HIERARCHICAL REINFORCEMENT LEARNING WITH TEMPORALLY-EXTENDED LATENT ACTION WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Models enable counterfactual reasoning and planning but struggle in long horizon tasks because prediction errors compound over the horizon. Temporal abstraction can mitigate these errors by simply requiring fewer consecutive predictions for a given horizon. We propose a Temporally-extended Latent Action World Model (Te-LAWM) and present a hierarchical model-based framework for goal-conditioned reinforcement learning that leverages it to generate subgoals for a value-based model-free policy. Our temporally-extended latent action model learns a multi-step action abstraction, a multi-step dynamics model conditioned on these abstract actions (the Te-LAWM), and a prior policy over the actions within a single objective. Subgoal planning is done by rolling out the prior policy within the Te-LAWM. We demonstrate superior performance of our framework compared to flat model-based and hierarchical model-free approaches on benchmark navigation and manipulation tasks. Ablations show that autoregressive subgoal generation allows the agent to stitch together experience from different trajectories, and that temporal action abstraction is essential for scaling world model predictions to long horizons and large action spaces.

1 INTRODUCTION

World Models (Sutton, 1991; Ha & Schmidhuber, 2018) have long been studied in the context of model-based Reinforcement Learning (RL) for simulating environment interactions useful for sample-efficient policy learning (Deisenroth & Rasmussen, 2011; Janner et al., 2019; Hafner et al., 2020) and planning (Hafner et al., 2019; Hansen et al., 2022; Zhou et al., 2025). However, the benefits of world models for long-horizon tasks have largely not been realized due to the curse of horizon—single-step prediction errors inevitably compound over the horizon, rendering long-term predictions unreliable (Asadi et al., 2019). While improving dynamics models' accuracy is necessary for unlocking their full potential, temporal abstraction offers an orthogonal approach to mitigating compounding errors by simply requiring fewer consecutive predictions.

Hierarchical Goal-Conditioned RL (GCRL) (Klissarov et al., 2025; Park et al., 2026b) is one such approach, where high-level modules make temporally-extended predictions that guide shorter-horizon, low-level modules. This structure allows reasoning over longer timescales with fewer iterative steps, reducing compounding error. What role should planning with world models play in such a hierarchy? We suggest that it is most effective at higher levels of abstraction, where only a few autoregressive predictions are required to propose distant subgoals that can efficiently be achieved by model-free low-level policies.

We introduce an offline GCRL framework that learns a high-level world model used to plan subgoals for a value-based model-free RL policy. We propose Temporally-extended Latent Action World Models (Te-LAWMs) as high-level world models, trained via a latent action modeling objective (Schmidt & Jiang, 2024; Gao et al., 2025). Our temporally-extended latent action models conveniently learn (1) a temporally-extended action abstraction, (2) a multi-step dynamics model conditioned on these abstract actions (the Te-LAWM) and (3) a prior policy over the abstract actions, within a single objective. Inferring abstract actions from dataset transitions and learning a corresponding policy allows for subgoal planning while remaining within the support of the dataset, which is key when learning from offline data without further environment interaction.

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.

We evaluate our agent on OGBench (Park et al., 2025) navigation and manipulation environments and compare to representative approaches, including both flat and hierarchical offline GCRL methods spanning various combinations of model-free and model-based components. Our approach matches or outperforms all baselines and does so by a large margin in long-horizon tasks that require stitching. A comprehensive ablation study of the temporally-extended world model design space shows that autoregressive subgoal generation allows the agent to stitch together experience from different trajectories, and that temporal action abstraction is essential for scaling world model predictions to long horizons and large action spaces.

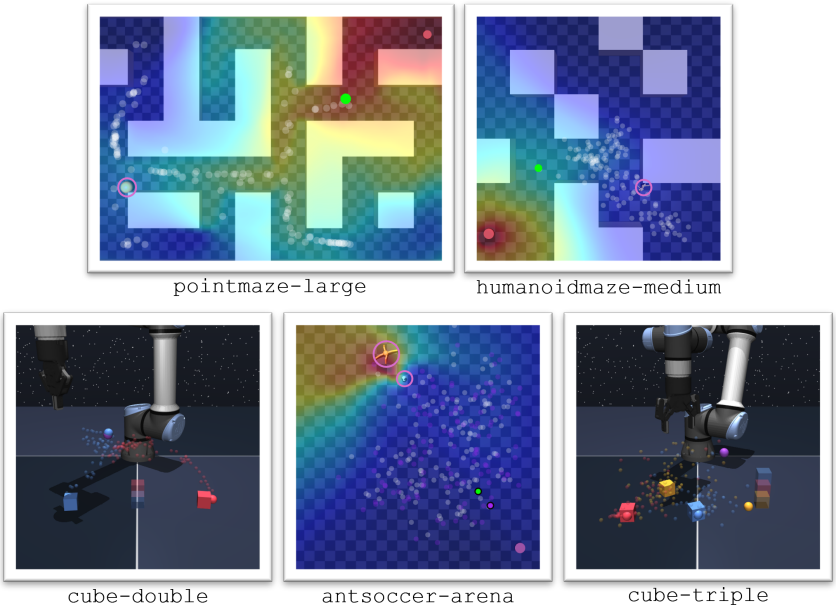


Figure 1: **Subgoals generated by planning with the Te-LAWM.** *Mazes*: the final goal is represented by a red circle. Candidate subgoals are shown in white, with the selected subgoal in green. The heatmap overlaid on the maze represents a learned goal-conditioned value landscape. *Cube Manipulation*: transparent cubes indicate the final target positions for solid cubes with the same color. Small spheres represent the proposed candidate subgoals, while the enlarged spheres denote the chosen subgoal. The purple enlarged sphere signifies the gripper position in the chosen subgoal. *Antsoccer*: the heatmap represents the goal-conditioned value for the position of the ant conditioned on the position of the ball. Subgoals for the ant are shown in white and for the ball in purple.

2 Background & Related Work

Problem Setting (Offline GCRL): GCRL (Kaelbling, 1993) considers a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the environment transition probabilities, $r : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ the reward function and γ the discount factor. We assume the goal space is equal to the state space. We also assume a sparse reward $r(s, g) = -\mathbb{I}[\delta(s, g) > \varepsilon]$, i.e., 0 when at an environment-defined distance threshold ε from the goal and -1 otherwise. The objective is to learn a policy $\pi : \mathcal{S} \times \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$, i.e., learn to reach any state from any other state in the minimum number of timesteps. *Value-based RL* algorithms learn a value function $V^\pi : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ which approximates the expected return, as an intermediate step to producing a policy. *Model-based RL* algorithms approximate $\mathcal{P}$ by learning a dynamics model $d : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, also known as a world model, that can be used for policy learning and planning. Experiments in this work focus on the *offline setting*, where the agent learns from a fixed dataset of state-action trajectories. Extracting performant goal-reaching policies requires improving over the dataset behavior. The goal-conditioned setting presents an additional challenge: learning to reach goals from states where no single dataset trajectory contains both, also refered to as *goal stitching* (Park et al., 2025).

Hierarchical Goal-conditioned Reinforcement Learning exploits subgoal structure in sequential data to reason across multiple levels of temporal abstraction. Various instantiations of subgoal

hierarchies have been proposed for tackling long-horizon tasks including both model-free (Vezhnevets et al., 2017; Nachum et al., 2018; Levy et al., 2019; Chane-Sane et al., 2021) and model-based (Bagaria et al., 2021; Hafner et al., 2022) approaches. In the offline setting, HIQL (Park et al., 2023) extracts both a low-level and high-level policy from a single value function, HECRL Haramati et al. (2026) employ rejection sampling with a subgoal-generating diffusion model while SHARSA (Park et al., 2026b) and OTA (Ahn et al., 2026) learn a separate high-level value function for subgoal generation. Our framework uses a single value function to extract a low-level policy and plan with a temporally-extended latent action world model.

Latent Action Models (LAMs): a LAM is a conditional Variational Autoencoder (VAE; Kingma & Welling, 2013, CVAE; Sohn et al., 2015) that models the distribution $p(s_{t+1} \mid s_t)$ where (s_t, s_{t+1}) is a state transition induced by some policy. It is optimized to maximize the evidence lower bound:

$$\log p(s_{t+1} \mid s_t) \geq \mathbb{E}_{q(z|s_t,s_{t+1})} \left[\log p(s_{t+1} \mid s_t, z)\right] - D_{\mathrm{KL}} \left[q(z \mid s_t, s_{t+1}) \parallel p(z \mid s_t)\right],$$

z denoting the latent variable. Assuming diagonal Gaussian encoder and prior and Gaussian decoder with fixed variance, maximizing the evidence lower bound is equivalent to minimizing the following loss:

$$\mathcal{L} = \left\|s_{t+1} - \hat{s}_{t+1}\right\|^2 + D_{\mathrm{KL}} \left[\mathcal{N}\left(\mu_z^q, \Sigma_z^q\right) \parallel \mathcal{N}\left(\mu_z^p, \Sigma_z^p\right)\right],$$

where the first term is a reconstruction loss and the second is a KL regularization which can be calculated in closed form. Given that the evidence $p(s_{t+1} \mid s_t)$ models state transitions, the approximate posterior (encoder) $q(z \mid s_t, s_{t+1})$ can be interpreted as an *inverse dynamics model* (infers the action that led from s_t to s_{t+1}), the likelihood (decoder) $p(s_{t+1} \mid s_t, z)$ as a *forward dynamics model* and the latent variable z as an action—hence the term *latent action*. To drive z to behave as a latent action and not simply compress s_{t+1}, strong latent space regularization is applied via vector quantization (discrete z; Schmidt & Jiang, 2024) or KL penalties (continuous z; Gao et al., 2025). Latent actions allow training on more abundant action-free data (Bruce et al., 2024; Ye et al., 2025) and can facilitate cross-embodiment transfer (Bu et al., 2025). Compared to the above single-step LAMs, we propose a *temporally-extended* LAM. Variants of multi-step LAMs have been studied in non-goal-conditioned offline settings (Mandlekar et al., 2020; Li et al., 2022). Compared to our framework, these use a fixed latent action prior and either employ complex planning procedures or do not perform multi-step autoregressive planning.

Temporally-extended World Models for Sequential Decision-making aim to mitigate compounding errors in long-horizon autoregressive predictions. A straightforward design conditions directly on action sequences (Asadi et al., 2019; Park et al., 2026a) while others condition on learned embeddings thereof (Shi et al., 2022; Zhang et al., 2026), options (Sutton et al., 2023; Rodriguez-Sanchez & Konidaris, 2024) and representations inferred from action-free multi-step state transitions (Li et al., 2022; Chitnis et al., 2024). Our Te-LAWM belongs to the latter category and we compare to other approaches in Sec. 4.2. Gumbsch et al. (2024) propose adaptive-timescale world models that predict dynamics context switches. Other classes of variable-timescale world models include policy-conditioned generative models of discounted future state occupancy (Janner et al., 2020; Farebrother et al., 2026) and horizon-conditioned world models (Chung et al., 2026). Our model is trained on fixed-horizon targets and, similar to single-step world models, makes variable-timescale predictions via autoregressive composition.

3 Temporally-extended Latent Action World Models

Long-horizon tasks present significant learning challenges rooted in compounding error. This limits both single-step world models and value functions, which rely on composing their own predictions to reason over longer horizons. This property is also what enables composing different behavior segments to produce new ones, i.e., stitching. We seek to obtain the benefits of model-based stitching and planning while mitigating compounding errors via temporal abstraction. How to design effective temporally-extended world models suitable for planning and how these should be used by an agent, remains an open question.

While goal-conditioned value functions can provide a useful estimate of state-goal distances, they may be less effective for extracting policies for distant goals, where discounting and approximation error obscures low-level action advantages. This has motivated extracting policies as well as subgoals for those policies using a single value function in both hierarchical (Park et al., 2023) and

flat (Zhou & Kao, 2026) offline GCRL methods. We present a hierarchical framework that leverages a temporally-extended world model to generate subgoals for a low-level value-based agent. Compared to previous methods, our model-based approach enables stitching on the high level, i.e., "subgoal stitching" (see Fig. 2 and 5). We describe our world model design and its role in our proposed framework in the following.

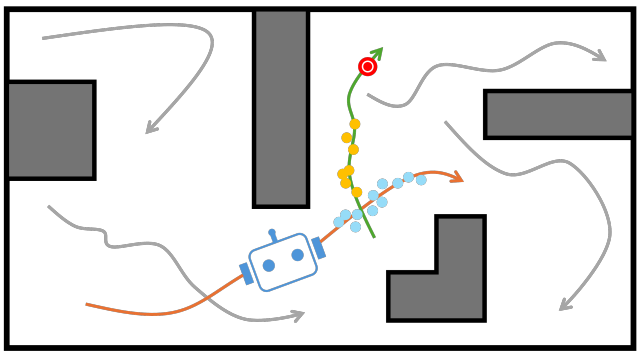


Figure 2: **Subgoal Stitching.** Curved arrows represent trajectories in the offline dataset. Blue circles denote subgoals sampled from future states within the same dataset trajectory the agent is traversing. Yellow circles denote subgoals from states not observed within the same trajectory, that lead to the goal (red concentric circles). Generating them from the agent's current state requires composing future state predictions from different dataset trajectories.

3.1 The Temporally-Extended Latent Action World Model

The Te-LAWM is simply the dynamics model component of a LAM (see Section 2) which is trained on multi-step next state transitions (s_t, s_{t+K}) rather than single timestep transitions (s_t, s_{t+1}), for some fixed K. The temporally-extended LAM is comprised of a multi-step inverse dynamics model $I_\theta\left(s_t, s_{t+K}\right) \to \left(\mu_z^{I_\theta}, \Sigma_z^{I_\theta}\right)$, a multi-step forward dynamics model $D_\theta\left(s_t, z\right) \to \hat{s}_{t+K}$ and a learned prior $P_\theta\left(s_t\right) \to \left(\mu_z^{P_\theta}, \Sigma_z^{P_\theta}\right)$ over the latent actions z. We are interested in continuous control and thus use continuous latent actions. The model is optimized to minimize the following loss:

$$\mathcal{L}_\theta = \left\|s_{t+K} - D_\theta\left(s_t, z\right)\right\|^2 + \beta \cdot D_{\mathrm{KL}}\left[\mathcal{N}\left(\mu_z^{I_\theta}, \Sigma_z^{I_\theta}\right) \;\|\; \mathcal{N}\left(\mu_z^{P_\theta}, \Sigma_z^{P_\theta}\right)\right].$$

The first term is the forward dynamics prediction loss given the latent action sampled from the inverse dynamics model $z \sim \mathcal{N}\left(\mu_z^{I_\theta}, \Sigma_z^{I_\theta}\right)$. The second term weighted by a hyperparameter β (following the β-VAE formulation (Higgins et al., 2017)), regularizes the latent space to contain less information about s_{t+K} (P_θ is only a function of s_t) such that it behaves as a latent action and does not simply compress s_{t+K}. Our choice of a learned per-state prior as opposed to the commonly used standard prior $z \sim \mathcal{N}\left(0, I\right)$ conveniently serves another key role—a policy over the latent actions. After training, we discard I_θ and remain with our Te-LAWM D_θ and the latent action policy P_θ for downstream sequential decision-making.

3.2 Hierarchical Reinforcement Learning with Te-LAWMs

The Te-LAWM $D_\theta : \mathcal{S} \times \mathcal{Z} \to \mathcal{S}$ coupled with the latent action policy $P_\theta : \mathcal{S} \to \mathcal{P}\left(\mathcal{Z}\right)$ trained on offline trajectories capture the dataset K-step-next-state distribution. These enable planning for future states while remaining within the support of the offline dataset. We present a hierarchical framework where the Te-LAWM facilitates high-level subgoal planning for a low-level value-based GCRL agent, consisting of a policy $\pi_\phi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ and value function $V_\phi : \mathcal{S} \times \mathcal{G} \to \mathbb{R}$. We leverage the fact that in our setting $-V_\phi$ approximates an asymmetric discounted timestep distance between states (Wang et al., 2023) for subgoal planning. Our framework is modular—the levels of the hierarchy are trained separately and are only related by the offline datasets they were trained on. They are composed at test time in the following manner: (1) N skip-state trajectories of length T, $\left(s_t, s_{t+K}, s_{t+2K} \ldots, s_{t+TK}\right)$, are generated by autoregressively rolling out latent actions sampled from P_θ. The set of final trajectory states serve as subgoal candidates $\{s^i_{t+TK}\}_{i=1}^N$; (2) Given a goal g, the candidates are scored using the value function and the highest valued candidate (shortest

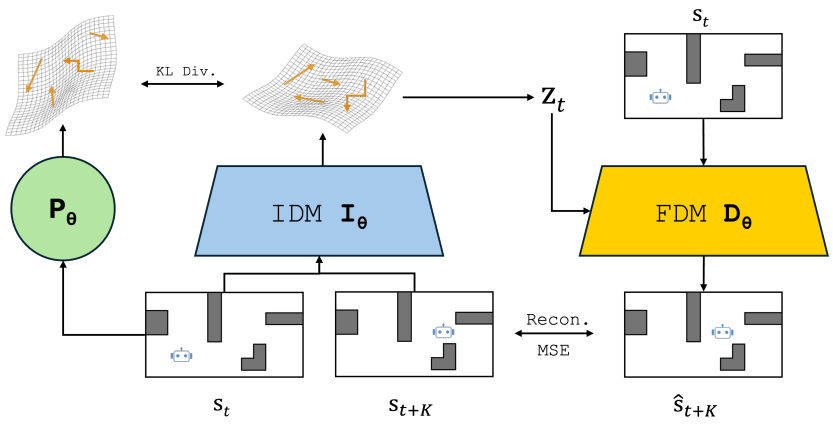


Figure 3: **Temporally-extended Latent Action World Model.** The Inverse Dynamics Model (IDM) I_θ predicts a latent action z_t describing the temporally-extended transition between the current state s_t and a multi-step future state s_{t+K}. The Forward Dynamics Model (FDM) D_θ then predicts the result of applying the latent action in the current state to reconstruct s_{t+K}. Concurrently, a state-conditioned prior (policy) is trained to regularize the latent action space. All components are optimized jointly using a combined loss objective comprising an MSE s_{t+K} reconstruction loss and a β-weighted KL divergence between the outputs of the IDM and the prior.

temporal distance to the goal) is selected as the subgoal $\tilde{g} \leftarrow \arg\max_{\tilde{g} \in \{s^i_{t+TK}\}} V_\phi(\tilde{g}, g)$; (3) The low-level policy conditioned on $\tilde{g}$, $\pi_\phi(s_t, \tilde{g})$, is executed until either reaching the subgoal (measured by a threshold on the value $V_\phi(s_t, \tilde{g})$) or after a timeout period T_{sg}. Steps (1)-(3) are repeated until the agent reaches the goal vicinity (measured by a threshold on the value $V_\phi(s_t, g)$), after which the subgoal is set to the final goal g. See Alg. 1 (App.) for pseudo-code format and Fig. 1 for visualization of generated subgoals in our experiment environment suite.

Note that compared to standard LAMs, our framework does not require mapping latent actions to the original action space for sequential decision-making since they are solely used as a control interface for generating subgoal states. That said, latent actions constitute a natural interface for model-based subgoal generation, as the effects of a latent action remain mostly consistent across the state space (Gao et al., 2025). We present qualitative empirical analysis of this property in Sec. 4.3.

4 Experiments

Our experiments are designed to study the capabilities and limitations of model-based and model-free approaches as well as their interplay in a hierarchical system for solving long-horizon tasks. The core aspects we seek to shed light on are: (1) our framework's performance on long-horizon goal-conditioned tasks compared to baselines; (2) whether planning with the Te-LAWM can reliably perform goal stitching; (3) key choices in the design of our temporally-extended world model. Code to reproduce the experiments in this paper will be released upon publication.

Environments and Datasets: We evaluate our framework and baselines on navigation and manipulation OGBench (Park et al., 2025) environments and datasets in an offline goal-conditioned setting with sparse rewards and termination signals. We refer to the pairing of environment-dataset as a task. *Navigation*: requires learning both low-level locomotion and high-level navigation behaviors. They vary in 3 ways—maze size, embodiment and dataset trajectory length (compared to `navigate`, `stitch` datasets contain significantly shorter trajectories). This allows studying how methods scale with respect to horizon, state-action dimensionality and goal stitching respectively. *Manipulation*: requires learning manipulation of various objects (cubes, buttons, sliding drawer/window) to goal configurations. They vary in 2 ways—number of objects and dataset collection policies (`play` and `noisy`). This allows studying how methods scale with respect to horizon and goal stitching as well as the effects of dataset coverage respectively. We train agents for $1M$ gradient steps with a batch

size of 1024 and report mean and standard deviation of success rates of the final checkpoint across 5 seeds for all experiments.



antmaze-large cube-triple scene

Figure 4: **Environments**. Representative environments from our evaluation suite.

Implementation and Baselines: We design our baselines to represent related offline GCRL approaches while sharing as many components across methods as possible, for a principled comparison that isolates the key aspects of interest from potential confounding factors. Specifically: (1) value functions are trained once per seed and are shared across all methods, which use them in different capacities; (2) all modules (e.g., policies, value functions, dynamics models) are implemented with the same MLP architecture; (3) all modules operate directly on the observation space. Thus, *names of baselines reflect the methods which they are inspired by and are not necessarily an exact implementation of the original*. We briefly present them here and provide further implementation details and hyperparameters in App. A.

IQL: a goal-conditioned variant of Implicit Q-Learning (Kostrikov et al., 2022). It learns both a value function and a Q-function trained independently from a policy. The policy is extracted via a DDPG+BC-style (Fujimoto & Gu, 2021) loss. The IQL value functions are trained once per seed and shared across all baselines. **TD-MPC**: planning with a learned single-step world model using MPPI (Williams et al., 2015) initialized with actions from a prior policy. The IQL value function is used for trajectory scoring and the IQL policy serves as the prior. This method is based on IQL-TD-MPC (Chitnis et al., 2024) which introduces an offline variant of TD-MPC (Hansen et al., 2022). **SAW**: Zhou & Kao (2026) distill a hierarchical policy into a single network by bootstrapping predictions from a short-horizon subgoal policy. The IQL value function is used to extract this flat policy via AWR (Peng et al., 2019). **HIQL**: an agent based on Hierarchical IQL (Park et al., 2023) which extracts both a high-level and a low-level policy from an IQL value function. **SD**: Subgoal Diffuser generates subgoals with a goal-conditioned diffusion model. The IQL value function is used to score candidates and the IQL policy for low-level action execution. **Te-LAWM (ours)**: the Temporally-extended Latent Action World Model paired with an IQL agent, including both value function for subgoal selection and low-level policy for action execution. Table 1 compares these methods across aspects of interest to our study.

Table 1: **Comparison of algorithmic aspects across Te-LAWM and baselines**.

Attribute	IQL	TD-MPC	SAW	HIQL	SD	Te-LAWM (ours)
Model-based	✗	✓	✗	✗	✓	✓
Hierarchical	✗	✗	✗	✓	✓	✓
Goal-agnostic Subgoal Module	-	-	-	✗	✗	✓
Test-time Subgoal Selection	-	-	-	✗	✓	✓
Test-time Forward Passes	1	$1+\frac{H\times I_{mppi}}{T_{replan}}$	1	$1+\frac{1}{T_{replan}}$	$1+\frac{N_{denoise}+1}{T_{replan}}$	$1+\frac{N_{step}+1}{T_{replan}}$

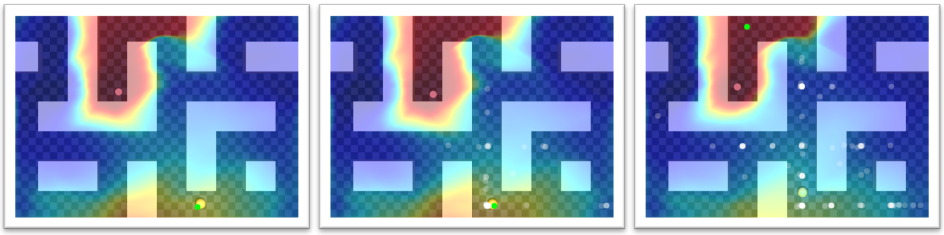
4.1 Main Results

The main results are presented in Tab. 2. Te-LAWM outperforms or matches all baselines on our task suite. **Long-horizon Reasoning and Goal Stitching**: hierarchical methods generally perform better than flat methods on navigation tasks. Notably, Te-LAWM surpasses other approaches by a significant margin in the `maze-large-stitch` tasks. We attribute this to the world model's ability to imagine subgoals outside of the dataset future state distribution by stitching subgoals via autoregressive prediction (see example in Fig. 5). **World Model Goal-conditioning**: SD differs from Te-LAWM in the subgoal-generating model—it uses a goal-conditioned diffusion model in contrast to a non-goal-conditioned VAE. It performs worse in navigation and comparable in manipulation.

In the navigation tasks we observe generalization failures where SD produces subgoals towards the goal that disregard the wall positions (i.e., subgoals are not compatible with the temporal distance in the dataset; see App. Fig. 10). **In-distribution Planning**: The flat IQL and TD-MPC perform similarly on navigation but differ in manipulation tasks. Interestingly, the model-based TD-MPC improves performance on `noisy` datasets but degrades it on `play` datasets. This may be explained by TD-MPC's reliance on good state-action coverage since planned actions can steer away from the support of the dataset, especially for longer rollouts. We observe similar failure modes in the maze environments where the agent plans trajectories that intersect with the walls (see App. Fig. 10). The model-based Te-LAWM on the other hand, suffers less from this problem since it makes fewer autoregressive predictions and captures the dataset distribution via the learned latent actions and policy.

Table 2: **Main Results.** Success rates averaged across 5 seeds. See App. A for method implementation details. Best results up to a standard deviation are highlighted in **bold**.

Environment	Dataset	IQL	TD-MPC	SAW	HIQL	SD	Te-LAWM
pointmaze	pointmaze-medium-navigate-v0	83 ±7	86 ±5	94 ±3	**97** ±2	**99** ±1	**99** ±1
	pointmaze-large-navigate-v0	58 ±3	64 ±4	**96** ±4	**98** ±2	95 ±2	**99** ±1
	pointmaze-medium-stitch-v0	46 ±14	59 ±17	74 ±9	**95** ±3	**99** ±1	**98** ±2
	pointmaze-large-stitch-v0	27 ±10	27 ±9	13 ±5	33 ±8	35 ±7	**84** ±5
antmaze	antmaze-medium-navigate-v0	67 ±4	59 ±4	**93** ±4	**95** ±3	80 ±5	**94** ±7
	antmaze-large-navigate-v0	50 ±9	44 ±9	42 ±5	75 ±5	57 ±14	**82** ±5
	antmaze-medium-stitch-v0	41 ±9	45 ±3	**87** ±7	**87** ±4	75 ±3	**88** ±7
	antmaze-large-stitch-v0	46 ±9	36 ±7	19 ±3	79 ±6	59 ±4	**91** ±2
humanoidmaze	humanoidmaze-medium-navigate-v0	55 ±1	57 ±1	46 ±7	**96** ±3	53 ±4	**97** ±2
	humanoidmaze-large-navigate-v0	61 ±6	60 ±5	2 ±2	59 ±3	34 ±3	**85** ±3
	humanoidmaze-medium-stitch-v0	52 ±9	52 ±2	48 ±6	**95** ±2	57 ±1	**97** ±2
	humanoidmaze-large-stitch-v0	21 ±6	25 ±5	3 ±1	30 ±4	31 ±3	**50** ±3
antsoccer	antsoccer-arena-navigate-v0	**89** ±3	**89** ±3	34 ±2	**87** ±4	**89** ±2	**91** ±4
	antsoccer-arena-stitch-v0	**46** ±7	**52** ±13	6 ±1	**45** ±8	**44** ±3	**44** ±7
	antsoccer-medium-navigate-v0	23 ±4	20 ±2	7 ±4	27 ±4	18 ±5	**34** ±4
cube	cube-single-play-v0	89 ±3	60 ±6	81 ±6	**91** ±3	**93** ±2	**94** ±3
	cube-double-play-v0	76 ±4	32 ±7	54 ±5	60 ±2	**85** ±5	73 ±6
	cube-triple-play-v0	10 ±4	6 ±3	9 ±5	16 ±4	**23** ±6	15 ±6
	cube-single-noisy-v0	**99** ±1	**100** ±0	**99** ±1	**99** ±1	**99** ±1	**99** ±1
	cube-double-noisy-v0	**97** ±2	**97** ±2	84 ±2	89 ±4	**97** ±2	**95** ±2
	cube-triple-noisy-v0	43 ±4	**58** ±8	43 ±3	43 ±4	**54** ±4	**56** ±4
scene	scene-play-v0	58 ±2	53 ±4	57 ±3	53 ±8	**61** ±5	**66** ±5
	scene-noisy-v0	48 ±5	**57** ±3	**55** ±3	50 ±2	**53** ±8	**53** ±4



HIQL SD Te-LAWM

Figure 5: **Subgoal Generation Comparison**; `pointmaze-large-stitch-v0`. The images contrast the subgoal candidates produced by each hierarchical method. The heatmap overlaid on the maze represents a learned goal-conditioned value landscape for the goal point in red. HIQL produces a single subgoal. SD and Te-LAWM capture a distribution over subgoals (white points) and select the one with the highest value (green point). SD is limited by the in-trajectory future states present in the `stitch` dataset while Te-LAWM stitches dataset trajectories to produce more distant subgoals that allow the agent to progress towards the goal.

4.2 Ablation Study

We ablate the key aspects of our approach, namely: (1) temporal action abstraction; (2) our method for learning that abstraction; (3) autoregressive subgoal prediction; (4) the learned latent action

prior. To this end we design the following methods, respectively: (1) *Action Chunk World Model (ACWM)*—a multi-step dynamics model conditioned on action chunks paired with an action-chunk diffusion policy for planning; (2) *Latent Action Chunk World Model (LACWM)*—a multi-step dynamics model conditioned on latent representations of action chunks from a CVAE (conditioned on the current state), with a learned prior for planning; (3) *Single-step*—the Te-LAWM with a single prediction step for subgoal generation. (4) *Fixed Prior*—the Te-LAWM with a Normal latent action prior. The method implementations follow the same design principles as our baselines (details in App. A). Tab. 3 presents the results. ACWM severely degrades with increased action chunk dimensionality, aligned with the findings of Park et al. (2026a). LACWM on the other hand, performs similarly to Te-LAWM, emphasizing the importance of temporal action abstraction. Compared to LACWM, Te-LAWM is simpler in that it does not require action-labeled data nor a separate dynamics model and is trained with a single objective. Single-step performs worse than Te-LAWM on `stitch` tasks, which highlights the role of autoregressive subgoal prediction for goal stitching. Fixed Prior performs mostly on par with Te-LAWM. We hypothesize that the dynamics model compensates for the prior being less expressive, potentially entangling the roles of the latent action and the dynamics model.

Table 3: **Ablation Results.** Success rates averaged across 5 seeds. See Sec. A for method implementation details. Best results up to a standard deviation are highlighted in **bold**.

Environment	Dataset	ACWM	LACWM	Single-step	Fixed Prior	Te-LAWM
pointmaze	pointmaze-medium-navigate-v0	**100** ± 0	**92** ± 8	**99** ± 1	**99** ± 1	**99** ± 1
	pointmaze-large-navigate-v0	**97** ± 2	**95** ± 4	**90** ± 9	**98** ± 2	**99** ± 1
	pointmaze-medium-stitch-v0	**99** ± 2	**98** ± 0	**97** ± 3	**99** ± 1	**98** ± 2
	pointmaze-large-stitch-v0	81 ± 10	**93** ± 6	41 ± 9	**94** ± 4	84 ± 5
antmaze	antmaze-medium-navigate-v0	68 ± 7	**94** ± 3	84 ± 5	**88** ± 6	**94** ± 7
	antmaze-large-navigate-v0	41 ± 4	**82** ± 6	**77** ± 6	34 ± 7	**82** ± 5
	antmaze-medium-stitch-v0	63 ± 8	80 ± 5	73 ± 4	79 ± 8	**88** ± 7
	antmaze-large-stitch-v0	41 ± 6	**86** ± 6	80 ± 5	83 ± 3	**91** ± 2
humanoidmaze	humanoidmaze-medium-navigate-v0	51 ± 2	89 ± 7	**97** ± 2	**92** ± 5	**97** ± 2
	humanoidmaze-large-navigate-v0	60 ± 10	**82** ± 6	**84** ± 5	**80** ± 7	**85** ± 3
	humanoidmaze-medium-stitch-v0	51 ± 5	**96** ± 3	78 ± 4	**97** ± 2	**97** ± 2
	humanoidmaze-large-stitch-v0	32 ± 7	**51** ± 7	25 ± 4	**54** ± 2	50 ± 3
antsoccer	antsoccer-medium-navigate-v0	20 ± 4	**36** ± 5	**31** ± 4	**34** ± 3	**34** ± 4
manipulation	cube-double-play-v0	**80** ± 8	75 ± 3	58 ± 5	**84** ± 1	73 ± 6
	cube-triple-noisy-v0	**56** ± 6	**54** ± 5	**54** ± 4	**56** ± 6	**56** ± 4
	scene-play-v0	64 ± 3	66 ± 5	66 ± 5	**73** ± 2	66 ± 5
	scene-play-v0-task5	1 ± 2	6 ± 4	**11** ± 7	3 ± 3	**15** ± 8

4.3 Latent Action Space Analysis

We qualitatively study whether the learned latent actions encode a compressed temporally-abstract action that is reusable and has similar effects across the state space (as we intended) or rather encode absolute states, i.e., s_{t+K}. We analyze the structure of the latent action space through visualization and statistics of the effects of latent actions. We focus on `pointmaze-large-navigate` and `cube-single-noisy` which have lower-dimensional state-action spaces compared to other tasks in our suite, making results simpler to visualize.

Pointmaze: Fig. 6 portrays the latent action space and its effects on 4 representative probe states in `pointmaze`. Each maze cell in the top row is color-coded by its position relative to the probed state, where colors correspond to cardinal directions and their brightness to geodesic distances (i.e., shortest distance in the maze). Both the state and latent action spaces are 2-dimensional in this setup, which enables a direct mapping between them. The second row maps latent actions to the colors of the states they lead to when input to the Te-LAWM at the probed state, i.e., $Color\{D_\theta\left(s = s_{probe}, z = (z_1, z_2)\right)\}$. Colors in each column represent cardinal directions *relative* to the probed state in each plot and appear in similar regions *across* plots, indicating some level of consistency across the state space, subject to environment dynamics and dataset support. We additionally visualize the learned policy prior, a 2-dimensional Gaussian, on the latent action space. The prior differs significantly between probed states, capturing the per-state K-step-future-state distribution in the dataset. Probe 3 is positioned in a central intersection of the maze and indeed its prior captures latent actions that lead to all directions while probe 2 is positioned in a "dead-end" and captures a much narrower distribution.

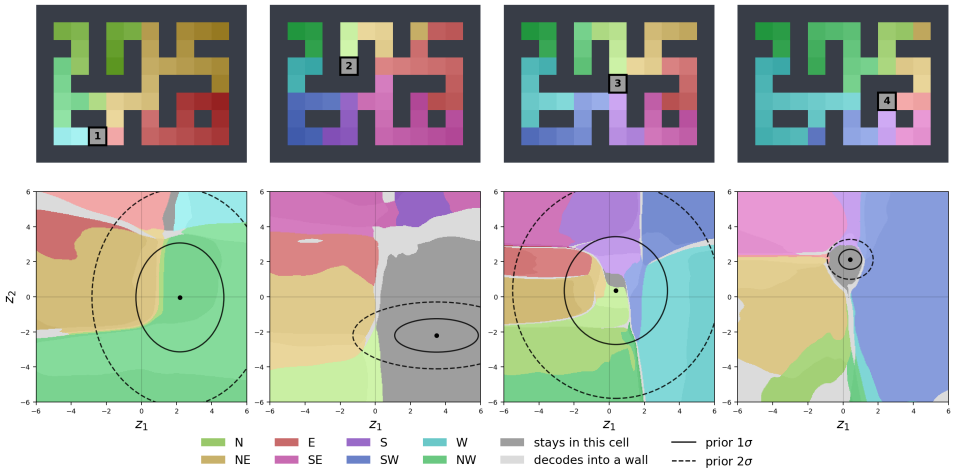


Figure 6: **The Temporally-extended Latent Action Space**; `pointmaze-large-navigate`. *Top*: maze cells colored by their cardinal position relative to the probed state (center of gray numbered cell). Color corresponds to cardinal direction and brightness to geodesic distance. *Bottom*: colors from the above maze mapped to latent actions $z = (z_1, z_2)$ based on where they lead to when input to the Te-LAWM at the probed state. See Sec. 4.3 for analysis.

Cube: We sample 300,000 temporally-extended transitions (s_t, s_{t+K}) from the dataset, K corresponding to the value Te-LAWM was trained with, and label each by events we define: Grasp, Release, Lift, Reach or Other (see App. B.3 for details). For each event (except Other) we draw 30,000 triplets of state transitions $\{A = (s_t^A, s_{t+K}^A), B = (s_t^B, s_{t+K}^B), C = (s_t^C, s_{t+K}^C)\}$ where A and B carry the same label while C carries a different one. Every transition is encoded with the LAM's inverse dynamics model, $z_X = I_\theta(s_t^X, s_{t+K}^X)$. We then measure whether applying z_A at s_t^B, i.e., decoding it with the Te-LAWM D_θ, results in the same event. B is chosen to have the same label as A to ensure the event is affordable in state B (e.g., the cube is close enough to be grasped), and C has a different label as a control that verifies that not all latent actions result in the same event at a given state B. z_A transfers to s_t^B near 100% of cases for Grasp, Release and Lift and 73% of cases for Reach, while z_C results in the same effect less than 28% of cases. See App. Fig. 11 for statistics and Fig. 12 for rendered examples.

5 Conclusion

We study using world models for high-level planning to overcome the curse of horizon limiting their look-ahead and stitching capabilities. We propose a simple design which jointly learns a temporal action abstraction, forward dynamics, and a prior policy under a single objective. We demonstrate its utility for generating subgoals in a hierarchical offline GCRL framework. Our method matches or outperforms baselines on a wide range of tasks, particularly excelling in long-horizon tasks that require composing experience from different dataset trajectories. Ablations show that temporal action abstraction and autoregressive subgoal prediction are properties essential to performance.

Our study isolates method capabilities from challenges in visual representation learning. We present results on visual navigation in App. B.2 using a simple pretrained latent representation as input to the Te-LAWM. Further exploring pretrained compact representations of images or learning them concurrently with latent actions within our framework is left for future work. Te-LAWM inherits the limitations of LAMs, including the inability to distinguish state dynamics induced by the agent due to its unsupervised nature. Future work can study integrating factored structure (Daniel et al., 2026; Wang et al., 2026) to the temporally-extended latent actions to create disentanglement that can benefit downstream supervision. Finally, using the same low-level value function for all methods allowed us to isolate and reveal the capabilities temporally-extended world models enable compared to single-step world models and model-free approaches, but limits our framework in its current form from solving even longer horizon tasks. Future work can investigate incorporating the temporally-extended latent actions in learning high-level value functions.

AI USE STATEMENT

In this work, we used generative AI tools solely for coding assistance and to polish writing in some parts of the paper. AI-generated code was verified and tested for correctness by the the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our work advances methods for Goal-conditioned Reinforcement Learning and World Models. The experiments in this paper were conducted in simulated environments, with no involvement of human subjects or privacy concerns, and we do not foresee any ethical or societal risks from this work.

REPRODUCIBILITY STATEMENT

We are committed to enabling full reproducibility of our work. We have included comprehensive implementation and evaluation details throughout the paper and in the Appendix. We will release code to reproduce all of the experiments in this work to facilitate future research upon publication.

REFERENCES

Hongjoon Ahn, Heewoong Choi, Jisu Han, and Taesup Moon. Option-aware temporally abstracted value for offline goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 38:99833–99861, 2026.

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.

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

Akhil Bagaria, Jason K Senthil, and George Konidaris. Skill discovery for exploration and planning using deep skill graphs. In *International conference on machine learning*, pp. 521–531. PMLR, 2021.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International conference on machine learning*, pp. 1430–1440. PMLR, 2021.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *International Conference on Learning Representations*, volume 2024, pp. 15979–16002, 2024.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Rohan Chitnis, Yingchen Xu, Bobak Hashemi, Lucas Lehnert, Urun Dogan, Zheqing Zhu, and Olivier Delalleau. Iql-td-mpc: Implicit q-learning for hierarchical model predictive control. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9154–9160. IEEE, 2024.

Hojun Chung, Junseo Lee, and Songhwai Oh. Offline reinforcement learning with universal horizon models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=GFmJ5XcGWl.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lTaPtGiUUc.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: a model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 465–472, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Marc G Bellemare, Alessandro Lazaric, and Ahmed Touati. Compositional planning with jumpy world models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=cwztmjMQ77.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. In *International Conference on Machine Learning*, pp. 18744–18771. PMLR, 2025.

Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TjCDNssXKU.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

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.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

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.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

Dan Haramati, Carl Qi, Tal Daniel, Amy Zhang, Aviv Tamar, and George Konidaris. Hierarchical entity-centric reinforcement learning with factored subgoal diffusion. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TimC6hxVHj.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pp. 1094–8, 1993.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Martin Klissarov, Akhil Bagaria, Ziyan Luo, George Konidaris, Doina Precup, and Marlos C Machado. Discovering temporal structure: An overview of hierarchical reinforcement learning. *arXiv preprint arXiv:2506.14045*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *International Conference on Learning Representations*, 2019.

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022.

Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pp. 20035–20064. PMLR, 2023.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7BQkXXM8Fy.

Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420. IEEE, 2020.

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

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PmLR, 2021.

Kwanyoung Park, Seohong Park, Youngwoon Lee, and Sergey Levine. Scalable offline model-based RL with action chunks. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=WXGb9unEHo.

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.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *Advances in Neural Information Processing Systems*, 38:8350–8389, 2026b.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Rafael Rodriguez-Sanchez and George Konidaris. Learning abstract world models for value-preserving planning with options. In *Reinforcement Learning Conference*, 2024. URL https://openreview.net/forum?id=h9IvopsMFS.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations*, volume 2024, pp. 9379–9395, 2024.

Lucy Xiaoyang Shi, Joseph J Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=iVxy2eO601U.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S Sutton, Marlos C Machado, G Zacharias Holland, David Szepesvari, Finbarr Timbers, Brian Tanner, and Adam White. Reward-respecting subtasks for model-based reinforcement learning. *Artificial Intelligence*, 324:104001, 2023.

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

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023.

Zizhao Wang, Chang Shi, Jiaheng Hu, Kevin Rohling, Roberto Martín-Martín, Amy Zhang, and Peter Stone. Factored latent action world models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=raUHOQnZyv.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VYOe2eBQeh.

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.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *Forty-second International Conference on Machine Learning*, 2025.

John Zhou and Jonathan Kao. Flattening hierarchies with policy bootstrapping. *Advances in Neural Information Processing Systems*, 38:82975–83007, 2026.

APPENDIX

A METHOD IMPLEMENTATION DETAILS

In this section we provide comprehensive implementation details for our method, baselines and ablations in our experiments. Tables 1 and 7 compare methods with respect to various algorithmic aspects. We implement all methods in PyTorch based on official implementations and LeanRL: `https://github.com/meta-pytorch/LeanRL`, a more efficient version of CleanRL (Huang et al., 2022). All methods utilize the value functions of a goal-conditioned variant of IQL (Kostrikov et al., 2022). Low-level policies are extracted for all methods (except SAW) from IQL Q-functions with DDPG+BC (Fujimoto & Gu, 2021). All agent components are 5-layer MLPs with a hidden dimension of 512, GeLU activation (Hendrycks & Gimpel, 2016) and layer normalization (Ba et al., 2016) (except for the output layer). All subgoal-generating methods follow the same planning procedure: (i) when the goal-conditioned value is within a goal vicinity threshold τ_{gv}, we terminate planning and condition the low-level policy on the goal until the end of the episode and (ii) when the subgoal-conditioned value is within a subgoal reaching thresold τ_{sg}, the method replans. We report shared hyperparameters in Table 4.

Table 4: **Shared Hyperparameters**.

Hyperparameter	Value
Batch size	1024
Learning rate	0.0003
Optimizer	Adam (Kingma & Ba, 2017)
Gradient steps	1000000
Gradient clip norm	20
Discount factor γ	0.999 (`large`, `humanoidmaze`), 0.99 (other)
Target smoothing coefficient τ	0.005
IQL expectile	0.9 (manipulation, `antsoccer`), 0.7 (locomotion)
BC weighting term (DDPG+BC) α	0.01 (`pointmaze`), 0.4 (`humanoidmaze`), 0.2 (`antsoccer-arena`), 0.5 (`antsoccer-medium-navigate`), 3.0 (`play`), 0.1 (others)
{SAW, HIQL} AWR temperature β	3.0
Goal vicinity threshold τ_{gv}	100 (`humanoidmaze`), 10 (`cube-single`), 70 (`antsoccer-arena`), 50 (`antsoccer-medium-navigate`), 30 (others)
Subgoal reaching threshold	3 (`pointmaze`), 0 (`cube`), 1 (`scene`), 10 (others)
Policy Goal Sampling $(p^{\mathcal{D}}_{\text{cur}}, p^{\mathcal{D}}_{\text{traj}}, p^{\mathcal{D}}_{\text{geom}}, p^{\mathcal{D}}_{\text{rand}})$	$(0, 0.5, 0, 0.5)$ (`stitch`), $(0, 1, 0, 0)$ (others)
Value Goal Sampling $(p^{\mathcal{D}}_{\text{cur}}, p^{\mathcal{D}}_{\text{traj}}, p^{\mathcal{D}}_{\text{geom}}, p^{\mathcal{D}}_{\text{rand}})$	$(0.2, 0, 0.5, 0.3)$

Te-LAWM (ours): Temporally-extended Latent Action World Model. As detailed in Sec. 3.1, Te-LAWM learns a temporally-extended latent action space by jointly training a multi-step latent action model (LAM), consisting of multi-step inverse and forward dynamics models. We also learn a state-conditioned prior to generate these temporally-extended latent actions, which the forward dynamics model can map to corresponding multi-step future states. At test-time, Te-LAWM performs autoregressive planning up to a desired depth d. Temporally-extended latent actions sampled from the prior generate a set of initial subgoal candidates S_1. This process is repeated, conditioning the prior on the newly generated states to produce subsequent subgoals up to S_d. To evaluate the imagined trajectories, a pretrained goal-conditioned IQL value function V scores the future states. Depending on the environment, we select the highest-scoring state from either the final waypoint ($\arg\max_{\tilde{g}_i \in S_d} V(s_t, \tilde{g}_i)$, where s_t is the current state of the agent) or the first waypoint ($\arg\max_{\tilde{g}_i \in S_1} V(s_t, \tilde{g}_i)$) to serve as the target subgoal. Finally, employing receding horizon control for n steps, a pretrained IQL policy is conditioned on this selected subgoal to execute the corresponding low-level control actions. The hyperparameters per environment are given in Table 5.

Fixed Prior Te-LAWM: We ablate Te-LAWM by dropping the learned prior. Temporally-extended latent actions are sampled from the standard normal distribution $\mathcal{N}(0, I)$. All other aspects of the model are identical to Te-LAWM.

Single-step Te-LAWM: We ablate the planning depth by restricting it to $d = 1$. All other aspects of the model are identical to Te-LAWM except for the subgoal distance K, which is increased based on the task's planning configuration: $K = 50$ for `pointmaze-medium`, `antmaze-medium`, `antmaze-large-navigate-v0`, and `cube-triple-noisy-v0`, $K = 100$ for `pointmaze-large-stitch-v0`, `antmaze-large-stitch-v0`,

Table 5: **Te-LAWM Hyperparameters.**

Environment		LAM latent dim.	KL β	K	n	d	Subgoal state depth	# subgoal candidates
pointmaze	pointmaze-medium-navigate-v0	2	0.004	25	10	2	1	128
	pointmaze-large-navigate-v0	2	0.004	100	10	2	1	128
	pointmaze-medium-stitch-v0	2	0.004	25	10	2	1	128
	pointmaze-large-stitch-v0	2	0.004	100	10	2	1	128
antmaze	antmaze-medium-navigate-v0	8	0.004	25	10	2	1	128
	antmaze-large-navigate-v0	8	0.004	25	10	2	1	128
	antmaze-medium-stitch-v0	8	0.004	25	10	2	1	128
	antmaze-large-stitch-v0	8	0.004	50	10	2	1	128
humanoidmaze	humanoidmaze-medium-navigate-v0	21	0.04	100	10	2	2	128
	humanoidmaze-large-navigate-v0	21	0.04	100	10	2	2	128
	humanoidmaze-medium-stitch-v0	21	0.04	100	10	2	2	128
	humanoidmaze-large-stitch-v0	21	0.04	100	10	2	2	128
antsoccer	antsoccer-arena-navigate-v0	8	0.004	50	10	2	2	128
	antsoccer-arena-stitch-v0	8	0.004	50	10	2	2	128
	antsoccer-medium-navigate-v0	8	0.004	100	10	2	1	128
cube	cube-single-play-v0	5	0.004	10	10	2	2	1024
	cube-double-play-v0	5	0.004	50	50	2	2	1024
	cube-triple-play-v0	5	0.004	50	50	2	2	1024
	cube-single-noisy-v0	5	0.004	10	10	8	8	1024
	cube-double-noisy-v0	5	0.004	10	10	8	8	1024
	cube-triple-noisy-v0	5	0.004	10	10	8	8	1024
scene	scene-play-v0	5	0.004	50	50	2	2	128
	scene-noisy-v0	5	0.004	50	10	2	2	128

Table 6: **Single-step Te-LAWM subgoal distance K.**

K	Environments
50	pointmaze-medium, antmaze-medium, antmaze-large-navigate-v0, cube-triple-noisy-v0
100	pointmaze-large-stitch-v0, antmaze-large-stitch-v0, antsoccer-medium-navigate-v0, cube-double-play-v0, scene-play-v0
200	pointmaze-large-navigate-v0, humanoidmaze

antsoccer-medium-navigate-v0, cube-double-play-v0, and scene-play-v0, and $K = 200$ for pointmaze-large-navigate-v0 and humanoidmaze.

ACWM: Action Chunk World Model. To ablate Te-LAWM's temporal action abstraction, we train a multi-step forward dynamics model, $D_\theta(s_t, a_{t:t+K-1}) \rightarrow \hat{s}_{t+K}$, that conditions directly on raw action chunks rather than latent representations thereof. Alongside this, we train a diffusion policy via imitation learning (Chi et al., 2023) to predict these action chunks. At test-time, candidate action chunks are sampled from the diffusion policy with sampling temperaure 1.0, while the subsequent planning, scoring, and execution procedures remain identical to Te-LAWM. The diffusion policy is trained with 20 denoising steps and a cosine noise schedule.

LACWM: Latent Action Chunk World Model. To ablate the Te-LAWM temporal action abstraction learning procedure, we derive our latent action space directly from raw action chunks rather than action-free data. Specifically, we employ a state-conditioned VAE (Sohn et al., 2015) to encode these action chunks into latent representations. A state-conditioned prior is jointly learned alongside the CVAE, followed by training a multi-step forward dynamics model, D_θ, which conditions these newly learned latents. At test-time, candidate latent representations are sampled from the learned prior, while all subsequent planning, scoring, and execution procedures remain identical to Te-LAWM.

IQL: Implicit Q-Learning. This baseline corresponds to a goal-conditioned agent based on IQL (Kostrikov et al., 2022). We extract a policy from the learned Q-value function using Deep Deterministic Policy Gradient with a Behavior Cloning penalty term (DDPG+BC) (Fujimoto & Gu, 2021).We base our implementation on the official OGBench implementation at https://github.com/seohongpark/ogbench. For humanoidmaze-large environments we additionally employ 5-step returns in value learning.

TD-MPC: Temporal-Difference Model Predictive Control. This baseline corresponds to the offline RL variant of TD-MPC (Hansen et al., 2022; 2024), called IQL-TD-MPC, introduced in Chitnis et al. (2024). A world model is trained offline to predict the one-step future state s_{t+1} given state s_t and action a_t using reconstruction loss. It is implemented as a 4-layer MLP with a hidden dimension of 512. We reuse the trained IQL's value functions and policy. At test-time, actions are dynamically se-

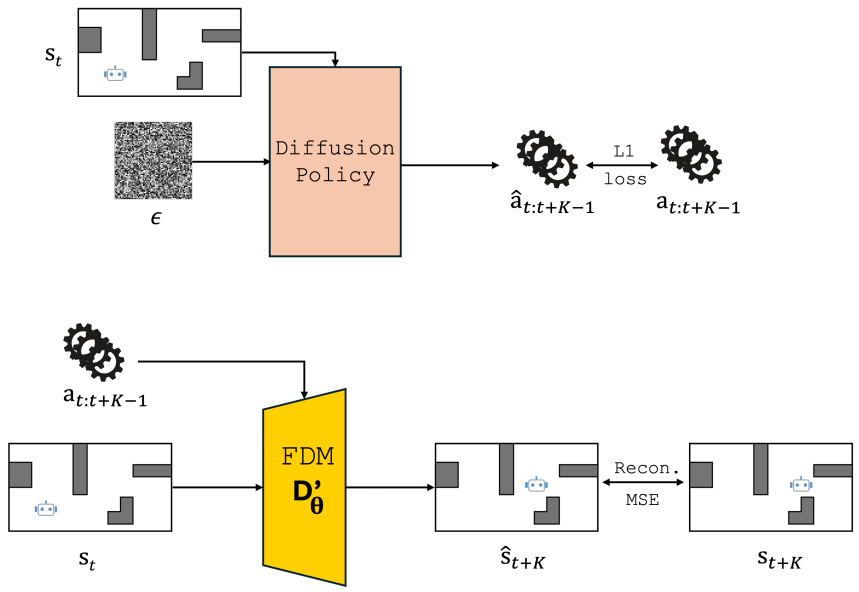


Figure 7: **Action Chunk World Model (ACWM).** The forward dynamics model conditions directly on raw action chunks $a_{t:t+K-1}$ to predict future states $\hat{s}_{t+K}$. An action-chunk diffusion policy trained via imitation learning serves as a prior for sampling candidate action chunks for subgoal planning.

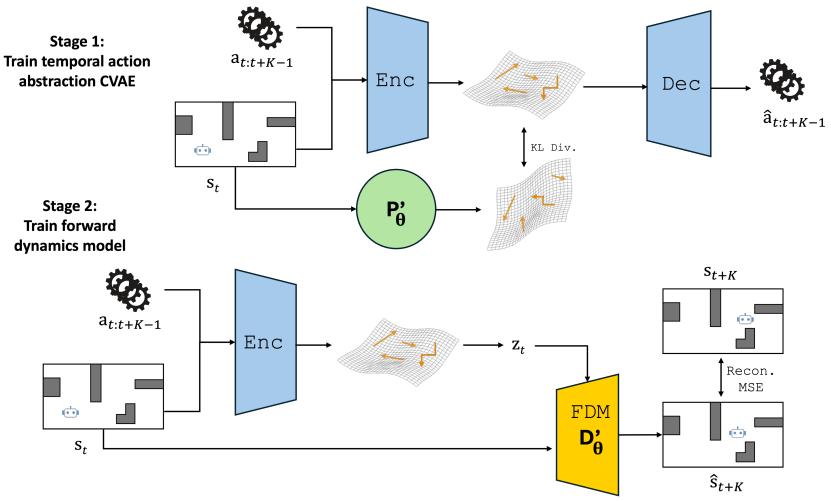


Figure 8: **Latent Action Chunk World Model (LACWM).** LACWM jointly learns a state-conditioned VAE and prior P'_θ to encode raw action chunks. A forward dynamics model D'_θ is trained separately to predict multi-step-next-states given latent representations of action chunks. Once trained, P'_θ and D'_θ are used for planning in the same way as the Te-LAWM.

lected using the Model Predictive Path Integral (MPPI) (Williams et al., 2015) algorithm to optimize 512 candidate action sequences over a fixed-length planning horizon. The optimization process runs for 6 iterations and in each iteration, the top 64 scoring trajectories are used to update the sampling distribution for the next pass. The world model simulates these trajectories to the end of the horizon, where the value function scores the resulting final states, guiding the policy to execute the first 3 steps of the highest-yielding sequence before replanning. We base our implementation on the official TD-MPC2 (Hansen et al., 2024) implementation at `https://github.com/seohongpark/ogbench`. `https://github.com/nicklashansen/tdmpc2`.

Table 7: **Comparison of Te-LAWM variants and ablations.**

Attribute	ACWM	LACWM	Fixed Prior	Single-Step	Te-LAWM
Action abstraction	✗	✓	✓	✓	✓
Learned from action-free data	✗	✗	✓	✓	✓
State-conditioned prior	✓	✓	✗	✓	✓
Autoregressive subgoal prediction	✓	✓	✓	✗	✓

SAW: Subgoal Advantage-Weighted Policy Bootstrapping. Based on the method proposed in Zhou & Kao (2026). SAW trains two policies — a waypoint policy and a low-level goal-reaching policy — and regularizes the low-level policy to match the waypoint policy whenever subgoals yield a high advantage. Through this policy bootstrapping, the low-level policy absorbs the high-level reasoning, allowing it to act entirely on its own at inference time without needing to query the explicit hierarchical structure. We reuse the trained IQL's value functions and extract both policies using AWR and hence do not employ goal representation. We note that the low-level policy is extracted using a $Q + V$ version of AWR for parity with the low-level policies of all the other methods we experiment with in this work. We tune two key hyperparameters across environments. For the scaling factor β of KL-divergence loss between the low-level policy and the waypoint policy, we use 0.3 for `cube-single`, 1.0 for `cube-{double, triple}` and 3.0 for the other environments. For the subgoal distance K, we use 100 for the `humanoidmaze` environments, 10 for the manipulation environments, and 25 for the rest. We base our implementation on the author-provided implementation at `https://github.com/johnlyzhou/saw`.

HIQL: Hierarchical Implicit Q-Learning. This baseline corresponds to an agent based on HIQL (Park et al., 2023). We adapt the official implementation to make it similar to the other methods for a fair comparison while keeping the core attributes unchanged. Specifically, we reuse the trained IQL's value functions and employ its policy as HIQL's low-level policy. Hence, the low-level policy is trained on the goal distribution rather than the subgoal distribution and goal representation networks are dropped. The high-level policy is trained with AWR using the IQL value function. For the subgoal distance K, we use 100 for the `humanoidmaze` environments, 10 for the manipulation environments, and 25 for the rest. We base our implementation on the official OGBench implementation at `https://github.com/seohongpark/ogbench`.

SD: Subgoal Diffuser. This approach represents Diffusers (Janner et al., 2022; Lu et al., 2025) as high-level subgoal planners (Li et al., 2023; Chen et al., 2024) in offline sequential decision-making. We construct a hierarchical baseline agent similar to Te-LAWM, replacing the latent action model with a goal-conditioned diffusion model (Sohl-Dickstein et al., 2015; Ho et al., 2020) for subgoal generation. Subgoal selection and low-level action execution is identical to Te-LAWM. SD is trained using a cosine noise schedule (Nichol & Dhariwal, 2021) over 20 diffusion steps to reconstruct s_{t+K} given the current state s_t and goal g. At test time, candidate subgoals are generated via 10 DDIM (Song et al., 2021) denoising steps. SD inherits all shared environment-specific hyperparameters from Te-LAWM (Table 5).

B Additional Results

B.1 Main Results

Our evaluation protocol tests performance on 20 episodes per 5 OGBench predefined initial state and goal pairs which are slightly randomized in each episode.

Fig. 9 presents an additional example comparing method's generated subgoals which highlights Te-LAWM's ability to perform subgoal stitching.

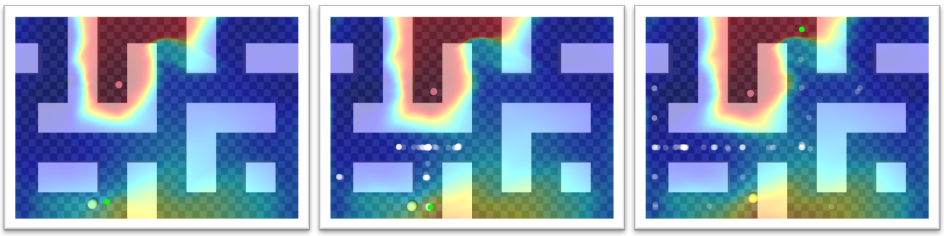
Fig. 10 presents planning failure modes in baselines compared to the Te-LAWM.

Algorithm 1 Hierarchical Subgoal Planning with the Te-LAWM

Input: Initial state s_t, goal g, Te-LAWM D_θ, latent action policy P_θ, low-level policy π_ϕ, value function V_ϕ, number of subgoal candidates N, planning horizon T, subgoal timeout T_{sg}, subgoal threshold ϵ_{sg}, goal threshold ϵ_g

Output: Actions a_t executed toward subgoal $\tilde{g}$

while $V_\phi(s_t, g) < \epsilon_g$ **do**
  # Generate N subgoal candidates in parallel
  $\mathcal{S}_0 \leftarrow \{s_t\}_{i=1}^N$
  for $k = 0, \dots, T-1$ **do**
    $\mathbf{z}_k \sim P_\theta(\cdot \mid \mathcal{S}_k)$
    $\mathcal{S}_{k+1} \leftarrow D_\theta(\mathcal{S}_k, \mathbf{z}_k)$
  end for
  $\{\tilde{g}^i\}_{i=1}^N \leftarrow \mathcal{S}_T$
  # Select subgoal with highest value
  $\tilde{g} \leftarrow \underset{\tilde{g}^i \in \{\tilde{g}^1, \dots, \tilde{g}^N\}}{\arg\max} \; V_\phi(\tilde{g}^i, g)$
  # Execute low-level policy toward subgoal
  $j \leftarrow 0$
  while $V_\phi(s_t, \tilde{g}) < \epsilon_{sg}$ **and** $j < T_{sg}$ **do**
    $a_t \leftarrow \pi_\phi(s_t, \tilde{g})$
    Execute a_t and observe s_{t+1}
    $t \leftarrow t + 1$
    $j \leftarrow j + 1$
  end while
end while
Execute $\pi_\phi(s_t, g)$ until end of episode



HIQL SD Te-LAWM

Figure 9: **Subgoal Generation Comparison**; `pointmaze-large-stitch-v0`. The images contrast the subgoal candidates produced by each hierarchical method. The heatmap overlaid on the maze represents a learned goal-conditioned value landscape for the goal point in red. HIQL produces a single subgoal. SD and Te-LAWM capture a distribution over subgoals (white points) and select the one with the highest value (green point). SD is limited by the in-trajectory future states present in the `stitch` dataset while Te-LAWM stitches dataset trajectories to produce more distant subgoals that allow the agent to progress towards the goal.

B.2 Visual Navigation

We demonstrate that our framework can be applied to high-dimensional observations by training the Te-LAWM on fixed latent representations of images. Results on `visual-antmaze` tasks compared to baselines are presented in Tab. 8. Te-LAWM outperforms or matches all baselines.

Implementation Details: IQL is trained directly from pixels following the OGBench implementation. All subgoal generation modules operate on a frozen 32-dimensional latent representation learned with a convolutional autoencoder trained per task on the offline dataset. It is trained for $200K$ steps with batch size 256 under three losses: (1) pixel MSE reconstruction; (2) a reward head on the latent with weight 1.0, whose gradients force goal-relevant information into the representation; (3) an L_2 penalty on the latent with weight 0.001. At test-time, subgoals are generated in the latent space and decoded before being evaluated and fed to the IQL policy. All methods use the

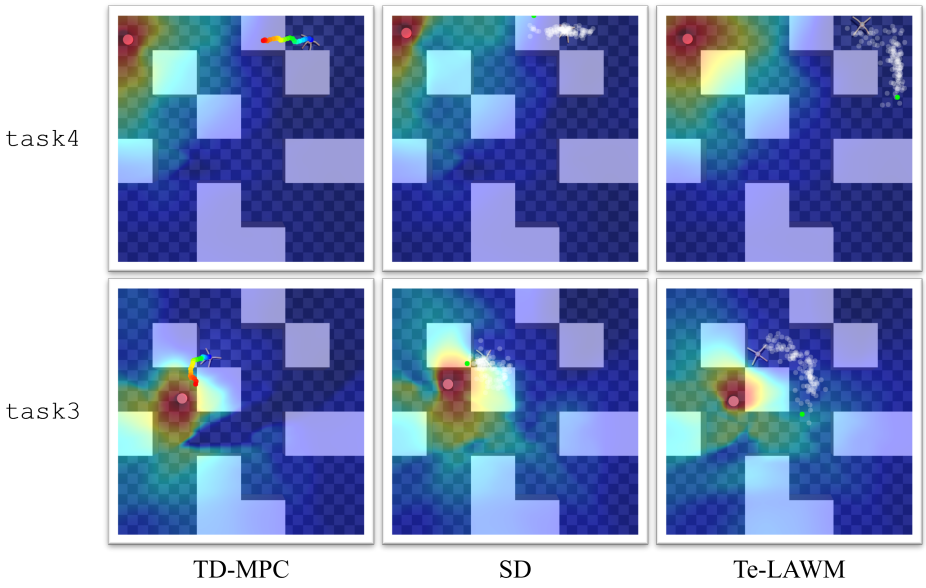


Figure 10: **Baseline Failure Modes**; `antmaze-medium-navigate-v0`. The heatmap overlaid on the maze represents a learned goal-conditioned value landscape for the goal point in red. TD-MPC plans with MPPI which can steer from the support of the dataset and this case, select plans that traverse walls. SD—a goal-conditioned subgoal generator—occasionally produces subgoals biased towards the goal that disregard the wall positions. Te-LAWM largely avoids these failure modes since it plans within the dataset distribution using the learned latent action prior and is not goal-conditioned.

Table 8: **Visual Antmaze Results.** Success rates averaged across 3 seeds. See App. A for method implementation details. Best results up to a standard deviation are highlighted in **bold**.

Task	IQL	HIQL	SD	Te-LAWM
`visual-antmaze-medium-navigate-v0`	**92** ± 3	86 ± 5	70 ± 5	**89** ± 3
`visual-antmaze-medium-stitch-v0`	73 ± 2	**91** ± 4	65 ± 8	**89** ± 2
`visual-antmaze-large-navigate-v0`	66 ± 6	75 ± 10	38 ± 3	**87** ± 5

same per-task hyperparameters as the corresponding state tasks, with the exception that Te-LAWM uses $K = 50$ and $d = 4$ on `visual-antmaze-large-navigate`.

B.3 Latent Action Space Analysis

We provide event definitions for our `cube-single-play` latent action space analysis in Sec. 4.3.

We define four manipulation events on K-step transitions (s_t, s_{t+K}). Each event is a rule on the gripper-cube contact, the gripper opening, and the gripper and cube positions:

Reach: the open gripper moves toward the cube without touching it, and the cube stays still.
Grasp: the gripper starts away from the cube and ends holding it, with the cube on the table.
Lift: the gripper holds the cube throughout and raises it off the table.
Release: the gripper starts holding a cube on the table and ends open and away from it.

The events are mutually exclusive. We score a transfer as successful when the transition decoded from the transferred latent action satisfies the same rule.

Fig. 11 presents the statistics of latent action transfer and Fig. 12 presents rendered examples.

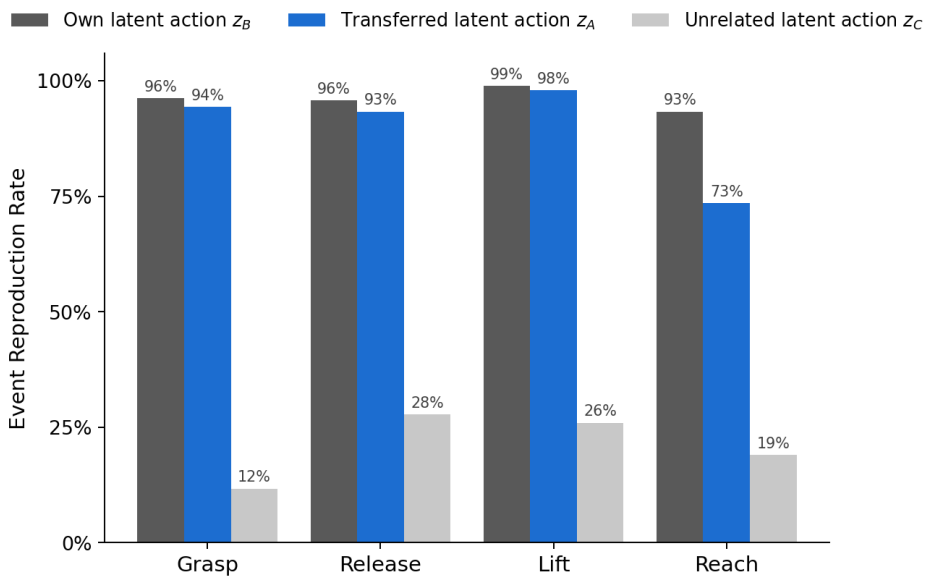


Figure 11: **Temporally-extended Latent Action Transfer**, `cube-single-noisy`. State s_t^B is decoded with three latent actions: B's own z_B, z_A transferred from another transition carrying the same event label, and an unrelated z_C. Bars give the event reproduction rate—the fraction of the outcomes that result in the same event. Results are measured over 30,000 triplets.

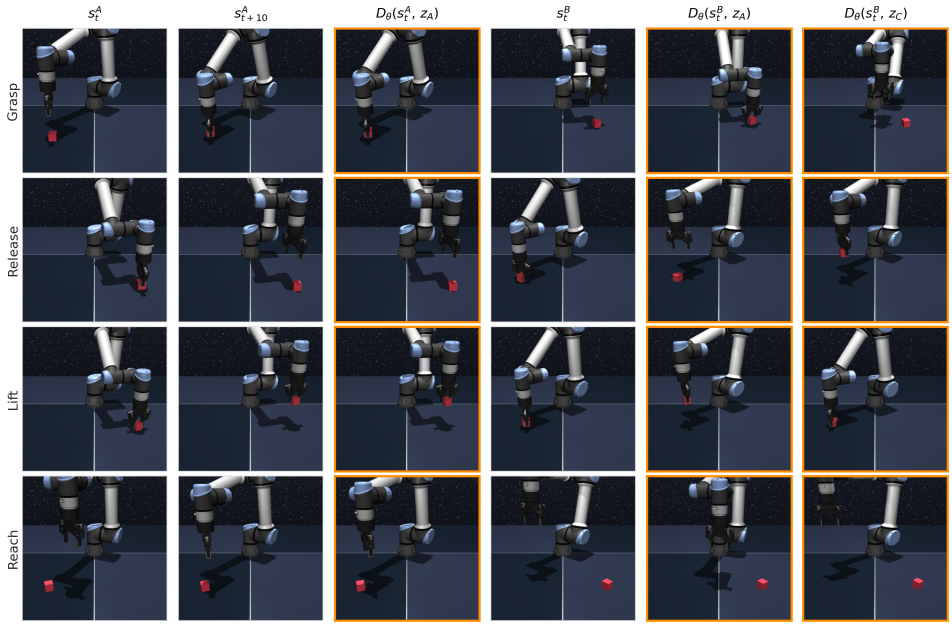


Figure 12: **Temporally-extended Latent Action Transfer**; `cube-single-noisy`. Each row applies the latent action z_A inferred from state transition $A = (s_t^A, s_{t+10}^A)$ to both s_t^A and s_t^B. z_C is inferred from a transition carrying a different label and confirms that other outcomes are available at s_t^B. Description of frames from left to right: s_t^A; s_{t+10}^A; reconstruction of s_{t+10}^A; s_t^B; result of applying z_A in s_t^B; result of applying z_C in s_t^B. Orange borders mark rendered decoder predictions, the remaining frames are rendered dataset states.