# PLANNING WITH VALUE-AWARE LEARNED GOALS FOR HIERARCHICAL REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Hierarchical reinforcement learning (HRL) decomposes long-horizon decision making across temporal scales, with a high-level policy periodically specifying goals and a low-level controller realizing them through primitive actions. Departing from the conventional use of learned goal-conditioned policies for low-level control, we introduce Value-Aware Latent-Space Planning (VALP), a novel low-level policy-free framework that combines reinforcement learning for strategic decision making with world model-based planning for closed-loop control. VALP learns a compact, value-aware latent space that preserves task-relevant information, in which a high-level policy specifies temporally extended abstract goals, while a model-predictive planner directly synthesizes primitive actions by planning through a learned latent world model toward these objectives. By coupling value-based representation learning and model-based planning, VALP separates learning task-relevant objectives for long-horizon decision making from their realization through short-horizon control. VALP significantly outperforms state-of-the-art HRL methods on long-horizon control tasks. Our findings highlight the importance of a task-relevant communication interface for hierarchical control and demonstrate how reinforcement learning for long-horizon decision making can be effectively combined with world model-based control for goal realization.

## 1 INTRODUCTION

Reinforcement learning (RL) has emerged as a dominant framework for sequential decision-making (Sutton & Barto, 2018; Silver et al., 2016). However, long-horizon control remains a persistent challenge: consequential decisions may need to be taken long before their effects become apparent, while successful behavior still requires precise control at the primitive-action level (Pateria et al., 2021). A natural response is hierarchical reinforcement learning (HRL), which decomposes decision making across temporal scales. Typically, a high-level policy specifies temporally extended objectives, while a lower-level controller realizes them (Sutton et al., 1999; Nachum et al., 2018). Yet this decomposition creates a coupled learning problem, as the policies at both levels evolve and must continually adapt to one another (Nachum et al., 2018; Levy et al., 2017). Moreover, temporal abstraction alone does not determine what information should be communicated across levels, leaving open the question of what language the hierarchy should use to express intermediate objectives.

The prevailing approach in HRL is policy-centric: a high-level policy specifies intermediate objectives, and a goal-conditioned low-level policy learns to realize them (Levy et al., 2017; Nachum et al., 2018). These objectives were traditionally expressed in predefined state-based or hand-designed goal spaces (Levy et al., 2017; Nachum et al., 2018), making the communication interface a hand-engineered bottleneck. Subsequent work therefore learns goal representations from transition structure, using criteria such as reachability (Nachum et al., 2019; Zhang et al., 2020; Li et al., 2021b). Such criteria, however, need not preserve distinctions relevant to task progression. Value-based representation learning offers a task-aware alternative and has shown promise for hierarchical offline RL (Park et al., 2023). However, during online training, the critic shaping the representation and the policy determining its training distribution evolve together, while changes in the goal space themselves induce nonstationarity in learning (Li et al., 2021a). Thus, learning task-relevant goals online creates a fundamental tension, as the representation should evolve with policy performance while its changing semantics reshape the coordination problem between hierarchical levels.



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

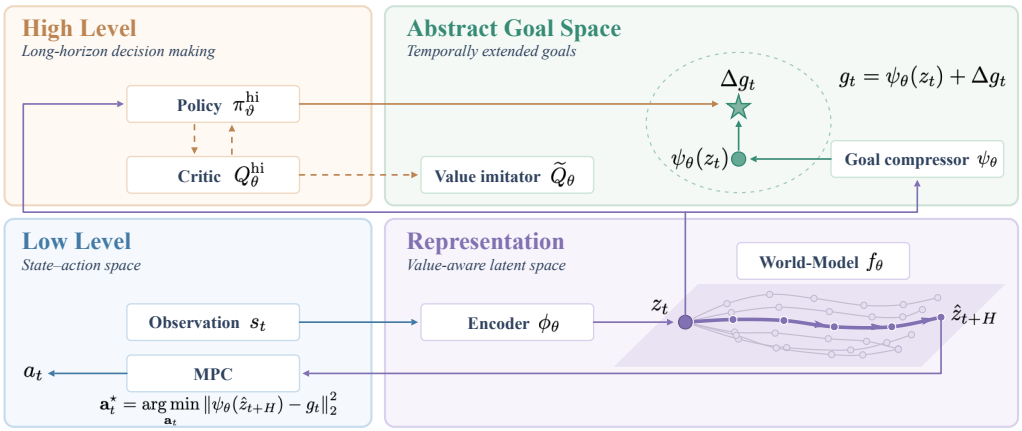



Figure 1: **Framework Figure.** The high-level policy selects temporally extended objectives in a value-aware goal space, whose representation is shaped by distillation from the high-level critic. A learned latent dynamics model predicts candidate trajectories, and the low-level model-predictive controller repeatedly optimizes primitive actions to track the selected goal, using receding-horizon control.

The evolving interface therefore motivates separating goal learning from goal realization. Instead of amortizing realization into a goal-conditioned policy, world-model-based control can optimize actions online toward the objective currently specified by the hierarchy. Recent advances in latent world models have made such planning increasingly effective for continuous control (Hansen et al., 2024; Wang et al., 2024a). By replanning through learned dynamics at each interaction, model-based control can respond to the current objective without first retraining a separate goal-conditioned controller (Assran et al., 2025; Ebert et al., 2018). However, planning remains constrained by its finite horizon and by errors in the learned dynamics. Prediction errors can accumulate along imagined trajectories (Lambert et al., 2022; Asadi et al., 2019), making increasingly long primitive-action rollouts difficult to optimize reliably. Reasoning beyond the planning horizon therefore typically requires a meaningful objective that measures task progress that cannot be generally captured by geometric proximity (Sikchi et al., 2022; Hansen et al., 2024). Thus, world-model-based control alleviates some limitations of policy-centric hierarchies while introducing others of its own.

These trade-offs motivate viewing hierarchy as a mechanism for extending world-model predictive control beyond its local planning horizon. We introduce Value-Aware Latent-Space Planning (VALP), which augments short-horizon latent planning with a slower high-level policy that periodically specifies intermediate objectives. To make these objectives useful for long-horizon decision making, VALP learns a compact value-aware goal representation shaped by high-level return, rather than relying on predefined geometric distances or transition structure alone. As the high-level policy and goal representation evolve during training, the objectives presented to the low level change as well. Rather than requiring a goal-conditioned policy to continually adapt to this evolving interface, VALP learns latent dynamics and re-optimizes primitive actions under the current goal definition at decision time. In this way, value learning determines what should be communicated across temporal scales, while model-based replanning determines how the current objective is realized.

**Contributions.** We introduce VALP, an online hierarchical control framework that uses reinforcement learning to select long-horizon objectives and model-predictive control with a world model to realize them through short-horizon closed-loop planning. Central to VALP is a value-aware goal-learning procedure that distills high-level critic information into a compact goal representation, providing a learned communication interface between high-level decision making and low-level planning. Rather than relying on a predefined goal space, this interface is learned online to capture distinctions relevant to long-horizon return. We demonstrate that VALP achieves higher performance and sample efficiency than representative flat and hierarchical baselines across long-horizon continuous-control benchmarks spanning dense and sparse rewards, visual observations, and increased task horizon. Further experiments examine whether the learned interface remains useful when the long-horizon task changes and characterize sensitivity to planning horizon and subgoal dimensionality. Finally, we demonstrate how model-based planning supports more stable hierarchical learning by adapting action selection through online replanning as the goal semantics evolve.

## 2 PRELIMINARIES

**Problem Setup.** We consider an infinite-horizon Markov decision process (MDP) (Puterman, 1994), specified by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mu_0)$. Here, $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ denote continuous state and action spaces, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, $\gamma \in [0, 1)$ the discount factor, and $\mu_0$ the initial-state distribution. At timestep $t$, the agent observes $s_t$, executes $a_t$, receives reward $r_t = r(s_t, a_t)$, and transitions to $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$. Using data collected through *online* interaction, the objective is to learn a policy that maximizes the expected discounted return $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, without access to the dynamics $\mathcal{P}$ or reward function $r$.

**Model predictive control (MPC).** Given a dynamics model $f$, MPC approximates the infinite-horizon objective by solving a finite-horizon trajectory optimization problem (Kouvaritakis et al., 2016). At state $s_t$, MPC optimizes an action sequence $\mathbf{a}_t = (a_t, \ldots, a_{t+H-1})$ over a horizon $H$:

$$\mathbf{a}_t^\star = \arg\max_{\mathbf{a}_t} \mathbb{E}\left[\sum_{h=0}^{H-1} \gamma^h r(\hat{s}_{t+h}, a_{t+h}) + \gamma^H V(\hat{s}_{t+H})\right],$$

where predicted states evolve as $\hat{s}_{t+h+1} = f(\hat{s}_{t+h}, a_{t+h})$ from $\hat{s}_t = s_t$, and the terminal term $V$ estimates return beyond the planning horizon. Only the first action $a_t^\star$ is executed, and the optimization problem is re-solved from each newly observed state, providing closed-loop feedback during execution. Because the maximization is generally intractable in high-dimensional action spaces, it is approximated by sampling-based optimizers (Williams et al., 2017; Pinneri et al., 2021).

In practice, the true environment dynamics are unknown and are typically approximated by a learned dynamics model, often in a learned latent space (Hansen et al., 2024). Such world model-based control enables MPC directly from experience, but remains constrained by its finite planning horizon (Hafner et al., 2019; Zhang et al., 2026). Extending this horizon increases computational cost and compounds model error, while short horizons require objectives that provide useful guidance toward long-horizon task completion. We address this limitation by learning a policy providing intermediate objectives, effectively guiding short-horizon planning toward long-horizon goals.

## 3 VALUE-AWARE LATENT-SPACE PLANNING

For long-horizon control, VALP combines model-based planning, which provides precise closed-loop control but is limited in how far ahead it can optimize, with reinforcement learning, which optimizes long-term return but faces credit assignment that degrades as primitive-action sequences lengthen. As illustrated in Figure 1, VALP connects these two timescales through a learned latent interface between high-level goal selection and low-level model-based planning. Both levels operate on a learned latent state rather than raw observations. An encoder $\phi_\theta : \mathcal{S} \to \mathcal{Z}$ maps an observation to $z_t = \phi_\theta(s_t)$, and a latent dynamics model $f_\theta : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ predicts its evolution under primitive actions, $\hat{z}_{t+1} = f_\theta(z_t, a_t)$. We learn this latent model without reconstructing observations, following prior work on decoder-free latent world models (Hansen et al., 2024; Assran et al., 2025), and $\mathcal{Z}$ serves both for model-based rollouts and as the state representation for high-level decision making.

The two timescales are connected through a goal compressor $\psi_\theta : \mathcal{Z} \to \mathcal{G}$, which maps latent states into a $d_g$-dimensional abstract goal space with $d_g \ll \dim \mathcal{Z}$. At the coarser timescale, a high-level policy $\pi_\vartheta^{\text{hi}}$ selects a new objective $g \in \mathcal{G}$ every $H$ environment steps. Rather than predicting an objective outright, it emits a displacement $\Delta g_\tau \sim \pi_\vartheta^{\text{hi}}(\cdot \mid z_\tau)$ relative to the agent's current position, giving $g_\tau = \psi_\theta(z_\tau) + \Delta g_\tau$, which is held fixed for the following $H$ steps. A high-level critic $Q_\theta^{\text{hi}}(z, \Delta g)$ evaluates these choices from the uncompressed latent state using the return accumulated over each segment, so it learns which temporally extended objectives advance the task without representing how they are achieved. A second critic $\widetilde{Q}_\theta(\psi_\theta(z_\tau), \Delta g)$ evaluates the same choices from the compressed representation $\psi_\theta(z_\tau)$ and is trained to match $Q_\theta^{\text{hi}}$, so $\mathcal{G}$ retains the information the high level uses to distinguish objectives. At the finer timescale, primitive actions are determined by a model-predictive planner that uses sampling-based optimization over latent dynamics.

### 3.1 FRAMEWORK COMPONENTS

**Learned components.** We first introduce the learned components underlying VALP. These comprise a latent world model for predicting action-conditioned transitions, a compact goal representation defining the interface between temporal scales, and a high-level actor–critic for selecting and evaluating intermediate objectives. We summarize their parameterization below:

| | | |
|---|---|---|
| **Encoder** | $z_t = \phi_\theta(s_t)$ | ▷ maps observations to latent states |
| **Latent dynamics** | $\hat{z}_{t+1} = f_\theta(z_t, a_t)$ | ▷ predicts latent transitions |
| **Goal compressor** | $g_t = \psi_\theta(z_t)$ | ▷ maps latent states to the compact goal space |
| **High-level policy** | $\Delta g_t \sim \pi^{\text{hi}}_\vartheta(\cdot \mid z_t)$ | ▷ selects temporally extended objectives |
| **High-level critic** | $Q^{\text{hi}}_\theta(z_t, \Delta g_t)$ | ▷ evaluates objectives from the latent state |
| **Imitator critic** | $\widetilde{Q}_\theta(\psi_\theta(z_t), \Delta g_t)$ | ▷ matches high-level values from the goal space |

**Model learning.** The encoder $\phi_\theta$, latent dynamics $f_\theta$, high-level critic $Q^{\text{hi}}_\theta$, goal encoder $\psi_\theta$, and auxiliary critic $\widetilde{Q}_\theta$ are jointly trained from replay data $\mathcal{D}$. We optimize:

$$\mathcal{L}_{\text{model}}(\theta) = \mathcal{L}_{\text{dyn}} + \lambda_Q \mathcal{L}_Q + \lambda_{\text{dist}} \mathcal{L}_{\text{dist}}, \tag{1}$$

where $\mathcal{L}_{\text{dyn}}$ learns action-conditioned latent dynamics, $\mathcal{L}_Q$ learns long-horizon task value, and $\mathcal{L}_{\text{dist}}$ distills this value information into the compact goal representation. We train the latent dynamics to predict the next latent state and the high-level critic using temporal-difference learning (Sutton & Barto, 2018) with a distributional value objective; we defer these standard objectives to Appendix C. Our goal representation is learned through value distillation. Specifically, the auxiliary critic $\widetilde{Q}_\theta$ predicts the high-level critic from the compressed representation,

$$\mathcal{L}_{\text{dist}} = \mathbb{E}_{(s,\Delta g)\sim\mathcal{D}} \left[ \text{CE}\left( \widetilde{Q}_\theta(\psi_\theta(z), \Delta g), \text{sg}\big[Q^{\text{hi}}_\theta(z, \Delta g)\big] \right) \right], \qquad z = \phi_\theta(s), \tag{2}$$

where sg denotes stop-gradient and CE is the distributional cross-entropy loss. By requiring the compressed representation to retain information sufficient to reproduce the high-level critic's predictions on replayed state–action pairs, this objective provides long-horizon value supervision to the goal space used for hierarchical communication and planning. The goal dimensionality thus induces a trade-off between value preservation and abstraction. A lower-dimensional representation imposes a stronger information bottleneck, potentially increasing critic imitation error, while providing the high-level policy with a more compact action space. Increasing the dimensionality relaxes this bottleneck but enlarges the space over which intermediate objectives must be selected.

**Policy learning.** The high-level policy can in principle be instantiated with any actor–critic algorithm (Konda & Tsitsiklis, 1999). We use Soft Actor-Critic (SAC) (Haarnoja et al., 2018), an off-policy maximum-entropy actor–critic method. The high-level actor $\pi^{\text{hi}}_\vartheta$ observes the latent state and outputs a relative goal, and is optimized separately from the model objective against:

$$\mathcal{L}_\pi(\vartheta) = \mathbb{E}_{s\sim\mathcal{D},\, \Delta g\sim\pi^{\text{hi}}_\vartheta(\cdot|z)} \left[ \alpha \log \pi^{\text{hi}}_\vartheta(\Delta g \mid z) - \text{sg}\big[Q^{\text{hi}}_\theta(z, \Delta g)\big] \right], \quad z = \text{sg}[\phi_\theta(s)], \tag{3}$$

where the temperature $\alpha$ is adapted toward a target entropy (Haarnoja et al., 2018). The stop-gradients isolate the actor update from representation and critic learning, such that $\mathcal{L}_\pi$ updates only the policy parameters. The high-level policy acts every $H$ environment steps, so its replay transitions span entire segments. Consequently, stored displacements become off-policy as both the high-level policy that issues them and the goal space in which they are expressed evolve. We account for both sources of mismatch by relabeling sampled transitions before learning, as detailed in App. F.

### 3.2 World-Model Planning toward Latent Goals

With intermediate objectives defined by the high-level policy, we next turn to their realization through primitive-action planning. Given a latent goal $g_t$ at state $s_t$, VALP optimizes a sequence of primitive actions through the learned latent dynamics to minimize its predicted terminal distance to $g_t$ in goal space. We formulate this as the finite-horizon model-predictive control problem

$$\mathbf{a}^\star_t = \arg\min_{\mathbf{a}_t} \|\psi_\theta(\hat{z}_{t+h_t}) - g_t\|_2^2, \qquad \hat{z}_{j+1} = f_\theta(\hat{z}_j, a_j), \qquad \hat{z}_t = \phi_\theta(s_t), \tag{4}$$

where $\mathbf{a_t} = (a_t, \ldots, a_{t+h_t-1})$ and $h_t$ denotes the remaining horizon of the current high-level segment. Starting from $H$, the planning horizon decreases as the segment progresses so that each rollout terminates at the segment boundary. Only the first action $a^\star_t$ is executed; after observing the resulting state, VALP replans toward the same goal over the shortened horizon. This receding-horizon procedure allows primitive control to adapt to observed transitions while keeping the goal and its intended endpoint fixed throughout the segment. We solve eq. 4 using the sampling-based iCEM optimizer (Pinneri et al., 2021); implementation details are provided in Appendix J.

**Algorithm 1** Online learning with VALP

**Require:** Segment length $H$; model and policy parameters $\theta, \vartheta$; replay buffers $\mathcal{D}_{\text{lo}}, \mathcal{D}_{\text{hi}}$

**for** each segment starting at $\tau = 0, H, 2H, \ldots$ **do**
  Sample $\Delta g_\tau \sim \pi^{\text{hi}}_\vartheta(\cdot \mid \phi(s_\tau))$ and set $g_\tau \leftarrow \psi(\phi(s_\tau)) + \Delta g_\tau$ *goal selection*
  **for** $t = \tau, \ldots, \tau + H - 1$ **do** *environment interaction*
    Plan toward $g_\tau$ using sampling-based optimization (cf. App. J) over eq. 4
    Execute $a_t$; observe $(r_t, s_{t+1})$; add $(s_t, a_t, s_{t+1})$ to $\mathcal{D}_{\text{lo}}$
  **end for**
  Add $(s_\tau, \Delta g_\tau, R_\tau, s_{\tau+H})$ to $\mathcal{D}_{\text{hi}}$, where $R_\tau = \sum_{t=\tau}^{\tau+H-1} r_t$
  Sample from $\mathcal{D}_{\text{lo}}, \mathcal{D}_{\text{hi}}$ and relabel high-level transitions (App. F) *parameter update*
  Update model parameters $\theta$ using eq. 1
  Update high-level policy $\vartheta$ using eq. 3
**end for**

**Value-aware goal geometry.** The planning objective in eq. 4 measures terminal error in our goal space. We characterize how proximity in this space relates to the value information it preserves.

**Proposition 3.1** (Value Preservation)**.** *Suppose the compressed critic approximates the high-level critic uniformly with error at most $\epsilon$, such that*

$$\sup_{z, \Delta g} \left| Q^{\text{hi}}_\theta(z, \Delta g) - \widetilde{Q}_\theta(\psi_\theta(z), \Delta g) \right| \leq \epsilon. \tag{5}$$

*Further suppose that $\widetilde{Q}_\theta(\cdot, \Delta g)$ is $L$-Lipschitz in its first argument uniformly over $\Delta g$. Then, for any latent states $z_1, z_2$,*

$$\sup_{\Delta g} \left| Q^{\text{hi}}_\theta(z_1, \Delta g) - Q^{\text{hi}}_\theta(z_2, \Delta g) \right| \leq 2\epsilon + L \left\| \psi_\theta(z_1) - \psi_\theta(z_2) \right\|_2 . \tag{6}$$

The proof is provided in App. H. Prop. 3.1 shows that under bounded distillation error and the stated regularity assumption, nearby representations cannot induce arbitrarily different high-level value predictions for the same goal displacement. In particular, states mapped to the same representation differ in their high-level critic predictions by at most $2\epsilon$ uniformly over $\Delta g$. This result concerns preservation of value information rather than controllability. It does not imply that nearby representations are mutually reachable within a high-level segment, nor that Euclidean distance corresponds to control effort or progress. VALP instead uses the learned latent dynamics and receding-horizon planning to realize selected targets, while value distillation determines which distinctions are retained by the hierarchical interface.

### 3.3 Online Learning

VALP jointly learns its higher-level decision-making and model-based control components from online interaction with the environment, as summarized in Alg. 1. During interaction, action selection follows the hierarchical procedure described in Sec. 3.2. At the beginning of each segment, the high-level policy selects a goal displacement $\Delta g_\tau$ and corresponding latent goal $g_\tau$, which is held fixed for the next $H$ steps. At each primitive step within the segment, the planner optimizes an action sequence toward $g_\tau$ according to Eq. 4, executes the first action, and replans from the resulting observation. The resulting experience is stored at two temporal scales. Each primitive transition $(s_t, a_t, s_{t+1})$ is added to $\mathcal{D}_{\text{lo}}$ for model learning, while at the end of each segment, the high-level transition $(s_\tau, \Delta g_\tau, R_\tau, s_{\tau+H})$ is added to $\mathcal{D}_{\text{hi}}$ for high-level value and goal encoder learning.

Learning from high-level replay requires accounting for changes in the learned goal space during training. As the representation evolves, a stored displacement $\Delta g_\tau$ may no longer describe the observed segment transition under the updated goal representation. We therefore relabel sampled high-level transitions before applying the learning updates, replacing the stored displacement with one that better matches the observed endpoint while keeping the segment return and transition fixed (App. F). Using the resulting replay batches, we update the model parameters $\theta$ according to equation 1, jointly learning the latent dynamics, high-level value function, and value-aware goal representation. The high-level policy parameters $\vartheta$ are subsequently updated using equation 3. As these components evolve, the planner realizes the resulting goals using the current model and goal representation, avoiding the need to retrain a separate low-level policy.

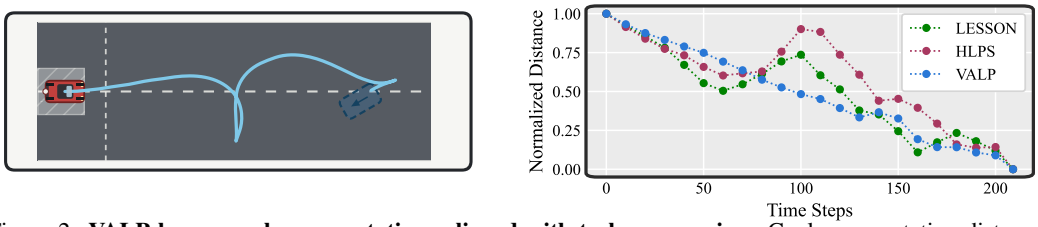



Figure 2: **VALP learns goal representations aligned with task progression.** Goal-representation distance to the final parking configuration along the same successful trajectory. VALP tracks progress throughout the maneuver, while LESSON and HLPS become less aligned during intermediate and final phases.

## 4 Experiments

We organize our empirical evaluation around four questions: **(Q1)** What structure do value-aware goal representations learn, and are these representations effective for control? **(Q2)** Does combining value-aware temporal abstraction with world-model-based control improve long-horizon decision making? **(Q3)** Can VALP learn rich goal representations that support adaptation to changes in the task? **(Q4)** How do the key design choices and hyperparameters of VALP affect its performance?

To answer these questions, we evaluate VALP against representative baselines on continuous-control tasks spanning dense and sparse rewards, visual observations, and increased task scale. Section 4.1 studies the structure and control utility of the learned goal representations **(Q1)**, while Section 4.2 presents the main benchmark comparison **(Q2)**. Section 4.3 evaluates the representations under changes to the long-horizon task **(Q3)**. Finally, Sec. 4.4 ablates key design choices in VALP, including the goal-learning objective and world-model-based subgoal tracking, and analyzes sensitivity to planning horizon and subgoal dimensionality **(Q4)**.

**Baselines.** We compare VALP against baselines spanning three families: *flat RL*, including SAC (Haarnoja et al., 2018), an off-policy actor–critic operating at the primitive-action timescale, and TD-MPC2 (Hansen et al., 2024), a model-based RL method combining learned latent dynamics, sampling-based planning, and a learned policy; *predefined-goal HRL*, represented by HIRO (Nachum et al., 2018), which operates over a goal space derived from state coordinates; and *learned-goal HRL*, including HLPS (Wang et al., 2024b) and LESSON (Li et al., 2021b), which learn representations for intermediate goals using transition structure of the environment. All methods use the same evaluation protocol; implementation details are provided in Appendix I.

### 4.1 Goal Learning Comparison

*Setup:* We first examine whether the learned goal representation preserves information relevant to long-horizon task progress. We consider a controlled parking task in which a car must execute a temporally extended maneuver to reach a designated parking configuration in an open environment (see Appendix D.1). We train HLPS, LESSON, and VALP on this task and compare their learned representations along the same successful trajectory. After training, we execute a complex parking maneuver and track the distance to the final parking configuration in each method's learned goal space throughout the trajectory. Training curves are provided for completeness in Appendix E.1.

*Results.* Figure 2 visualizes the representations induced by each method along the trajectory. VALP exhibits a smooth progression aligned with task structure, changing consistently as the car approaches the final parking pose. In contrast, LESSON and HLPS become less aligned with task progress during intermediate and final maneuvering phases. This reflects their different objectives: LESSON and HLPS organize their goal spaces primarily through transition structure, whereas VALP incorporates value information to preserve task-relevant distinctions. These results address **(Q1)**, showing that value-aware goal learning can produce representations aligned with task progress.

### 4.2 Benchmarks

*Setup.* We evaluate long-horizon continuous-control tasks in MuJoCo (Todorov et al., 2012), considering four environments commonly used in hierarchical RL: `AntMaze`, `AntPush`, `AntFall`, and `AntFourRooms`. For each environment, we consider four settings: dense reward, sparse reward, image-based observations, and large-scale variants with dense reward. Together, these settings vary reward structure, observation modality, and task scale. We train all methods with randomly sampled task goals and evaluate them both on randomly sampled goals and a fixed challenging goal for each environment. As an additional baseline, we evaluate VALP-XY, which replaces the learned goal rep-

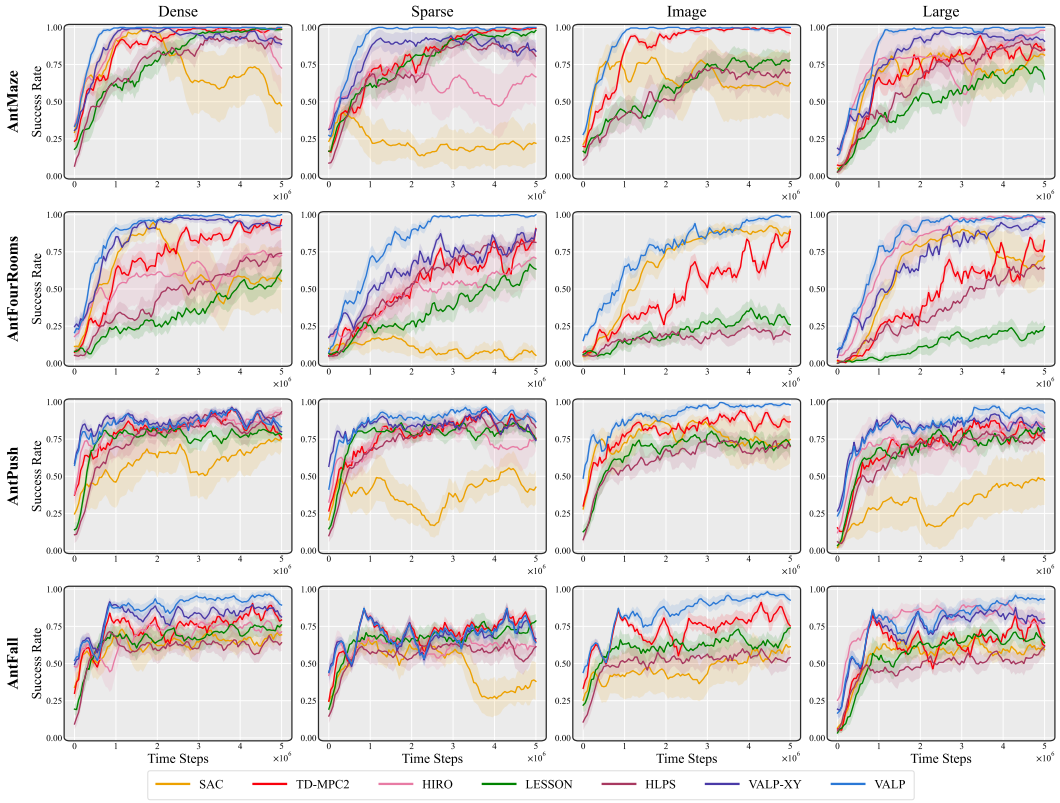



Figure 3: **VALP demonstrates increased sample efficiency.** Training curves report success rate on randomly sampled goals across environments (rows) and settings (columns), varying reward structure, observation modality, and task scale. Curves show the mean over five runs, with shaded area denoting standard error.

Table 1: **VALP outperforms baselines on challenging fixed goals.** Success rate (%) at final checkpoint, averaged across settings within each environment. Subscripts denote variation across settings; HIRO and VALP-XY exclude image-based settings as their goal spaces use position coordinates. See App. E for extended results.

| Environment | SAC | TD-MPC2 | HIRO | LESSON | HLPS | VALP-XY | VALP |
|---|---|---|---|---|---|---|---|
| AntMaze | 39.7±13.8 | 74.6±23.5 | 67.1±7.6 | 55.7±19.3 | 73.1±7.5 | 77.3±12.3 | **99.9**±0.1 |
| AntFourRooms | 57.2±20.6 | 17.9±17.1 | 76.9±11.7 | 0.9±0.9 | 26.9±13.2 | 68.1±19.5 | **91.0**±6.2 |
| AntPush | 7.0±6.6 | 9.0±9.0 | 15.7±11.1 | 19.2±2.1 | 35.2±12.9 | 25.2±10.0 | **60.6**±15.0 |
| AntFall | 0.0±0.0 | 22.5±7.7 | 23.1±18.5 | 9.0±4.4 | 7.2±4.1 | 42.4±14.6 | **61.5**±15.8 |
| Average | 26.0±8.3 | 31.0±9.7 | 45.7±9.7 | 21.2±7.0 | 35.6±7.7 | 53.3±8.8 | **78.2**±6.8 |

resentation with the Ant's $XY$ position predictions, providing a hand-designed goal space against which to assess the contribution of our goal learning scheme. Environment details are provided in App. D.2. We report results over five independent runs for each method and environment.

*Results.* Fig. 3 and Tab. 1 show that VALP achieves higher success rates and reaches strong performance earlier across the benchmark. The flat baselines SAC and TD-MPC2 perform competitively on simpler navigation tasks but degrade substantially on `AntPush` and `AntFall`, suggesting a benefit from temporal abstraction as tasks require longer sequences of coordinated decisions. Among hierarchical methods, HIRO remains competitive but relies on a predefined, position-based goal space. In contrast, HLPS and LESSON learn their goal spaces but generally underperform HIRO, highlighting the difficulty of learning an effective communication interface. VALP combines learned goals with long-horizon value information and exhibits greater sample efficiency, reaching high success rates earlier and achieving the highest fixed-goal success rate in all four environments. VALP-XY isolates the effect of our goal learning algorithm: replacing it with coordinate supervision substantially reduces performance. This comparison provides evidence that value-aware goal learning can offer advantages over a predefined coordinate-based interface. Together, these results address **(Q2)**, showing that combining value-aware temporal abstraction with model-based control enables effective and sample-efficient long-horizon control.

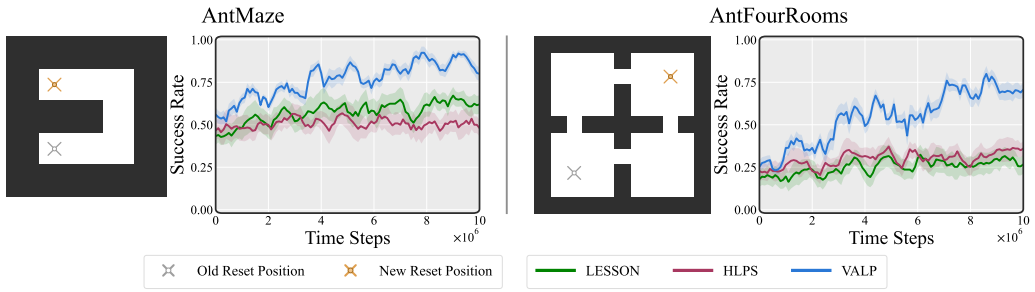



Figure 4: **VALP learns goal representations that remain useful as the task changes.** Fine-tuning in `AntMaze` and `AntFourRooms` after shifting the reset position from the training configuration (gray) to the opposite side of the environment (orange), while keeping the learned goal representation fixed. Curves show the mean over five seeds, with shading denoting standard error.

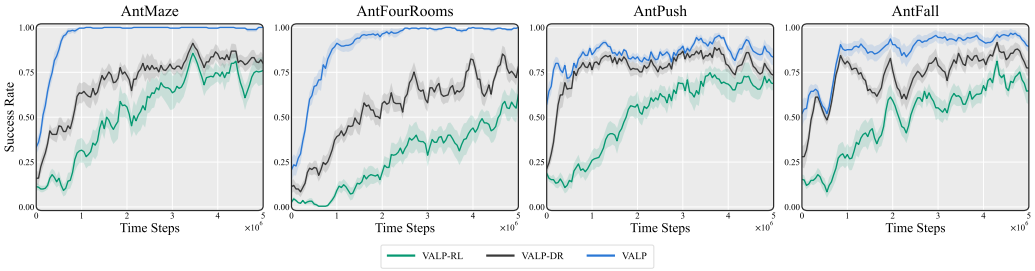



Figure 5: **Value-aware goal learning and model-based control are key design choices in VALP.** Success rate throughout training compared with VALP-DR, which uses dynamics-based goal learning, and VALP-RL, which replaces model-based control with a learned low-level policy.

### 4.3 Goal Representation Transfer

*Setup.* We evaluate whether learned goal representations remain useful when the agent is initialized in a different region (cf. Figure 4). After training, we freeze the goal representation and low-level control mechanism and retrain only the high-level actor and critic. For VALP, this freezes the world model and value-aware representation; for LESSON and HLPS, also the goal-conditioned low-level policy. We change the Ant's reset position to the opposite side of the environments, requiring the high-level policy to solve the task from an unseen region using the fixed abstraction and controller.

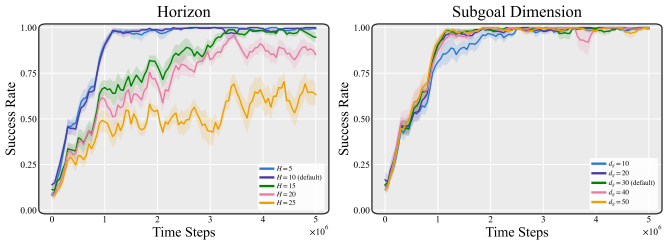
*Results.* Figure 4 shows a clear difference in transfer after shifting the reset position. LESSON and HLPS exhibit little recovery in success rate during fine-tuning, despite retraining their high-level policies. In contrast, VALP recovers a substantial fraction of its original performance while keeping the learned representation and low-level control mechanism fixed. Although the environment dynamics and geometry remain unchanged, the new initialization requires a different sequence of intermediate decisions, testing whether the learned interface can be reused beyond the original task configuration. VALP's stronger recovery is consistent with its goal space retaining decision-relevant distinctions that the high-level policy can exploit from different regions. These results address **(Q3)**, supporting the reusability of VALP's combined representation and planning interface, although they do not isolate the contribution of representation quality from the low-level controller.

### 4.4 Analysis

We analyze four key design choices in VALP: low-level goal tracking, task alignment of the goal representation, planning horizon, and latent goal dimensionality. We compare model-based control with a learned goal-conditioned policy and contrast our value-aware goals with dynamics-only representations across all four environments, and then study planning horizon and goal dimensionality.

*Task-Aligned Goal Representation.* We examine the role of task-relevant value information in shaping the learned goal space. We replace our value-aware objective with a dynamics-based objective following Li et al. (2021b), while keeping all other components of VALP unchanged. Figure 5 shows that VALP consistently outperforms this variant, denoted VALP-DR, showing the benefit of incorporating value information beyond dynamics when learning goals for hierarchical control.

*Low-Level Goal Realization.* We next replace the model-predictive controller in VALP with a learned goal-conditioned RL policy, denoted VALP-RL, while retaining the same learned goal rep-




| *Horizon* | | *Subgoal Dimension* | |
|---|---|---|---|
| $H = 5$ | $0.99_{\pm 0.00}$ | $d_g = 10$ | $1.00_{\pm 0.00}$ |
| $H = 10^{\dagger}$ | $1.00_{\pm 0.00}$ | $d_g = 20$ | $1.00_{\pm 0.00}$ |
| $H = 15$ | $0.96_{\pm 0.02}$ | $d_g = 30^{\dagger}$ | $1.00_{\pm 0.00}$ |
| $H = 20$ | $0.88_{\pm 0.02}$ | $d_g = 40$ | $1.00_{\pm 0.00}$ |
| $H = 25$ | $0.65_{\pm 0.06}$ | $d_g = 50$ | $1.00_{\pm 0.00}$ |

Figure 6: **VALP is robust across horizon and subgoal dimensions.** Planning horizon $H$ and subgoal dimension $d_g$ on Large `AntMaze`, varying one parameter at a time around the default ($H = 10$, $d_g = 30$, †). Curves show the mean and standard error over five seeds; the table reports final success rate and standard errors.

resentation. Figure 5 shows a substantial performance gap in favor of VALP. Unlike a learned low-level policy, which must adapt as the goal representation and selected-goal distribution evolve during training, model-based control replans using the current model and goal representation without requiring a separate policy to track this changing interface. These results support planning as an effective mechanism for realizing learned abstract goals.

*Horizon.* We sweep the planning horizon $H \in \{5, 10, 15, 20, 25\}$. Short horizons, particularly $H = 5$ and $H = 10$, achieve nearly identical performance and rapidly converge to near-perfect success. Beyond $H = 10$, performance progressively degrades, consistent with the increasing difficulty of optimizing longer action sequences and compounding prediction errors. Strong performance at short horizons suggests that the low-level planner need not reason far into the future to realize learned subgoals. Instead, VALP uses short-horizon model-predictive control for goal realization while delegating longer-horizon decisions to the high-level policy.

*Subgoal Dimension.* We sweep the latent subgoal dimension $d_g \in \{10, 20, 30, 40, 50\}$. Performance remains consistently near-perfect across the entire range, with only minor differences in convergence speed. This robustness indicates that VALP is largely insensitive to the precise dimensionality of the goal space over a broad range, suggesting that the learned value-aware representation can provide effective planning objectives under substantially different degrees of latent compression.

Together, these experiments highlight the complementary roles of value-aware goal learning and model-based goal realization in VALP. Value information improves the learned hierarchical interface, while model-based control realizes evolving goals without an additional goal-conditioned policy; performance also remains robust across a broad range of goal dimensions and short planning horizons. These findings support both components as key design choices, answer **(Q4)**, and, together with our preceding experiments, support the central premise of VALP. Value-aware goals provide an effective interface for temporal abstraction and model-based control.

## 5 Conclusion

We introduced VALP, a model-based hierarchical RL framework for long-horizon decision-making. Our results suggest that the communication interface between controllers with different temporal scales need not be predefined or learned solely from transition structure. Instead, we argue value information can instead shape what is communicated, while planning provides a flexible mechanism for acting on these learned objectives. VALP instantiates this perspective by learning compact, value-aware goals and realizing them through repeated planning in a learned world model. Across our experiments, this combination improves performance and sample efficiency across diverse long-horizon control settings and remains useful when the task configuration changes. More broadly, these results motivate separating the learning of *what* should be communicated across temporal scales from *how* those objectives are realized.

Several directions could extend this perspective beyond the setting considered here. First, replacing our sampling-based planner with a learned diffusion policy could provide a more expressive mechanism for realizing complex or multimodal behaviors (Chi et al., 2023). Second, scaling VALP to richer observations and larger world models could broaden the learned interface beyond the state-based domains considered here. In particular, action-conditioned video world models such as V-JEPA 2 (Assran et al., 2025) provide a promising foundation for learning value-aware abstractions directly from high-dimensional visual predictions. Finally, the separation between high-level RL and model-based low-level control creates a natural interface for incorporating explicit control constraints. Potential applications to safety-critical control present another interesting direction.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with software development and to refine the clarity, grammar, and presentation of author-written text. We did not use generative AI tools for research ideation, formulation of the research questions, or development of the core methodology. AI-assisted code was reviewed by the authors and tested for correctness before being incorporated into the experimental codebase. Similarly, AI-assisted edits to the manuscript were reviewed and revised by the authors to ensure that they accurately reflected the intended technical content, results, and claims. Generative AI was not used to autonomously conduct experiments, interpret experimental results, or determine the scientific conclusions of the work. We take responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work involves studies in simulated continuous-control environments and does not involve human subjects, personal data, or sensitive datasets. We do not identify any significant ethical concerns specific to the methodology or experimental setting considered in this work.

### REPRODUCIBILITY STATEMENT

We use standard MuJoCo-based continuous-control benchmarks and describe the environments, evaluation protocol, implementation details, and hyperparameters in the main paper and Appendix. All reported results are obtained over multiple random seeds, with the corresponding experimental settings documented to facilitate replication. Our implementation builds on publicly available benchmark environments and baseline codebases where applicable. The full source code, configuration files, and scripts required to reproduce the experiments will be released publicly upon acceptance.

## REFERENCES

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019. 2

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. 2, 3, 9

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 14

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023. 9

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018. 13

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018. 2, 13

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018. 4, 6, 16, 21

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019. 3, 13

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022. 13

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024. 2, 3, 6, 13, 14, 15, 21

Nick Hansen, Jyothir SV, Vlad Sobal, Yann LeCun, Xiaolong Wang, and Hao Su. Hierarchical world models as visual whole-body humanoid controllers. In *International Conference on Learning Representations*, volume 2025, pp. 62175–62195, 2025. 13

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022. 13

Junsu Kim, Younggyo Seo, and Jinwoo Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. *Advances in neural information processing systems*, 34:28336–28349, 2021. 13

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999. 4

Basil Kouvaritakis, Mark Cannon, et al. *Model predictive control: Classical, robust and stochastic*, volume 1. Springer, 2016. 3, 14

Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022. 2

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017. 1, 13

Siyuan Li, Jin Zhang, Jianhao Wang, Yang Yu, and Chongjie Zhang. Active hierarchical exploration with stable subgoal representation learning. *arXiv preprint arXiv:2105.14750*, 2021a. 1, 13

Siyuan Li, Lulu Zheng, Jianhao Wang, and Chongjie Zhang. Learning subgoal representations with slow dynamics. In *International Conference on Learning Representations*, 2021b. 1, 6, 8, 13, 14, 16, 17, 21, 22

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019. 14

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018. 1, 6, 13, 14, 21

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1emus0qF7. 1, 13, 16, 17

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 7559–7566. IEEE, 2018. 13

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023. 1, 13

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021. 1

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pp. 1049–1065. PMLR, 2021. 3, 4, 14, 22

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887. URL https://doi.org/10.1002/9780470316887. 3

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pp. 1622–1633. PMLR, 2022. 2

David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 1

Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. Learning goal embeddings via self-play for hierarchical reinforcement learning. *arXiv preprint arXiv:1811.09083*, 2018. 13

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018. 1, 4, 15

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

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. 6

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. Efficientzero v2: Mastering discrete and continuous control with limited data. *arXiv preprint arXiv:2403.00564*, 2024a. 2

Vivienne Huiling Wang, Tinghuai Wang, Wenyan Yang, Joni kristian Kamarainen, and Joni Pajarinen. Probabilistic subgoal representations for hierarchical reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=b6AwZauZPV. 6, 13, 14, 16, 17, 21

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 1714–1721. IEEE, 2017. 3, 14

Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4974–4981. IEEE, 2025. 14

Mehdi Zadem, Sergio Mover, et al. Goal space abstraction in hierarchical reinforcement learning via set-based reachability analysis. In *2023 ieee international conference on development and learning (icdl)*, pp. 423–428. IEEE, 2023. 13

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Advances in neural information processing systems*, 33:21579–21590, 2020. 1, 13

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026. 3, 13

# APPENDIX

This appendix provides additional experimental, architectural, and theoretical details that complement the main text. Section A provides additional discussion of related work and connections to prior approaches. Section B reports the hyperparameters, and Section C details the learning objectives. Sections D and E describe the experimental settings and present extended results, respectively. Section F explains the off-policy correction procedure. Section G reports computational cost and runtime details. Section H presents the supporting theoretical derivations. Finally, Sections I and J provide implementation details for the baselines and the iCEM planner.

## A RELATED WORK

**Hierarchical Reinforcement Learning.** Goal-conditioned HRL decomposes long-horizon control across temporal scales, with a high-level policy selecting subgoals and a goal-conditioned low-level policy realizing them through primitive actions (Levy et al., 2017; Nachum et al., 2018). Early approaches define this interface directly from the state space: Levy et al. (2017) use target states, while HIRO (Nachum et al., 2018) operates over a predefined state subspace. Subsequent methods improve subgoal selection within such spaces, for example through reachability constraints (Zhang et al., 2020) or state-space landmarks (Kim et al., 2021). However, predefined goal spaces can become difficult to use when observations are high-dimensional or contain task-irrelevant information. This has motivated learning compact subgoal representations. Sukhbaatar et al. (2018) learn goal embeddings through self-play, while NOR (Nachum et al., 2019) derives a representation objective from hierarchical suboptimality. LESSON (Li et al., 2021b) learns temporally slow features, HESS (Li et al., 2021a) augments this objective with active exploration, and HLPS (Wang et al., 2024b) models long-range correlations and uncertainty through a probabilistic goal representation. GARA (Zadem et al., 2023) instead constructs abstractions from reachability relations. These methods primarily organize the goal space using transition or reachability structure; VALP instead couples the representation to the high-level value function so that the learned objectives preserve distinctions relevant to long-horizon return. HIQL (Park et al., 2023) is the closest prior work to VALP in using value information to learn a compact subgoal representation. However, HIQL operates offline, where the fixed dataset avoids the continual co-adaptation between the policy, critic, data distribution, and goal representation that arises in VALP's online setting. VALP further separates value learning from goal compression by distilling a richer latent value function into the goal space through a value imitator. Rather than requiring a learned low-level policy to track this evolving representation, VALP realizes goals through world-model-based receding-horizon planning, reducing an additional source of policy co-adaptation.

**World-model-based control.** Learned dynamics models enable model-predictive control by optimizing primitive actions online rather than amortizing behavior into a policy. Early methods demonstrated this in continuous control through neural dynamics models (Nagabandi et al., 2018), probabilistic ensembles (Chua et al., 2018), and visual prediction (Ebert et al., 2018). More recent approaches move planning into compact latent spaces: PlaNet (Hafner et al., 2019) performs latent-space planning from images, while TD-MPC (Hansen et al., 2022) and TD-MPC2 (Hansen et al., 2024) learn task-oriented latent world models for short-horizon trajectory optimization. Despite their strong closed-loop control, these methods remain fundamentally single-timescale: long-horizon behavior must ultimately be induced through finite-horizon primitive-action planning and value estimates. Recent work introduces temporal abstraction into world-model-based control. Director (Hafner et al., 2022) learns a manager and goal-conditioned worker, Puppeteer (Hansen et al., 2025) performs model-based control through a predefined humanoid reference-motion interface, and HWM (Zhang et al., 2026) plans through world models operating at multiple temporal scales. VALP differs in making the interface itself learned and task-aware: a high-level policy selects objectives in a value-aware latent goal space, and the same representation defines the objective optimized by the low-level world-model planner. This couples long-horizon strategic decisions to short-horizon closed-loop control without requiring either primitive-action planning to span the full task horizon or a learned low-level policy to realize evolving goals.

**Optimization-based control.** Optimization-based control computes action sequences online from a predictive model and objective rather than amortizing the entire control strategy into a learned policy. Model predictive control (MPC) repeatedly solves a finite-horizon trajectory optimization

problem and replans after each executed action (Kouvaritakis et al., 2016). Sampling-based variants extend this approach to nonlinear dynamics and general, potentially nonconvex or non-differentiable objectives. MPPI (Williams et al., 2017) optimizes control sequences from sampled trajectories, while iCEM (Pinneri et al., 2021) improves sampling efficiency through temporally correlated action sampling and reuse of previous trajectories. Recent work such as DIAL-MPC (Xue et al., 2025) further demonstrates the effectiveness of sampling-based optimization for high-dimensional control. VALP builds on these methods as a local control mechanism rather than relying on them for long-horizon strategic reasoning. A slower high-level policy selects value-aware latent objectives, while the optimizer focuses on realizing each objective over a short horizon. This temporal abstraction allows optimization-based control to operate within its effective local planning regime while long-horizon decisions are handled at the higher level.

## B Hyperparameters

Table 2 summarizes the hyperparameters used in our experiments.

**Networks.** We parameterize our networks as MLPs whose hidden layers apply LayerNorm (Ba et al., 2016) and a Mish activation (Misra, 2019). We apply SimNorm (Hansen et al., 2024) to the outputs of the observation encoder and latent dynamics model. Specifically, each output is partitioned into groups of eight coordinates, and a softmax is applied within each group, constraining it to a probability simplex.

**Hierarchy.** Prior HRL methods rescale high-level actor outputs to pre-defined action ranges, including coordinate-dependent ranges in HIRO (Nachum et al., 2018), corresponding action-space ranges in LESSON (Li et al., 2021b; Wang et al., 2024b). To avoid specifying such displacement scales separately for each environment, we use a normalized goal representation with a common high-level action range.

The high-level actor outputs a displacement $\Delta g_t \in [-1, 1]^{d_g}$ in the learned goal representation space. The planner's absolute target is obtained by adding this displacement to the current state's goal representation, without clipping the resulting target. We bound the goal representation to $[-c, c]^{d_g}$ using

$$\psi_\theta(z) = c \tanh\left(\frac{u_\theta(z) - \mu}{c\sqrt{v + \epsilon}}\right),$$

where $u_\theta(z) \in \mathbb{R}^{d_g}$ denotes the LayerNorm-normalized output projection of the goal compressor before running-statistics normalization and the scaled $\tanh$. The vectors $\mu$ and $v$ are running per-coordinate means and variances of $u_\theta(z)$, respectively. All operations are elementwise, and $\epsilon = 10^{-6}$ ensures numerical stability. This parameterization fixes the representation's range while adapting its centering and scale during training, providing a common coordinate scale for high-level goal selection and world-model planning. We estimate the statistics from sampled high-level transitions. We update them once per gradient step as:

$$\mu \leftarrow (1 - \eta_{\text{stat}})\mu + \eta_{\text{stat}}\hat{\mu}, \qquad v \leftarrow (1 - \eta_{\text{stat}})v + \eta_{\text{stat}}\hat{v},$$

where $\hat{\mu}$ and $\hat{v}$ are the current batch statistics.

## C Learning Objectives

We complement the learning formulation in Section 3.1 with the objectives and implementation details deferred from the main text. In particular, we specify the latent dynamics and high-level value-learning losses in Eq. 1, and provide additional details on entropy-temperature adaptation. Primitive transitions are sampled from $\mathcal{D}_{\text{lo}}$, while high-level segments are sampled from $\mathcal{D}_{\text{hi}}$.

**World-model.** The encoder and dynamics model are concurrently trained by minimizing the one-step prediction error,

$$\mathcal{L}_{\text{dyn}} = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[ \| f_\theta(\phi_\theta(s_t), a_t) - \text{sg}[\phi_\theta(s_{t+1})] \|_2^2 \right]. \tag{7}$$

The next-state representation is treated as a stop-gradient target, ensuring that the dynamics model predicts toward the next latent encoding rather than reducing the loss by modifying the target itself.

Table 2: Hyperparameters for VALP.

| **Hyperparameter** | |
|---|---|
| *Hierarchy* | |
| Horizon $H$ | 10 |
| Subgoal dimension (Ant / CarPark) | 30 / 5 |
| Latent state dimension | 512 |
| Goal-representation bound | 3.0 |
| Goal statistics momentum ($\eta_{\text{stat}}$) | 0.001 |
| *Networks* | |
| Encoder / actor / critic hidden dims | $[256, 256, 256]$ |
| Latent dynamics hidden dims | $[512, 512, 512, 512]$ |
| Goal-representation hidden dims | $[256, 256, 256]$ |
| *iCEM* | |
| Initial population size $N$ | 300 |
| Optimization iterations | 3 |
| Population decay $\gamma_p$ | 1.25 |
| Number of elites $K$ | 10 |
| Momentum coefficient $\rho$ | 0.1 |
| Initial sampling std. $\sigma_{\text{init}}$ | 0.5 |
| Variance floor | 0.025 |
| Colored-noise exponent $\beta$ | 2.0 |
| Elite reuse fraction | 0.3 |
| Shifted-elite fraction | 0.3 |
| *Off-policy Correction* | |
| Candidate subgoals | 256 |
| Candidate noise std. | 0.5 |
| *Optimization* | |
| Optimizer | AdamW |
| Learning rate | $3 \times 10^{-4}$ |
| Temperature learning rate | $7 \times 10^{-3}$ |
| Batch size | 256 |
| Discount factor $\gamma_h$ | 0.99 |
| Target network update | 0.005 |
| Distillation loss weight ($\lambda_{\text{dist}}$) | $1 \times 10^{-4}$ |
| Critic loss weight ($\lambda_Q$) | $1 \times 10^{-3}$ |
| *Training* | |
| Evaluation environments | 10 |
| High-level buffer size | $3 \times 10^5$ |
| Low-level buffer size | $1 \times 10^6$ |
| Evaluation interval | $5 \times 10^4$ |

**Value function.** We train $Q_\theta^{\text{hi}}$ using soft temporal-difference learning on high-level transitions. For a segment starting at $\tau$ with length $\ell_\tau \leq H$, let $R_\tau = \sum_{j=0}^{\ell_\tau - 1} r_{\tau+j}$ be its accumulated reward. With $z_\tau = \phi_\theta(s_\tau)$ and $z_\tau^+ = \phi_\theta(s_{\tau+\ell_\tau})$, the target is

$$y_\tau = R_\tau + \gamma_h \left( Q_{\bar{\theta}}^{\text{hi}}(z_\tau^+, \Delta g_\tau^+) - \alpha \log \pi_\vartheta^{\text{hi}}(\Delta g_\tau^+ \mid z_\tau^+) \right), \tag{8}$$

where $\Delta g_\tau^+ \sim \pi_\vartheta^{\text{hi}}(\cdot \mid z_\tau^+)$, $\bar{\theta}$ denotes slowly updated target-critic parameters, and $\gamma_h$ is the discount applied per high-level transition. Following TD-MPC2 (Hansen et al., 2024), we use a distributional cross-entropy objective for value prediction:

$$\mathcal{L}_Q = \mathbb{E}_{\mathcal{D}} \left[ \text{CE}\big( Q_\theta^{\text{hi}}(z_\tau, \Delta g_\tau), \text{sg}(y_\tau) \big) \right]. \tag{9}$$

This loss updates the critic and encoder while holding the target fixed via stop-gradient operator. Following standard practice, we update the target critic using an exponential moving average of the online critic parameters (Sutton & Barto, 2018).

**Temperature updates.** We automatically tune the entropy temperature $\alpha$ using the SAC objective (Haarnoja et al., 2018),

$$\mathcal{L}_\alpha = -\mathbb{E}\left[\log \alpha \, \mathrm{sg}\left[\log \pi^{\mathrm{hi}}_\vartheta(\Delta g \mid z_\tau) + \mathcal{H}_{\mathrm{target}}\right]\right], \tag{10}$$

where $\Delta g \sim \pi^{\mathrm{hi}}_\vartheta(\cdot \mid z_\tau)$ and $\mathcal{H}_{\mathrm{target}}$ is the target entropy. The stop-gradient isolates this update from the high-level policy, such that minimizing $\mathcal{L}_\alpha$ adjusts only $\alpha$ toward the prescribed entropy level.

# D Experimental Details

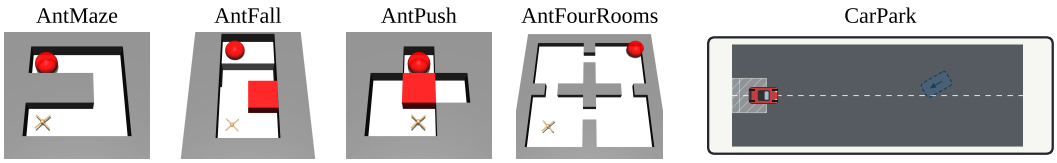



Figure 7: Environments we use in our experiments.

## D.1 Car Parking Environment

We consider a parking task in an open rectangular arena with boundary walls and no internal obstacles, as illustrated in Fig. 7. The car follows kinematic bicycle dynamics, with continuous actions controlling acceleration and steering rate, allowing both forward and reverse motion. The observation is $(x, y, \cos\theta, \sin\theta, v, \delta)$, where $(x, y)$ denotes the rear-axle position, $\theta$ the heading, $v$ the signed speed, and $\delta$ the normalized steering angle. The target specifies a parking position and heading. Reaching the target therefore requires coordinating navigation and orientation. The car starts at rest, facing along the arena, from a position sampled within a designated start region. Target positions are sampled within a corridor beyond this region, with random headings.

## D.2 MuJoCo Environments

We evaluate VALP on four long-horizon continuous-control environments based on the MuJoCo Ant agent: `AntMaze`, `AntPush`, `AntFall`, and `AntFourRooms` (see Fig. 7). Following prior work (Nachum et al., 2019; Li et al., 2021b; Wang et al., 2024b), the environments are constructed from cubic maze blocks. During training, target locations are sampled from the navigable regions of the environment. For evaluation we provide both results where the goal is randomly sampled from the same regions of training and on a fixed challenging target.

**`AntMaze`.** `AntMaze` is a U-shaped navigation environment in which the Ant must reach a target located at the opposite end of the maze. The maze blocks a direct route to the target, forcing the Ant to follow the U-shaped corridor around the central wall before approaching the target from the opposite side. The environment is constructed from $4 \times 4 \times 4$ blocks, resulting in corridors of width 4.

**`AntPush`.** `AntPush` introduces object interaction into the navigation problem. A movable $4 \times 4 \times 4$ block obstructs the route to the target. Directly pushing the block forward eventually blocks the passage; instead, the Ant must first move around the block and push it aside to clear the path before proceeding toward the target.

**`AntFall`.** `AntFall` extends the navigation problem to three dimensions. The Ant and target are located on a raised platform separated by a chasm that cannot be crossed directly. A movable $4 \times 4 \times 4$ block is provided near the Ant and can be pushed into the chasm to form a bridge. The Ant must first move toward the block, push it into the gap, and then safely traverse it to reach the target. Accordingly, the task goal is specified in three dimensions, $(x, y, z)$.

**`AntFourRooms`.** `AntFourRooms` is a substantially larger navigation environment consisting of four rooms connected by narrow passages. The Ant must navigate between rooms through the corresponding doorways to reach the target, requiring a sequence of intermediate navigation decisions over a longer horizon. The environment uses $2 \times 2 \times 2$ maze blocks, allowing the larger four-room topology while maintaining narrow passages that require precise locomotion.

### D.3 Settings

**Dense Reward.** In the dense-reward setting, the agent receives a shaped reward based on its $\ell_2$-distance to the target throughout the episode, providing continuous feedback on task progress.

**Sparse Reward.** In the sparse-reward setting, intermediate progress is not rewarded. The agent receives a reward of $0$ only upon reaching the target and $-1$ otherwise, making exploration and long-horizon credit assignment substantially more challenging.

**Large Environments.** To evaluate performance over longer decision horizons without changing the underlying task structure, we additionally consider enlarged versions of all four environments under the dense-reward setting. We double the physical dimensions of the maze blocks: blocks in `AntMaze`, `AntPush`, and `AntFall` are increased from $4 \times 4 \times 4$ to $8 \times 8 \times 8$, while those in `AntFourRooms` are increased from $2 \times 2 \times 2$ to $4 \times 4 \times 4$. The maze topology and task structure remain unchanged, but the distances that must be traversed are doubled. Therefore, episode lengths are also doubled.

**Image Observations.** For the image-based variants of these environments, we use the dense-reward setting, remove the Ant's $x$- and $y$-coordinates from the state observation and augment the remaining proprioceptive features with a low-resolution $5 \times 5 \times 3$ top-down view of the local environment, following prior work (Nachum et al., 2019; Li et al., 2021b; Wang et al., 2024b).

## E Extended Results

### E.1 Car Parking Environment

**Learning Curves.** As shown in Figure 8, VALP learns the car parking task substantially faster and converges to near-perfect success, while LESSON and HLPS remain at markedly lower success rates throughout training. Unlike the navigation benchmarks, CarParking requires coordinated position and heading control under kinematic vehicle dynamics. These results extend the main benchmark findings to a qualitatively different long-horizon control problem, suggesting that VALP can generalize across tasks with different dynamics and control requirements.

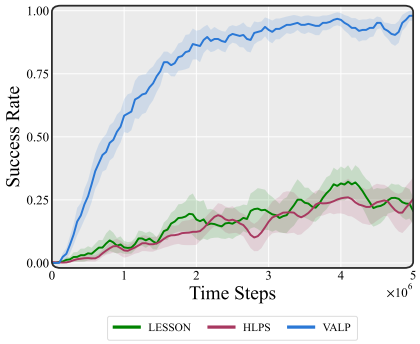



Figure 8: **VALP extends effectively to a distinct long-horizon control task.** Learning curves show the success rate of VALP, LESSON, and HLPS on the CarParking environment. Training curves report success rate on randomly sampled goals. Curves report the mean over five seeds, with shaded regions denoting standard error.

**Goal Representation.** Figure 9 shows additional successful parking trajectories and the corresponding normalized representation distance to the final configuration. The same qualitative pattern as in Fig. 2 appears across these examples: VALP remains most consistently aligned with task progress, with its distance consistently decreasing as the car approaches the target pose. LESSON is broadly aligned but exhibits more fluctuations, while HLPS shows larger deviations and is less consistent throughout the maneuver. These results further support that VALP learns a goal representation that better tracks task-relevant progress.

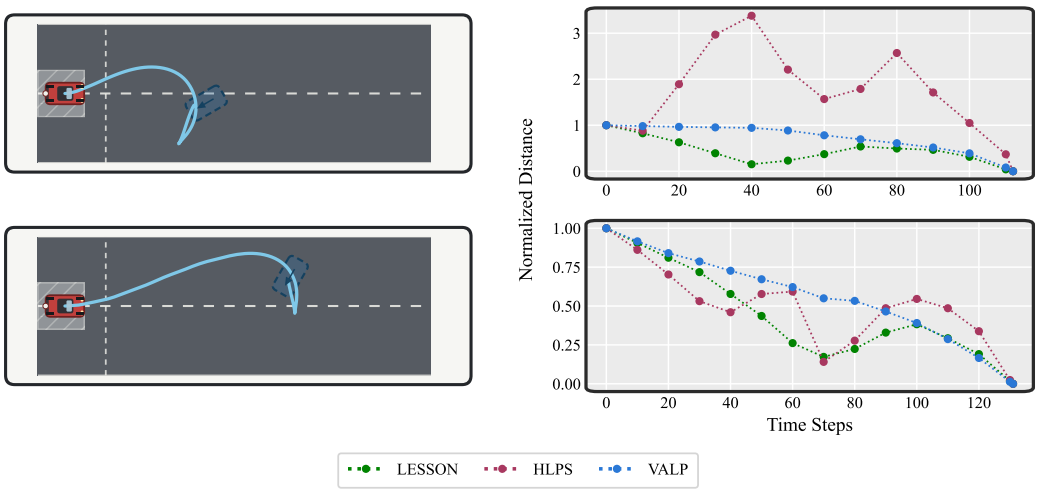



Figure 9: **VALP learns goal representations consistently aligned with task progression.** Additional successful parking trajectories comparing goal-representation distance to the final configuration for VALP, LESSON, and HLPS.

### E.2 MuJoCo Environments

**Benchmark.** Table 3 reports the final success rates corresponding to the benchmark learning curves in Figure 3. VALP achieves the highest average performance across all configurations, with an average success rate of 94.6%, compared with 85.0% for VALP-XY, 83.8% for TD-MPC2, and lower averages for the remaining baselines. VALP attains the highest final success rate in 13 of the 16 configurations, highlighting its consistency across reward structures, observation modalities, and task scales.

Table 3: **VALP performs consistently across diverse benchmark settings.** Final success rates are reported across all environments and settings with randomly sampled evaluation goals, averaged over five independent seeds.

| Task | Setting | SAC | TD-MPC2 | HIRO | LESSON | HLPS | VALP-XY | VALP |
|---|---|---|---|---|---|---|---|---|
| AntMaze | Dense | 48.4±18.2 | 98.8±0.8 | 76.8±14.5 | 98.4±1.0 | 92.4±1.5 | 89.6±1.5 | **100**±0.0 |
| | Sparse | 21.2±12.6 | 99.2±0.8 | 68.8±19.1 | 97.2±1.4 | 84.0±6.3 | 84.8±5.9 | **100**±0.0 |
| | Image | 60.8±21.6 | 96.8±1.0 | – | 78.4±5.1 | 69.2±11.7 | – | **100**±0.0 |
| | Large | 82.0±6.0 | 87.2±1.4 | 98.0±2.0 | 71.2±9.1 | 84.0±3.7 | 91.6±2.0 | **100**±0.0 |
| AntFourRooms | Dense | 58.0±20.5 | 91.6±1.3 | 72.4±17.7 | 58.4±6.1 | 73.6±4.6 | 92.8±2.6 | **99.6**±0.4 |
| | Sparse | 8.4±3.1 | 81.2±1.8 | 70.8±14.5 | 65.2±3.8 | 82.4±3.0 | 84.4±2.6 | **99.2**±0.5 |
| | Image | 87.6±2.3 | 81.2±2.2 | – | 29.2±2.0 | 20.8±2.8 | – | **98.8**±0.5 |
| | Large | 68.8±17.2 | 76.8±2.3 | **98.0**±0.6 | 22.0±3.2 | 64.8±3.5 | 95.6±1.2 | 96.4±1.9 |
| AntPush | Dense | 74.8±7.2 | 79.6±0.4 | **92.8**±3.1 | 78.4±1.6 | 91.2±2.9 | 83.2±1.8 | 86.0±3.9 |
| | Sparse | 41.2±7.9 | 79.6±1.0 | 73.6±9.5 | 78.0±1.9 | 86.0±2.5 | 78.8±1.0 | **89.6**±4.1 |
| | Image | 73.6±16.4 | 87.2±2.9 | – | 74.8±6.5 | 68.8±3.9 | – | **98.4**±0.4 |
| | Large | 48.0±17.2 | 77.6±1.7 | 85.2±4.0 | 80.0±1.9 | 76.8±1.6 | 82.8±1.4 | **94.0**±3.3 |
| AntFall | Dense | 70.8±5.0 | 81.6±2.6 | 72.0±2.0 | 75.6±2.9 | 64.4±2.0 | 85.6±1.9 | **92.0**±1.4 |
| | Sparse | 38.4±12.7 | 73.2±3.2 | 61.6±4.5 | **77.2**±2.6 | 57.2±2.9 | 72.0±2.6 | 71.2±4.5 |
| | Image | 62.8±3.3 | 80.4±4.4 | – | 71.2±1.8 | 50.8±2.7 | – | **94.8**±1.4 |
| | Large | 59.6±1.8 | 68.8±1.6 | 79.6±5.4 | 67.2±5.0 | 58.4±2.8 | 78.8±1.2 | **93.6**±0.8 |
| Average | | 56.5 | 83.8 | 79.1 | 70.2 | 70.3 | 85.0 | **94.6** |

**Fixed Challenging Target.** Figure 10 and Table 4 report performance when every method is evaluated on a fixed challenging target. The advantage of VALP becomes substantially more pronounced in this setting: it achieves an average success rate of 78.2%, compared with 53.3% for the next-best method. The learning curves show that this gap is not limited to final performance; VALP typically reaches high success earlier and maintains it more consistently. Flat world-model control with TD-MPC2 remains strong on `AntMaze`, but degrades sharply on tasks requiring longer-horizon co-

ordination, indicating that accurate short-horizon control alone is insufficient for reliably reaching difficult distant objectives. Hierarchical baselines remain competitive in individual configurations, but their performance varies considerably across tasks and settings. The substantial gap between VALP and VALP-XY further highlights the role of the learned value-aware goal space: retaining the same world-model controller while restricting the high-level interface to coordinate-based goals leads to a large drop in average performance. Together, these results indicate that coupling long-horizon goal selection with closed-loop world-model planning is particularly beneficial when success requires repeatedly realizing a specific, difficult sequence of intermediate objectives.

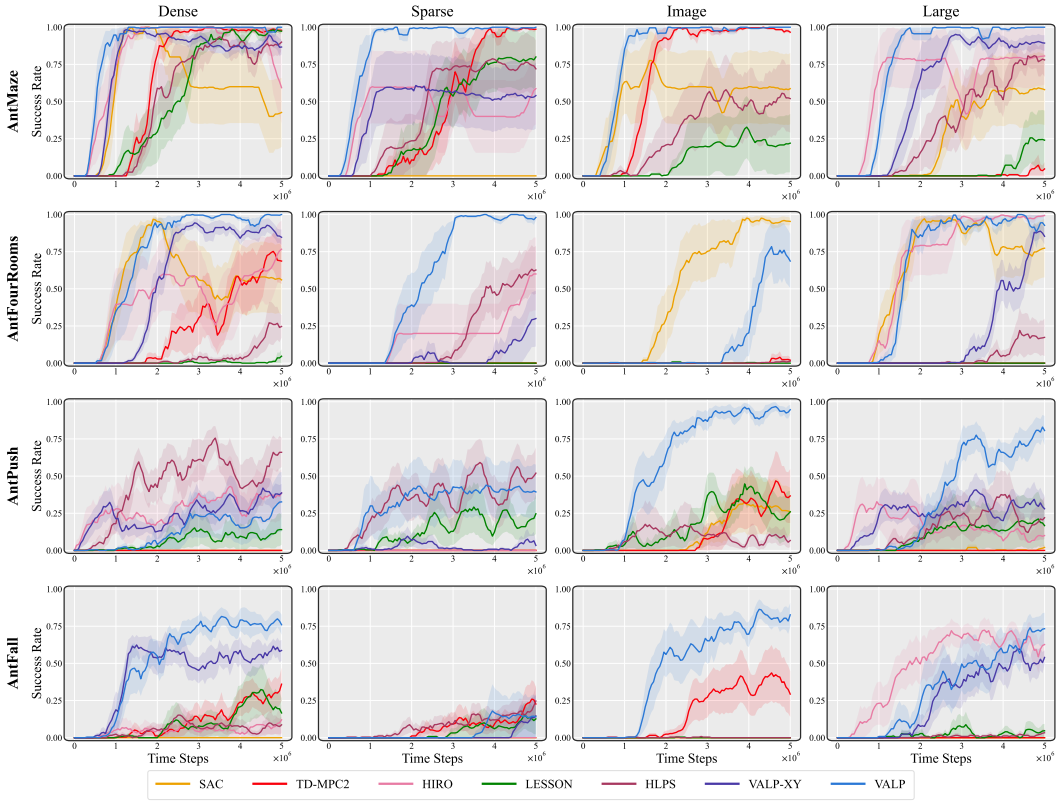



Figure 10: **VALP is more robust to fixed challenging targets.** Training curves report success rate on a fixed challenging evaluation target across environments (rows) and settings (columns). Curves show the mean over five runs, with shaded area denoting standard error.

## F Off-Policy Correction

VALP trains from replayed trajectory segments collected under earlier versions of the planner and the learned goal representation. As both evolve, a stored displacement $\Delta g_\tau$ may specify a different objective in the current goal space, and the current planner may produce different behavior when pursuing it. The stored displacement may therefore no longer be consistent with the segment's observed outcome under the current controller. To mitigate this mismatch, we relabel sampled high-level transitions before gradient updates.

For a segment starting at $\tau$ with length $\ell_\tau$, we construct a candidate set $\mathcal{C}(\Delta g_\tau)$ containing the stored displacement and Gaussian perturbations clipped to $[-1, 1]^{d_g}$. We select

$$\widetilde{\Delta g}_\tau = \arg\min_{\delta \in \mathcal{C}(\Delta g_\tau)} \left\| \psi_\theta(z_{\tau+\ell_\tau}) - \left[ \psi_\theta(z_\tau) + \delta \right] \right\|_2, \qquad z_t = \phi_\theta(s_t). \tag{11}$$

This selects the candidate whose implied absolute goal is closest to the observed endpoint under the current representation. The selected displacement $\widetilde{\Delta g}_\tau$ replaces the stored displacement $\Delta g_\tau$ for high-level critic learning and value distillation, while the segment reward and observed states remain unchanged.

Table 4: **VALP achieves the strongest overall performance on fixed challenging targets.** Final success rates are reported across all environments and settings, averaged over five independent seeds.

| Task | Setting | SAC | TD-MPC2 | HIRO | LESSON | HLPS | VALP-XY | VALP |
|---|---|---|---|---|---|---|---|---|
| AntMaze | Dense | 41.6±22.4 | 98.0±1.1 | 66.8±20.7 | 98.0±1.6 | 88.0±5.3 | 89.2±3.0 | **100**±0.0 |
| | Sparse | 0.0±0.0 | 98.8±0.5 | 54.0±22.7 | 78.8±19.2 | 74.4±19.1 | 52.8±21.7 | **99.6**±0.4 |
| | Image | 58.4±23.9 | 97.6±0.4 | – | 21.6±19.2 | 52.4±21.6 | – | **100**±0.0 |
| | Large | 58.8±24.0 | 4.0±4.0 | 80.4±19.6 | 24.4±19.0 | 77.6±10.2 | 90.0±1.8 | **100**±0.0 |
| AntFourRooms | Dense | 56.8±23.2 | 69.2±12.8 | 73.2±19.0 | 3.6±3.6 | 26.8±8.0 | 86.4±2.5 | **99.6**±0.4 |
| | Sparse | 0.0±0.0 | 0.0±0.0 | 58.8±24.0 | 0.0±0.0 | 63.2±15.8 | 29.2±18.4 | **97.2**±2.3 |
| | Image | **95.6**±1.2 | 2.4±2.4 | – | 0.0±0.0 | 0.8±0.8 | – | 72.8±14.7 |
| | Large | 76.4±19.1 | 0.0±0.0 | **98.8**±0.5 | 0.0±0.0 | 16.8±11.4 | 88.8±3.3 | 94.4±4.7 |
| AntPush | Dense | 0.0±0.0 | 0.0±0.0 | 37.2±16.0 | 13.6±10.8 | **64.0**±7.9 | 38.0±3.4 | 31.2±19.1 |
| | Sparse | 0.0±0.0 | 0.0±0.0 | 0.0±0.0 | 21.2±10.2 | **48.4**±14.1 | 5.6±5.1 | 39.2±17.0 |
| | Image | 26.8±16.5 | 36.0±14.9 | – | 23.2±14.4 | 7.2±3.7 | – | **92.4**±2.7 |
| | Large | 1.2±0.8 | 0.0±0.0 | 10.0±10.0 | 18.8±16.4 | 21.2±13.0 | 32.0±4.2 | **79.6**±6.5 |
| AntFall | Dense | 0.0±0.0 | 30.0±8.3 | 9.6±7.7 | 19.6±11.1 | 8.0±4.2 | 60.8±3.3 | **80.0**±4.0 |
| | Sparse | 0.0±0.0 | **26.0**±10.8 | 0.0±0.0 | 12.8±8.8 | 18.4±5.1 | 13.6±13.6 | 14.4±14.4 |
| | Image | 0.0±0.0 | 34.0±14.2 | – | 0.0±0.0 | 0.0±0.0 | – | **80.0**±5.9 |
| | Large | 0.0±0.0 | 0.0±0.0 | 59.6±5.7 | 3.6±3.6 | 2.4±1.5 | 52.8±5.8 | **71.6**±6.2 |
| Average | | 26.0 | 31.0 | 45.7 | 21.2 | 35.6 | 53.3 | **78.2** |

## G Computation Time

We compare action-selection and update times on a single `AntMaze` environment using an NVIDIA RTX 5090 GPU. For a fair comparison, we use a horizon of $H = 10$ steps for all hierarchical methods. After compilation, we measure action selection and updates separately over 100 consecutive environment steps, grouped into ten windows of ten steps. Each action window covers one complete subgoal segment. Update timings follow each method's training update-to-data ratio and are amortized per environment step. Environment simulation and replay sampling are excluded.

Table 5 reports the mean and standard error across the ten window averages. VALP has a similar action-selection cost to TD-MPC2, while both are slower than direct policy execution. Conversely, VALP requires less update time per environment step than HIRO under its' respective training schedule.

Table 5: Action-selection and amortized update time in milliseconds per environment step, reported as mean ± standard error over ten 10-step window averages from one run.

| Method | Action (ms/step) | Update (ms/step) |
|---|---|---|
| SAC | $0.133 \pm 0.001$ | $0.140 \pm 0.001$ |
| TD-MPC2 | $2.015 \pm 0.002$ | $0.314 \pm 0.001$ |
| HIRO | $0.200 \pm 0.002$ | $1.641 \pm 0.008$ |
| VALP | $1.936 \pm 0.008$ | $0.377 \pm 0.002$ |

## H Proof of Value Preservation

We provide the proof of Proposition 3.1. Recall that $Q_\theta^{\text{hi}}$ denotes the fixed high-level critic, $\widetilde{Q}_\theta$ its compressed imitator, and $\psi$ the mapping from latent states to the learned goal representation. The high-level action is a goal displacement $\Delta g$. We assume that the compressed critic uniformly approximates the high-level critic,

$$\sup_{z,\Delta g} \left| Q_\theta^{\text{hi}}(z, \Delta g) - \widetilde{Q}_\theta(\psi(z), \Delta g) \right| \leq \epsilon, \tag{12}$$

and that $\widetilde{Q}_\theta$ is uniformly $L$-Lipschitz in its representation argument, i.e.,

$$\left| \widetilde{Q}_\theta(u_1, \Delta g) - \widetilde{Q}_\theta(u_2, \Delta g) \right| \leq L \|u_1 - u_2\|_2 \tag{13}$$

for all representations $u_1, u_2$ and high-level actions $\Delta g$.

*Proof of Proposition 3.1.* Consider arbitrary latent states $z_1, z_2$ and fix a high-level action $\Delta g$. Adding and subtracting the corresponding predictions of the compressed critic gives

$$\begin{aligned}\left|Q_\theta^{\mathrm{hi}}(z_1,\Delta g) - Q_\theta^{\mathrm{hi}}(z_2,\Delta g)\right| = \Big|&Q_\theta^{\mathrm{hi}}(z_1,\Delta g) - \widetilde{Q}_\theta(\psi(z_1),\Delta g)\\ &+ \widetilde{Q}_\theta(\psi(z_1),\Delta g) - \widetilde{Q}_\theta(\psi(z_2),\Delta g)\\ &+ \widetilde{Q}_\theta(\psi(z_2),\Delta g) - Q_\theta^{\mathrm{hi}}(z_2,\Delta g)\Big|.\end{aligned} \tag{14}$$

Applying the triangle inequality yields

$$\begin{aligned}&\left|Q_\theta^{\mathrm{hi}}(z_1,\Delta g) - Q_\theta^{\mathrm{hi}}(z_2,\Delta g)\right|\\ &\le \left|Q_\theta^{\mathrm{hi}}(z_1,\Delta g) - \widetilde{Q}_\theta(\psi(z_1),\Delta g)\right|\\ &\quad + \left|\widetilde{Q}_\theta(\psi(z_1),\Delta g) - \widetilde{Q}_\theta(\psi(z_2),\Delta g)\right|\\ &\quad + \left|\widetilde{Q}_\theta(\psi(z_2),\Delta g) - Q_\theta^{\mathrm{hi}}(z_2,\Delta g)\right|.\end{aligned} \tag{15}$$

By the uniform approximation assumption in eq. 12, the first and third terms in eq. 15 are each bounded by $\epsilon$. Applying the Lipschitz condition in eq. 13 to the second term gives

$$\left|\widetilde{Q}_\theta(\psi(z_1),\Delta g) - \widetilde{Q}_\theta(\psi(z_2),\Delta g)\right| \le L\left\|\psi(z_1) - \psi(z_2)\right\|_2. \tag{16}$$

Combining these bounds,

$$\left|Q_\theta^{\mathrm{hi}}(z_1,\Delta g) - Q_\theta^{\mathrm{hi}}(z_2,\Delta g)\right| \le 2\epsilon + L\left\|\psi(z_1) - \psi(z_2)\right\|_2. \tag{17}$$

Since eq. 17 holds for every $\Delta g$, taking the supremum over the high-level action establishes

$$\sup_{\Delta g}\left|Q_\theta^{\mathrm{hi}}(z_1,\Delta g) - Q_\theta^{\mathrm{hi}}(z_2,\Delta g)\right| \le 2\epsilon + L\left\|\psi(z_1) - \psi(z_2)\right\|_2, \tag{18}$$

which proves Proposition 3.1. □

**Scope of the result.** Proposition 3.1 is conditional on uniform approximation and Lipschitz assumptions. The empirical distillation objective used by VALP minimizes imitation error over sampled transitions and therefore does not itself establish the uniform bound in eq. 12. Moreover, the result is one-directional: proximity in the learned representation bounds differences in high-level value predictions, but similar value predictions need not imply proximity in representation space. Most importantly, the proposition does not establish reachability or controllability in the learned goal space. Value-equivalent states may differ in their local controllability, and Euclidean distance in the representation need not correspond to control effort or temporal distance. These properties are therefore not implied by Proposition 3.1; VALP instead uses learned latent dynamics and model-predictive replanning to track selected targets.

# I Baseline Implementations

We describe the source and any modifications for each baseline. Unless stated otherwise, we use the hyperparameters recommended by the original authors and evaluate all methods under the common protocol described in Sec. 4 .

**SAC and TD-MPC-2.** We implement SAC (Haarnoja et al., 2018), HIRO (Nachum et al., 2018) and TD-MPC-2 (Hansen et al., 2024) in JAX, following the implementations described in the original papers.

**HIRO.** We implement HIRO (Nachum et al., 2018) in JAX following the algorithmic formulation of the original paper. We use the Ant's planar $(x, y)$ position as the two-dimensional subgoal space.

**LESSON and HLPS.** We use the official implementations released by the authors of LESSON (Li et al., 2021b) and HLPS (Wang et al., 2024b), retaining their published hyperparameter settings without modification.

**VALP-XY.** We replace VALP's value-aware goal representation with a two-dimensional representation trained to predict the Ant's ground-truth $(x, y)$ coordinates, while keeping the remaining architecture unchanged.

**VALP-RL** We replace VALP's model-predictive low-level controller with a learned goal-conditioned RL policy trained to reach the same value-aware latent goals, while keeping the remaining architecture unchanged.

**VALP-DR.** We replace VALP's value-aware goal representation with a representation learned from transition dynamics using the slowness-based objective of Li et al. (2021b), while keeping the remaining architecture unchanged.

## J iCEM

Improved Cross-Entropy Method (iCEM) iteratively refines a sampling distribution over candidate action sequences (Pinneri et al., 2021). At each iteration, sequences are sampled and evaluated through the dynamics model, and the highest-scoring candidates, called elites, are used to update the distribution's mean and variance. Temporally correlated sampling, elite reuse, and a progressively smaller population improve sampling efficiency. After optimization, the first action of the best evaluated sequence is executed, and planning is repeated from the next observed state.

Within VALP, iCEM serves as the sampling-based trajectory optimizer for low-level goal realization. We follow the original paper for the core optimization procedure, while evaluating candidate action sequences through the learned latent dynamics using terminal Euclidean distance to the high-level subgoal as the planning objective. Following the paper's fixed hyperparameters (Table S5), we use $K = 10$ elites, initial standard deviation $\sigma_{\text{init}} = 0.5$, momentum coefficient $\rho = 0.1$, population decay factor $\gamma_p = 1.25$, and an elite reuse fraction of $0.3$. Following its low-budget learned-model setting, we use three optimization iterations, with an initial population of 300 candidate sequences. Sampling uses temporally correlated Gaussian noise with exponent $\beta = 2$, which we select specifically for VALP, followed by clipping to the action bounds. Elites are reused between optimization iterations and shifted between consecutive control steps. Across consecutive replanning steps before the planning horizon resets, the sampling mean is warm-started from the previous solution, while the variance is reset at each solve. The mean candidate is evaluated in the final iteration, and the first action of the best evaluated sequence is executed.