)\right],$$
$$\mathcal{L}^{\text{DDPGBC}}_{\pi_l,\phi_\omega} = -\mathbb{E}_{(s,a,s',\cdots g_s)\sim\mathcal{D}}\left[Q_l(s,g_s,\mu^{\pi_l}(s,\hat{\phi}_\omega(s,g_s))) + \alpha_l\log\pi_l(a \mid s, \hat{\phi}_\omega(s,g_s))\right].$$

Algorithm 1 Abstractive RL implicitly learning relativised options (ARLi)

Training
Initialise low-level policy $\pi_l(a \mid s, \omega)$, high-level policy $\pi_h(\omega \mid s, g)$, and representation ϕ_ω.
while not converged **do**
  Sample batch $\mathcal{D}$
  ▷ **Hierarchical Policy**
    Update low-level policy π_l and representation $\phi_\omega$using $\pi_l(a \mid s, \hat{\phi}_\omega(s, g_s))$ (Equation 4)
    $\omega \leftarrow \text{stopgrad}(\hat{\phi}_\omega(s, g_s))$
    Update high-level policy $\pi_h(\omega \mid s, g)$ (Equation 6)
  ▷ **Hierarchical Value**
    Update low-level value function $V_l(s, g_s)$ (Equation 2)
    Update high-level value function $V_h(s, g)$ (Equation 5)
    Update critic $Q_l(s, g_s, a)$ (Equation 3)
end while
return $\pi_l, \pi_h, \phi_\omega$

Deployment (state s, goal g)
Sample option from high-level policy $\omega \sim \pi_h(\cdot \mid s, g)$
Length-Normalise $\omega \leftarrow \frac{\omega}{\|\omega\|} \cdot \sqrt{d}$
Sample action from low-level policy $a \sim \pi_l(\cdot \mid s, \omega)$

C.6 Abstractive Reinforcement Learning Explicitly Enforcing Translational Invariance (ARLe)

ARL uses relativised options and relativised states for the low-level value function. Unlike ARLi, the low-level value and low-level critic now use *relativised* goals, and options are explicitly relativised. As for ARLi, the option representations are learned with the low-level policy. The high-level value function is learned identically to HIQL2v and ARLi, but the low-level value function, low-level Q function and high-level Q function are learned as follows:

$$\begin{aligned}
\mathcal{L}_{V_l} &= \mathbb{E}_{(s,a)\sim\mathcal{D}, g_s \sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\ell^2_\tau\left(V_l(g_s - s) - \tilde{Q}_l(s, g_s - s, a)\right)\right], \\
\mathcal{L}_{Q_l} &= \mathbb{E}_{(s,a,s')\sim\mathcal{D}, g_s \sim p^{\mathcal{D}}_{\gamma_l}(\cdot|s,a))}\left[\left(Q_l(s, g_s - s, a) - r(s, g_s) - \gamma_l V_l(g_s - s')\right)^2\right], \\
\mathcal{L}_{Q_h} &= \mathbb{E}_{(s,\cdots,g_s)\sim\mathcal{D}, g \sim p^{\mathcal{D}}(\cdot|s,a))}\left[\left(Q_h(s, g, \hat{\phi}_\omega(g_s - s)) - V_h(g_s, g)\right)^2\right].
\end{aligned}$$

where, as before, $\tilde{Q}_l$ and $\tilde{V}_h$ denote the target networks, $\gamma_l = 1 - \frac{1}{n}$ denotes the low-level discount factor, and ℓ^2_τ denotes the expectile loss $\ell^2_\tau(x) = |\tau - (x < 0)|x^2$. The policies are extracted using one of the following two loss functions for the low-level policy,

$$\begin{aligned}
\mathcal{L}^{\text{AWR}}_{\pi_l, \phi_\omega} &= -\mathbb{E}_{(s,a,s'\cdots g_s)\sim\mathcal{D}}\left[e^{\alpha_l(Q_l(s, g_s - s, a)) - V_l(g_s - s))} \log \pi_l\left(a \mid s, \hat{\phi}_\omega(s, g_s)\right)\right], \\
\mathcal{L}^{\text{DDPGBC}}_{\pi_l, \phi_\omega} &= -\mathbb{E}_{(s,a,s',\cdots g_s)\sim\mathcal{D}}\left[Q_l(s, g_s, \mu^{\pi_l}(s, \hat{\phi}_\omega(s, g_s))) + \alpha_l \log \pi_l(a \mid s, \hat{\phi}_\omega(s, g_s))\right],
\end{aligned}$$

and one of the following two loss functions for the high-level policy:

$$\begin{aligned}
\mathcal{L}^{\text{AWR}}_{\pi_h} &= -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g \sim p^{\mathcal{D}}_{\gamma}(\cdot|s)}\left[e^{\alpha_h(Q_h(s, g, \hat{\phi}_\omega(g_s - s))) - V_h(s, g))} \log \pi_h\left(\hat{\phi}_\omega(s, g_s) \mid s, g\right)\right], \\
\mathcal{L}^{\text{DDPGBC}}_{\pi_h} &= -\mathbb{E}_{(s\cdots g_s)\sim\mathcal{D}, g \sim p^{\mathcal{D}}_{\gamma}(\cdot|s)}\left[Q_h(s, g, \mu^{\pi_h}(s, g)) + \alpha_h \log \pi_g(\hat{\phi}_\omega(s, g_s) \mid s, g)\right].
\end{aligned}$$

Algorithm 2 Abstractive RL explicitly enforcing translation invariance (ARLe)

Training
Initialise low-level policy $\pi_l(a \mid s, \omega)$, high-level policy $\pi_h(\omega \mid s, g)$, and representation ϕ_ω.
while not converged **do**
  Sample batch from $\mathcal{D}$
  ▷ **Hierarchical Policy**
    Update low-level policy π_l and representation $\phi_\omega$using $\pi_l(a \mid s, \hat{\phi}_\omega(s, g_s))$ (Equation 4)
    $\omega \leftarrow$ stopgrad$(\hat{\phi}_\omega(s, g_s))$
    Update high-level policy $\pi_h(\omega \mid s, g)$ (Equation 6)
  ▷ **Hierarchical Value**
    Update low-level value function $V_l(g_s - s)$ (Equation 2)
    Update high-level value function $V_h(s, g)$ (Equation 5)
    Update critic $Q_l(s, g_s - s, a)$ (Equation 3)
end while
return $\pi_l, \pi_h, \phi_\omega$

Deployment (state s, goal g)
Sample option from high-level policy $\omega \sim \pi_h(\cdot \mid s, g)$
Soft-Normalise $\omega \leftarrow \frac{\omega \cdot \tanh(\|\omega\|)}{\|\omega\|} \cdot \sqrt{d}$
Sample action from low-level policy $a \sim \pi_l(\cdot \mid s, \omega)$

D Full Results

Table 2: **Full Results 1.** We report each method's average (binary) success rate (%) across the five test-time goals on each task. Bootstrapped 95% CI over 4 seeds and 20 evaluation runs. Blue bold indicates the highest mean; black bold overlapping confidence intervals.

Environment	Task	IQL	HIQL1vr	HIQL2v	HIQL2vr	ARLi	ARLe
pointmaze-giant-navigate-v0	1	0±0	1±2	6±9	2±4	38±38	4±4
	2	0±0	66±22	64±24	75±19	86±7	31±14
	3	0±0	49±21	1±2	24±16	8±8	2±4
	4	0±0	71±33	4±6	60±20	10±10	22±21
	5	0±0	89±7	31±19	69±18	96±4	21±14
	Overall	0±0	**55±11**	21±7	**46±6**	**48±8**	16±2
pointmaze-giant-stitch-v0	1	0±0	0±0	0±0	0±0	0±0	0±0
	2	0±0	0±0	0±0	0±0	0±0	0±0
	3	0±0	0±0	0±0	0±0	0±0	0±0
	4	0±0	0±0	0±0	0±0	0±0	0±0
	5	0±0	0±0	0±0	0±0	0±0	0±0
	Overall	0±0	0±0	0±0	0±0	0±0	0±0
antmaze-giant-navigate-v0	1	0±0	18±8	19±11	29±9	18±5	25±4
	2	1±2	44±19	56±6	50±10	48±8	42±8
	3	0±0	34±7	25±8	39±11	51±18	35±10
	4	0±0	30±6	44±7	54±14	56±9	44±6
	5	0±0	64±6	57±9	71±16	66±6	64±11
	Overall	0±0	38±3	**40±3**	**48±6**	**48±3**	**42±4**
antmaze-giant-stitch-v0	1	0±0	8±8	9±4	30±5	44±14	26±7
	2	0±0	6±4	30±12	26±11	22±5	14±4
	3	0±0	2±2	5±8	4±4	15±6	10±13
	4	0±0	16±16	16±11	36±9	54±14	45±11
	5	0±0	4±4	15±6	6±2	26±21	10±0
	Overall	0±0	7±7	15±3	20±3	**32±7**	21±1
antmaze-teleport-stitch-v0	1	38±8	41±7	55±8	45±9	42±9	45±4
	2	55±4	36±11	61±6	46±7	45±13	55±11
	3	55±9	16±4	31±12	34±4	35±18	29±4
	4	45±9	24±9	22±7	39±11	27±4	36±14
	5	51±6	26±9	46±11	34±11	31±6	41±6
	Overall	**49±2**	29±4	43±3	40±6	36±5	41±4
humanoidmaze-giant-navigate-v0	1	1±2	16±9	8±8	2±4	40±9	29±11
	2	2±2	36±19	29±14	16±14	50±6	46±4
	3	0±0	12±5	11±6	10±9	41±9	32±5
	4	0±0	19±18	11±6	15±12	57±10	40±10
	5	0±0	26±9	1±2	10±13	56±14	45±4
	Overall	1±1	22±11	12±5	11±10	**49±6**	38±4
humanoidmaze-giant-stitch-v0	1	0±0	5±8	0±0	0±0	12±5	11±6
	2	2±2	11±8	2±2	0±0	31±16	21±9
	3	0±0	2±2	0±0	0±0	11±2	8±4
	4	0±0	0±0	0±0	0±0	9±9	9±7
	5	0±0	0±0	0±0	0±0	2±2	4±2
	Overall	0±0	4±3	0±0	0±0	**13±2**	**10±3**

Table 3: **Full Results 2.** We report each method's average (binary) success rate (%) across the five test-time goals on each task. Bootstrapped 95% CI over 4 seeds and 20 evaluation runs. Blue bold indicates the highest mean; black bold overlapping confidence intervals.

Environment	Task	IQL	HIQL1vr	HIQL2v	HIQL2vr	ARLi	ARLe
cube-double-play-v0	1	94±6	8±2	0±0	14±2	96±4	94±4
	2	46±11	0±0	0±0	0±0	59±6	78±11
	3	59±8	0±0	0±0	0±0	52±5	68±4
	4	15±6	0±0	0±0	0±0	14±7	35±6
	5	34±9	0±0	0±0	0±0	44±6	61±9
	Overall	50±3	2±0	0±0	3±0	53±2	**67±3**
cube-triple-play-v0	1	51±7	35±16	0±0	4±4	70±10	69±9
	2	1±2	0±0	0±0	0±0	1±2	2±2
	3	4±2	0±0	0±0	0±0	0±0	2±4
	4	0±0	0±0	0±0	0±0	0±0	0±0
	5	0±0	0±0	0±0	0±0	0±0	0±0
	Overall	11±2	7±3	0±0	1±1	**14±2**	**15±3**
cube-quadruple-play-v0	1	0±0	0±0	0±0	0±0	4±2	2±2
	2	0±0	0±0	0±0	0±0	0±0	0±0
	3	0±0	0±0	0±0	0±0	1±2	0±0
	4	0±0	0±0	0±0	0±0	0±0	0±0
	5	0±0	0±0	0±0	0±0	0±0	0±0
	Overall	0±0	0±0	0±0	0±0	**1±0**	0±0
puzzle-3x3-play-v0	1	100±0	100±0	0±0	100±0	100±0	100±0
	2	100±0	14±21	0±0	15±12	88±12	45±34
	3	99±2	4±6	0±0	0±0	78±19	22±30
	4	100±0	2±4	0±0	2±2	77±28	21±28
	5	100±0	15±12	0±0	1±2	89±11	29±32
	Overall	**100±0**	27±8	0±0	24±3	86±14	44±25
puzzle-4x4-play-v0	1	50±10	66±36	0±0	29±12	91±9	100±0
	2	6±4	32±24	0±0	15±6	70±15	75±18
	3	38±3	59±32	0±0	18±8	81±7	90±9
	4	29±13	48±29	0±0	10±4	72±18	89±6
	5	29±7	41±25	0±0	12±5	72±13	85±5
	Overall	30±3	49±27	0±0	17±6	**78±11**	**88±6**
puzzle-4x5-play-v0	1	72±12	82±11	0±0	62±15	90±13	68±36
	2	1±2	2±4	0±0	0±0	0±0	0±0
	3	0±0	0±0	0±0	0±0	0±0	0±0
	4	0±0	0±0	0±0	0±0	0±0	0±0
	5	0±0	0±0	0±0	0±0	0±0	0±0
	Overall	**15±3**	**17±3**	0±0	**12±3**	**18±3**	**14±7**
puzzle-4x6-play-v0	1	51±9	0±0	0±0	76±6	65±32	45±45
	2	15±6	0±0	0±0	16±9	2±2	0±0
	3	0±0	0±0	0±0	0±0	0±0	0±0
	4	0±0	0±0	0±0	0±0	0±0	0±0
	5	0±0	0±0	0±0	0±0	0±0	0±0
	Overall	13±1	0±0	0±0	**18±2**	**14±7**	**9±9**
scene-play-v0	1	31±6	22±5	26±23	26±7	50±6	44±13
	2	16±11	5±4	11±4	6±4	16±11	18±3
	3	6±8	8±8	5±8	10±4	19±4	12±2
	4	9±6	16±6	11±4	19±9	30±8	26±9
	5	1±2	9±6	8±5	4±4	8±5	6±4
	Overall	13±2	12±3	12±7	13±1	**24±3**	**21±3**

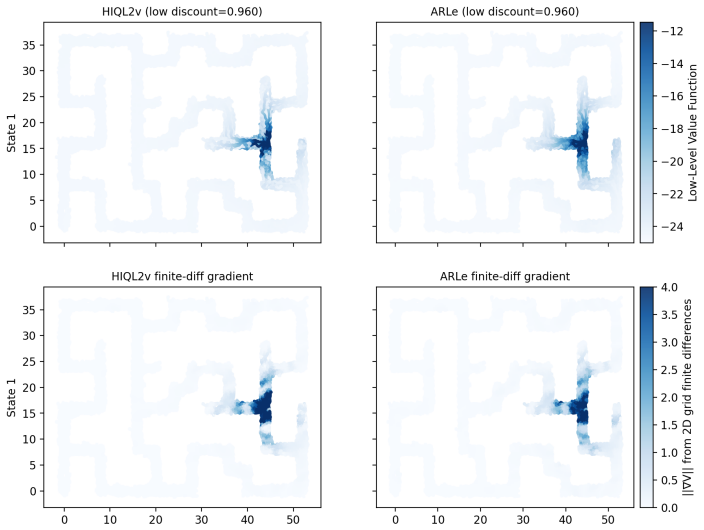

Figure 4: **Low-level** value functions (**top**) and **gradient** of low-level value function (**bottom**). IQL and HIQL1vr are excluded as they have a single value function.

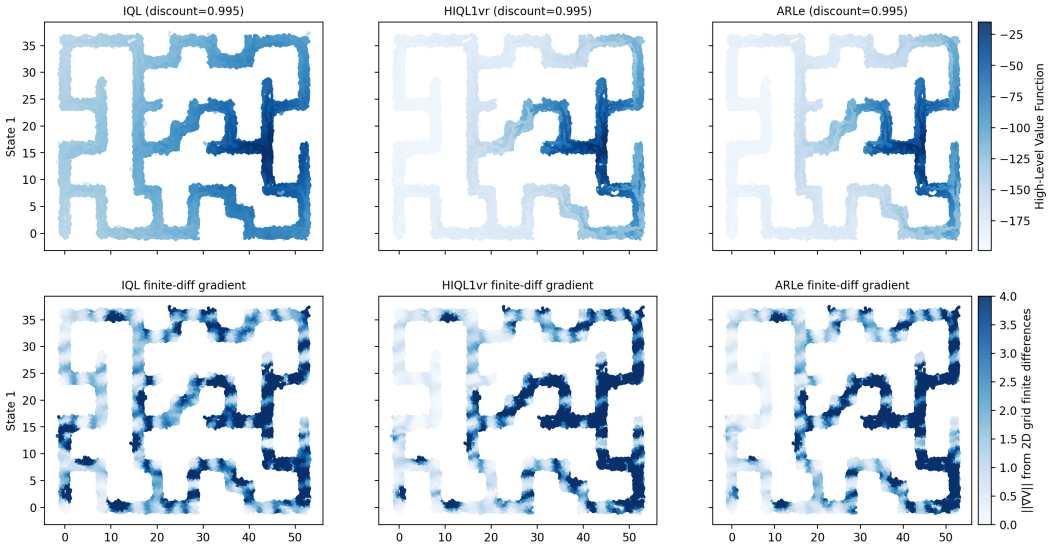

Figure 5: **High-level** value functions (**top**) and **gradient** of high-level value function (**bottom**). HIQL2v and HIQL2vr are excluded, as they have identical ones to ARL.

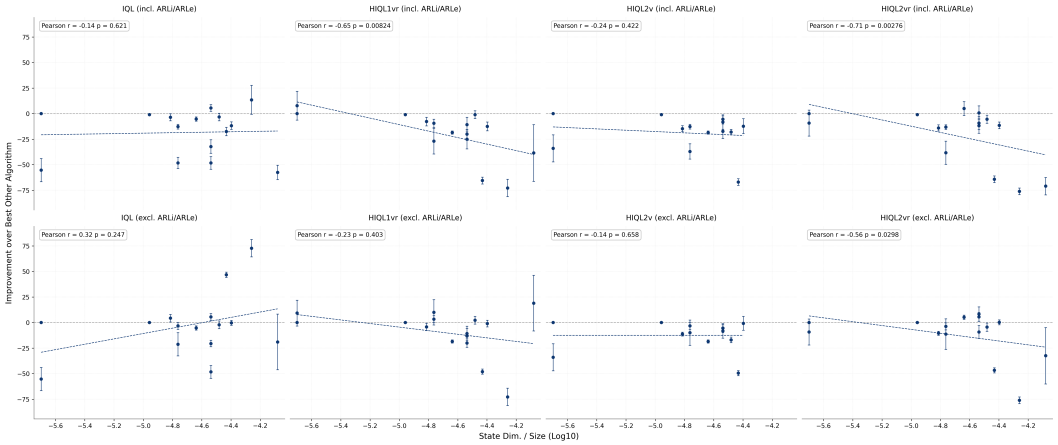

Figure 6: **Performance improvements** over next best performing algorithm against number of state dimensions per dataset sample: **IQL** (**left**), **HIQL1vr** (**centre left**), **HIQL2v** (**centre right**) and **HIQL2vr** (**right**). Including ARLi and ARLe (**top**), and excluding ARLi and ARLe (**bottom**). Bootstrapped 95% CI over 4 seeds and 20 evaluation runs.

E Experimental Details

We release all code and hyperparameters in the following repository: `https://github.com/CWibault/arl`.

Datasets. We use the standard OGBench datasets. States are randomly and uniformly sampled from the dataset. Goals for the value function learning and policy extraction are sampled using a certain probability of sampling the current state $p^{\mathcal{D}}_{\text{cur}}$, from the current trajectory $p^{\mathcal{D}}_{\text{traj}}$ (geometrically, according to the discount factor, or uniformally), or randomly from the dataset $p^{\mathcal{D}}_{\text{rand}}$. waypoints g_s are taken as the states n steps ahead of the current state.

Table 4: **Environment Characteristics.** Environment properties to provide intuition for interpreting results.

Task	State Dim.	Action Dim.	Max. Episode Length	Dataset Size
pointmaze-giant	2	2	1000	1M
antmaze-giant	29	8	1000	1M
antmaze-teleport	29	8	1000	1M
humanoidmaze-large	69	21	1000	4M
humanoidmaze-giant	69	21	4000	4M
cube-double	37	5	500	1M
cube-triple	46	5	1000	3M
cube-quadruple	55	5	1000	5M
puzzle-3x3	55	5	500	1M
puzzle-4x4	83	5	500	1M
puzzle-4x5	99	5	1000	3M
puzzle-4x6	115	5	1000	5M
scene-play	40	5	750	1M

Reward Relabelling. Unlike prior work on OGBench [5], we use the original environment reward functions for relabeling rewards rather than using a binary indicator of the state-index to ensure that the agent remains focused on the primary task objectives and prevents the value function from becoming overly specific to irrelevant dimensions. To ensure fair comparison, this relabeling strategy is applied consistently across all algorithms and tasks. Note that assuming access to the environment reward function in robotic tasks is an entirely valid assumption.

Hyperparameters. We provide the full list of hyperparameters. We follow those from Park et al. [5] and Park et al. [12]. Notably, while these parameters were specifically tuned for HIQL, we apply them to ARL without further adjustment. The fact that ARL achieves strong performance using parameters optimised for a different algorithm demonstrates its robustness. We use DDPGBC with a behaviour cloning strength of 0.1 to extract the high-level policy in manipulation tasks, which allows for more extrapolation [11]. This was generally found to outperform AWR.

Table 5: Hyperparameters

Hyperparameter	**Value**
Gradient steps	10^6
Optimiser	Adam [57]
Learning rate	0.0003
Batch size	1024
Layer Normalisation [58]	True
Nonlinearity	GELU [59]
Value MLP	[1024, 1024, 1024, 1024]
Actor MLP	[1024, 1024, 1024, 1024]
Representation MLP	[512, 512, 512]
Representation Dimension	10
Target network update rate	0.005
IQL Expectile τ	0.9 (IQL), 0.7 (HIQL1vr, HIQL2v, HIQL2vr, ARLi, ARLe)
Value ratio $(p^{\mathcal{D}}_{\text{cur}}, p^{\mathcal{D}}_{\text{traj}}, p^{\mathcal{D}}_{\text{rand}}, p^{\mathcal{D}}_{\text{geom}})$	$(0.2, 0.5, 0.3, 0)$
Low Value ratio $(p^{\mathcal{D}}_{\text{cur}}, p^{\mathcal{D}}_{\text{traj}}, p^{\mathcal{D}}_{\text{rand}}, p^{\mathcal{D}}_{\text{geom}})$	$(0.10, 0.85, 0.05, 1)$
High Value ratio $(p^{\mathcal{D}}_{\text{cur}}, p^{\mathcal{D}}_{\text{traj}}, p^{\mathcal{D}}_{\text{rand}}, p^{\mathcal{D}}_{\text{geom}})$	$(0.2, 0.5, 0.3, 0)$
Policy ratio $(p^{\mathcal{D}}_{\text{cur}}, p^{\mathcal{D}}_{\text{traj}}, p^{\mathcal{D}}_{\text{rand}}, p^{\mathcal{D}}_{\text{geom}})$	$(0.0, 0.5, 0.5, 1)$

Table 6: Task-Specific Hyperparameters

Task	n	γ	**Loss** π	α	**Loss** π_l	α_l	**Loss** π_h	α_h
pointmaze-giant-navigate-v0	25	0.995	DDPGBC	0.1	AWR	3.0	AWR	3.0
pointmaze-giant-stitch-v0	25	0.995	DDPGBC	0.1	AWR	3.0	AWR	3.0
antmaze-giant-navigate-v0	25	0.995	DDPGBC	0.1	AWR	3.0	AWR	3.0
antmaze-giant-stitch-v0	25	0.995	DDPGBC	0.1	AWR	3.0	AWR	3.0
antmaze-teleport-stitch-v0	25	0.990	DDPGBC	0.1	AWR	3.0	AWR	3.0
humanoidmaze-giant-navigate-v0	100	0.999	DDPGBC	0.1	AWR	3.0	AWR	3.0
humanoidmaze-giant-stitch-v0	100	0.999	DDPGBC	0.1	AWR	3.0	AWR	3.0
cube-double-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1
cube-triple-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1
cube-quadruple-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1
puzzle-3x3-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1
puzzle-4x4-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1
puzzle-4x5-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1
puzzle-4x6-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1
scene-play-v0	25	0.99	DDPGBC	1.0	AWR	3.0	DDPGBC	0.1

The low-level discount factor is computed as $\gamma_l = 1 - \frac{1}{n}$. The high-level discount factor is computed as $\gamma_h = \gamma^n$.

F Limitations

ARLe's performance depends on the alignment between its inductive bias (assuming translational invariance) and the environment's structure. We discuss these limitations with the experiments (Section 5) and in the conclusion (Section 6).

Like other prior methods learning action-free value functions, learning an action-free high-level value function biases our instantiations of ARL towards being optimistic in stochastic environments [24]. Such optimism bias could be addressed by disentangling controllable parts of the state [60], but we leave this to future work. We also note that, since only the high-level value function is action-free, performance degradation compared to IQL for both ARLe and ARLi in the stochastic environment (*antmaze-teleport-stitch-v0*) is less significant than for HIQL1vr, for example.

Including more than 15 environments would have helped to strengthen our hypothesis in Section 6, but we were limited by compute resources.

G Compute

All experiments were conducted on NVIDIA L40 GPUs, lasting 3 hours per run, including evaluation.