COMPOSITIONAL PLANNING WITH JUMPY WORLD MODELS

Jesse Farebrother[2,3,*] **Matteo Pirotta**[1] **Andrea Tirinzoni**[1] **Marc G. Bellemare**[2,3,†]
Alessandro Lazaric[1] **Ahmed Touati**[1]
[1]FAIR at Meta [2]Mila – Québec AI Institute [3]McGill University

ABSTRACT

The ability to plan with temporal abstractions is central to intelligent decision-making. Rather than reasoning over primitive actions, we study agents that compose pre-trained policies as temporally extended actions, enabling solutions to complex tasks that no constituent alone can solve. Such compositional planning remains elusive as compounding errors in long-horizon predictions make it challenging to estimate the visitation distribution induced by sequencing policies. Motivated by the *geometric policy composition* framework introduced in Thakoor et al. (2022), we address these challenges by learning predictive models of multi-step dynamics — so-called *jumpy world models* — that capture state occupancies induced by pre-trained policies across multiple timescales in an off-policy manner. Building on Temporal Difference Flows (Farebrother et al., 2025), we enhance these models with a novel consistency objective that aligns predictions across timescales, improving long-horizon predictive accuracy. We further demonstrate how to combine these generative predictions to estimate the value of executing arbitrary sequences of policies over varying timescales. Empirically, we find that compositional planning with jumpy world models significantly improves zero-shot performance across a wide range of base policies on challenging manipulation and navigation tasks, yielding, on average, a 200% relative improvement over planning with primitive actions on long-horizon tasks.

1 INTRODUCTION

In recent years, the success of large-scale foundation models in domains such as computer vision (e.g., Radford et al., 2021; Ravi et al., 2025; Assran et al., 2025) and natural language processing (e.g., Meta, 2024; OpenAI, 2024; Google DeepMind, 2025) has inspired a similar shift in Reinforcement Learning (RL). Foundation policies pre-trained on diverse unlabeled data or via unsupervised objectives can now generalize to a wide range of downstream tasks, without additional training or explicit planning. This has led to remarkable progress in areas like humanoid control (e.g., Peng et al., 2021; 2022; Tessler et al., 2023; 2024; Tirinzoni et al., 2025; Alegre et al., 2025) and real-world robotics (e.g., Brohan et al., 2023b;a; Luo et al., 2024; Ghosh et al., 2024; Black et al., 2024; 2025; Physical Intelligence et al., 2025; NVIDIA, 2025; Li et al., 2026).

Despite these advances, a key limitation persists: while foundation policies can handle many tasks out of the box, they often fall short when faced with complex, long-horizon problems that require reasoning over extended sequences of decisions. In such cases, the planning horizon of a single policy is insufficient, and agents must compose policies to achieve their goals (Schmidhuber, 1991; Singh, 1992; Dayan & Hinton, 1992; Kaelbling, 1993b;a; Parr & Russell, 1997; Dietterich, 1998; Sutton et al., 1999; Precup, 2000).

Hierarchical RL (Barto & Mahadevan, 2003; Klissarov et al., 2025), including the options framework (Sutton et al., 1999; Precup, 2000), aims to achieve compositionality by training task-specific high-level policies to leverage manual (e.g., Nachum et al., 2018; Barreto et al., 2019; Carvalho et al., 2023; Park et al., 2023; 2025b) or automatic (e.g., Bacon et al., 2017; Machado et al., 2017; 2018; 2023; Bagaria et al., 2021; Sutton et al., 2023) task decompositions. In this paper, we take a fundamentally different approach: instead of learning task-specific hierarchies, we develop a framework

[*]Work done at Meta [†]CIFAR AI Chair

for direct compositional planning over parameterized policies, requiring *no task-specific* training. By learning "jumpy" multi-step dynamics models – also known as "jumpy world models" (Murphy, 2024) – we enable flexible composition of existing policies at planning time, transforming how agents tackle novel tasks through intelligent recombination of existing behavior.

To operationalize this idea, we propose learning a *policy* and *horizon* conditioned *jumpy world model* that captures the distribution of future states for all parameterized policies over a continuum of geometrically decaying time horizons (Janner et al., 2020; Thakoor et al., 2022). To make this possible, we first generalize the recent Temporal Difference Flow (Farebrother et al., 2025) framework using a novel consistency objective that enforces coherence between predictions at different timescales, consistently improving long-horizon predictions. Additionally, we develop a novel estimator of the value of executing an arbitrary sequence of policies, each with its own variable timescale. With these two pieces in place, we demonstrate how to plan over a wide range of parameterized policies, allowing us to flexibly compose behaviors to solve complex, long-horizon tasks without the need for further environment interaction or fine-tuning. Empirically, across multiple classes of base policies evaluated on a suite of OGBench navigation and manipulation tasks (Park et al., 2025a), behavior-level planning consistently improves over zero-shot performance, often by a large margin. Finally, our approach outperforms state-of-the-art hierarchical baselines as well as alternative planning methods; in particular, planning with jumpy world models achieves a 200% relative improvement over action-level planning with a one-step world model on long-horizon tasks. These results demonstrate that planning with jumpy world models offers a powerful complement that is particularly effective for long-horizon decision-making.

2 PRELIMINARIES

We model the environment as a reward-free discounted Markov Decision Process (MDP) defined as the 4-tuple $\mathcal{M} = (\mathsf{S}, \mathsf{A}, P, \gamma)$. Here, S and A represent the state and action spaces, respectively; $P : \mathsf{S} \times \mathsf{A} \to \mathscr{P}(\mathsf{S})$ characterizes the distribution over next states; and $\gamma \in [0, 1)$ is the discount factor. At each step k, the agent follows a policy $\pi : \mathsf{S} \to \mathscr{P}(\mathsf{A})$, generating a trajectory of state-action pairs $(S_k, A_k)_{k\geq 0}$ where $A_k \sim \pi(\cdot \,|\, S_k)$ and $S_k \sim P(\cdot \,|\, S_{k-1}, A_{k-1})$. When unambiguous, we use S' as the immediate next state of S, and S^+ for a successor state at some future step $k > 0$. We use $\Pr(\cdot \mid S_0 = s, A_0 = a, \pi)$ and $\mathbb{E}[\cdot \mid S_0 = s, A_0 = a, \pi]$ to denote the probability and expectation over sequences induced by starting from (s, a) and following π thereafter.

Successor Measure For a policy π and an initial state-action pair (s, a), *the (normalized) successor measure* (Dayan, 1993; Blier et al., 2021), denoted by $m_\gamma^\pi(\cdot \mid s, a)$, is a probability measure over the state space S. For any subset $\mathsf{X} \subseteq \mathsf{S}$, it represents the cumulative discounted probability of visiting a state in X, with each visit discounted geometrically by its time of arrival. Formally, this is defined as:

$$m_\gamma^\pi(\mathsf{X} \mid s, a) = (1-\gamma) \sum_{k=0}^{\infty} \gamma^k \Pr(S_{k+1} \in \mathsf{X} \mid S_0 = s,\, A_0 = a,\, \pi)\,.$$

The normalization factor $1 - \gamma$ ensures m_γ^π is a probability distribution. This admits an intuitive interpretation: rather than viewing γ as a discount factor, one may equivalently consider an *auxiliary* process with a geometrically distributed lifetime – halting at each step with probability $1 - \gamma$ (Derman, 1970). Under this view, $m_\gamma^\pi(\mathsf{X} \mid s, a)$ equivalently characterizes the probability that the state visited at the halting time lies in X. This yields a convenient reparameterization of the action-value function as:

$$Q_\gamma^\pi(s, a) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r(S_{k+1}) \mid S_0 = s, A_0 = a, \pi\Big] \equiv (1-\gamma)^{-1} \mathbb{E}_{S^+ \sim m_\gamma^\pi(\cdot|s,a)}\left[r(S^+)\right], \quad (1)$$

expressing value as the expected reward at the geometrically distributed halting time scaled by the average lifetime $(1-\gamma)^{-1}$. Note that this equivalence is purely mathematical[1]; the underlying MDP remains unchanged.

[1]This preserves expected cumulants and occupancy measures, but not trajectory-level statistics (Bellemare et al., 2023)

Geometric Horizon Model A Geometric Horizon Model (GHM; Janner et al., 2020; Thakoor et al., 2022) instantiates a jumpy world model as a *generative model* of the successor measure. It can be learned off-policy via temporal-difference learning, exploiting the fact that m_γ^π is a fixed point of the Bellman equation:

$$m_\gamma^\pi(\cdot \mid s, a) = (1-\gamma)P(\cdot \mid s, a) + \gamma \mathbb{E}_{S' \sim P(\cdot|s,a),\, A' \sim \pi(\cdot|S')} \left[m_\gamma^\pi(\cdot \mid S', A') \right]. \tag{2}$$

Due to the Bellman equation's reliance on bootstrapping, the choice of generative model is critical for maintaining stability. Farebrother et al. (2025) demonstrate that prior approaches suffer from systemic bias at long horizons due to these bootstrapped predictions. To address this, Farebrother et al. (2025) propose the use of flow-matching techniques (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023), which construct probability paths that evolve smoothly from a source distribution to the desired target distribution. By designing these paths to exploit structure in the temporal difference target distribution, they show that bootstrapping bias can be controlled, enabling more accurate long-horizon predictions.

In this framework, the GHM models a d-dimensional continuous state space[2] as an ordinary differential equation (ODE) parameterized by a time-dependent vector field $v_t : \mathbb{R}^d \times \mathsf{S} \times \mathsf{A} \to \mathbb{R}^d$. Sampling from the GHM begins by drawing initial noise $X_0 \in \mathbb{R}^d$ from a prior distribution $p_0 \in \mathscr{P}(\mathbb{R}^d)$ and subsequently following the flow $\psi_t : \mathbb{R}^d \times \mathsf{S} \times \mathsf{A} \to \mathbb{R}^d$, defined by the following Initial Value Problem (IVP) for $t \in [0, 1]$:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}\psi_t(x \mid s, a) = v_t\big(\psi_t(x \mid s, a) \mid s, a\big),\ \psi_0(x \mid s, a) = x \\ &\iff \psi_t(x \mid s, a) = x + \int_0^t v_\tau\big(\psi_\tau(x|s, a) \mid s, a\big)\, \mathrm{d}\tau\,. \end{aligned}$$

We can solve this IVP using standard numerical integration techniques (Butcher, 2016). In doing so, we obtain an ODE-induced probability path defined as the pushforward $p_t := \psi_t(\cdot \mid S, A)_\sharp p_0(\cdot)$, i.e., the distribution of $\psi_t(X_0 \mid S, A)$ where $X_0 \sim p_0(\cdot)$. To ensure that p_1 coincides with the successor measure m_γ^π, Farebrother et al. (2025) propose to learn the parameterized vector field $v_t(\cdots; \theta)$ by minimizing the TD-FLOW loss:

$$\begin{aligned} \ell_{\text{TD-FLOW}}(\theta; \bar{\theta}) = (1-\gamma)\, &\mathbb{E}_{\substack{t \sim \mathcal{U}([0,1]),\, (S,A,S') \sim \mathcal{D} \\ X_0 \sim p_0(\cdot),\, X_t = (1-t)X_0 + tS'}} \left[\left\| v_t(X_t \mid S, A; \theta) - (S' - X_0) \right\|^2 \right] \\ &+ \gamma\, \mathbb{E}_{\substack{t \sim \mathcal{U}([0,1]),\, (S,A,S') \sim \mathcal{D},\, A' \sim \pi(\cdot|S') \\ X_t \sim \psi_t(\cdot|S', A'; \bar{\theta})_\sharp p_0(\cdot)}} \left[\left\| v_t(X_t \mid S, A; \theta) - v_t(X_t \mid S', A'; \bar{\theta}) \right\|^2 \right], \end{aligned} \tag{3}$$

over transitions sampled from the dataset $\mathcal{D}$ with non-trainable target parameters $\bar{\theta}$ (Mnih et al., 2015). Recall that the Bellman equation (2) defines the successor measure as a mixture distribution with weights $1 - \gamma$ and γ. The TD-FLOW objective reflects this: the first term is a conditional flow-matching loss (Lipman et al., 2023) targeting the one-step transition kernel $P(\cdot \mid S, A)$, while the second is a marginal flow-matching term targeting the bootstrapped successor measure $m_\gamma^\pi(\cdot \mid S', A')$. Farebrother et al. (2025) show that jointly optimizing these components with the mixture weighting recovers the successor measure at convergence.

3 Planning via Geometric Policy Composition

In the sequel, we consider an agent equipped with a repertoire of pretrained policies $\{\pi_z\}_{z \in \mathsf{Z}}$ indexed by z (e.g., a state for goal-conditioned policies or, more generally, a latent variable parameterizing diverse behaviors). Our objective is to learn a predictive model of the policies' behaviors that enables planning for arbitrary downstream tasks, without requiring further online interaction with the environment or additional fine-tuning.

To this end, we first formalize how GHM predictions compose to evaluate plans that stochastically switch among a subset of policies, showing how the successor measures of constituent policies

[2]While we assume a continuous state space $\mathsf{S} \subseteq \mathbb{R}^d$ for ease of exposition, this flow-matching approach readily extends to non-Euclidean and discrete spaces (e.g., Huang et al., 2022; Chen & Lipman, 2024; Gat et al., 2024; Kapusniak et al., 2024).

combine to yield the successor measure of the composite policy. We then address the challenge of learning accurate GHMs across timescales by introducing a consistency objective that enforces coherence across horizons, improving long-horizon predictions. Finally, with these tools we detail our complete compositional planning procedure.

3.1 Evaluating Geometric Switching Policies

A natural way to chain policies is through *geometric switching*: for each policy π_{z_i} in a sequence, execution continues with probability $1 - \alpha_i \in [0, 1]$, or switches to the subsequent policy $\pi_{z_{i+1}}$ with probability α_i. Such a policy can be written as: $\nu := \pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}$. By definition, the final policy π_{z_n} is absorbing, meaning the agent commits to it for the remainder of the episode, so its switching probability is $\alpha_n = 0$. These non-Markovian policies are called Geometric Switching Policies (GSPs; Thakoor et al., 2022). The term "geometric" captures that each policy π_{z_i} is followed for a geometrically distributed duration $T_i \sim \mathrm{Geom}(\alpha_i)$.

To analyze these policies, we must understand how this switching mechanism interacts with the MDP's *global* discount factor, γ. Recall that γ can be interpreted as the probability the episode continues to the next step, while $1 - \gamma$ gives the probability of halting. When following policy π_{z_k} within a GSP, there are two reasons it might stop executing that policy: (1) the episode halts, with probability $1 - \gamma$; or (2) the policy switches to $\pi_{z_{k+1}}$, with probability α_k. Thus, continuing to follow π_{z_k} for one more step requires that neither event occur, which happens with probability $\beta_k := \gamma\,(1 - \alpha_k)$. This quantity acts as an effective discount factor for the duration spent executing π_{z_k}. Note that $\beta_n = \gamma$, since the final policy is absorbing.

We can now characterize the successor measure of a GSP. Note that the agent might reach a successor state s^+ through multiple paths: arriving while still executing π_{z_1}, or switching to π_{z_2} and reaching s^+ from there, and so on. Each path contributes to the overall successor measure and must be weighted appropriately.

Definition 1. *Let $\nu := \pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}$ be a geometric switching policy with global discount factor $\gamma \in (0, 1)$ and effective discount factors $\beta_k := \gamma(1 - \alpha_k)$ for $k \in [\![n]\!]$. The weight of the k-th policy is*

$$w_k \;:=\; \frac{1-\gamma}{1-\beta_k} \prod_{i=1}^{k-1} \frac{\gamma - \beta_i}{1 - \beta_i},$$

where an empty product equals 1 (hence $w_1 = \frac{1-\gamma}{1-\beta_1}$).

These weights capture the relative contribution of each policy phase to the successor measure of the GSP. Intuitively, w_k reflects the probability that the agent (i) survives the first $k - 1$ policy phases without the episode halting, and (ii) reaches states under policy π_{z_k} rather than having already switched to a later policy. With these weights, we now define the successor measure of the composite GSP in the following result.

Theorem 1. *Let $\nu := \pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}$ be a geometric switching policy with global discount factor $\gamma \in (0, 1)$, effective discount factors $\{\beta_k\}_{k=1}^n$, and weights $\{w_k\}_{k=1}^n$ from Definition 1. For any state-action pair (s, a), the successor measure of ν decomposes as:*

$$m_\gamma^\nu(\mathrm{d}s^+ \mid s, a) = \sum_{k=1}^{n} w_k \int_{\substack{s_1,\ldots,s_{k-1} \\ a_1,\ldots,a_{k-1}}} m_{\beta_1}^{\pi_{z_1}}(\mathrm{d}s_1 \mid s, a)\pi_{z_2}(\mathrm{d}a_1 \mid s_1) \cdots m_{\beta_k}^{\pi_{z_k}}(\mathrm{d}s^+ \mid s_{k-1}, a_{k-1})\,.$$

This theorem decomposes the successor measure of a GSP as a mixture distribution with n components. The k-th component, weighted by w_k, captures the state distribution under policy π_{z_k}, having passed through all intermediate states visited under $\pi_{z_1}, \ldots, \pi_{z_{k-1}}$[3]. Practically, such a decomposition yields a natural strategy for estimating action-value function similar to (1): sample a state from each component, evaluate its reward, and form a weighted sum using the weights w_k. Rather

[3]Each component can be seen as an application of the Chapman–Kolmogorov equation (Chapman, 1928; Kolmogoroff, 1931), retaining kernel composition via marginalization but replacing single-step dynamics $P(\cdot \mid s, a)$ with jumpy dynamics via $m_\beta^\pi(\cdot \mid s, a)$.

than sampling from each component independently, we can leverage their overlapping sequential structure to draw samples much more efficiently. We achieve this through composition: starting from (s, a), we sample $S_1^+ \sim m_{\beta_1}^{\pi_{z_1}}(\cdot \mid s, a)$, then use S_1^+ to sample $S_2^+ \sim m_{\beta_2}^{\pi_{z_2}}(\cdot \mid S_1^+, A_1^+)$ where $A_1^+ \sim \pi_{z_2}(\cdot \mid S_1^+)$, and so on. As the following lemma formalizes, the weighted sum $\sum_k w_k\, r(S_k^+)$ yields an unbiased estimator of Q_γ^ν.

Lemma 1. *Let $\nu := \pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}$ be a geometric switching policy with global discount factor $\gamma \in (0, 1)$, effective discount factors $\{\beta_k\}_{k=1}^n$, and weights $\{w_k\}_{k=1}^n$ from Definition 1. For any reward function $r : \mathsf{S} \to \mathbb{R}$ and state-action pair (s, a), set $(S_0^+, A_0^+) = (s, a)$ and for $k = 1, \ldots, n$ sample $S_k^+ \sim m_{\beta_k}^{\pi_{z_k}}(\cdot \mid S_{k-1}^+, A_{k-1}^+)$ and $A_k^+ \sim \pi_{z_{k+1}}(\cdot \mid S_k^+)$. Then the single-sample monte-carlo estimator*

$$\widehat{Q}_\gamma^\nu := (1-\gamma)^{-1} \sum_{k=1}^{n} w_k\, r(S_k^+)\,,$$

is an unbiased estimate of $Q_\gamma^\nu(s, a)$, i.e., $\mathbb{E}\big[\,\widehat{Q}_\gamma^\nu\,\big] = Q_\gamma^\nu(s, a)$ where the expectation is over the joint distribution of $(S_1^+, A_1^+, \ldots, S_n^+)$ induced by the sampling procedure above.

This lemma generalizes several previous results: it extends Thakoor et al. (2022, Theorem 3.2), which assumed fixed switching probabilities, i.e., $\alpha_k = \alpha, \forall\, k \in [\![n-1]\!]$, and further generalizes results in Janner et al. (2020), that additionally impose a fixed policy throughout, i.e., $\pi_{z_k} = \pi, \forall\, k \in [\![n]\!]$. By allowing both policies and switching probabilities to vary, this result enables the evaluation of a wider class of switching policies and brings us closer to the options framework (Sutton, 1995; Sutton et al., 1999; Precup, 2000).

3.2 Learning Geometric Horizon Models Across Multiple Timescales

A core requirement of our planning framework is the ability to predict the behavior of many policies over multiple timescales, enabling us to calculate the expected return of candidate geometric switching policies. A natural extension of the TD-FLOW objective (3) conditions the vector field v on both the policy encoding z and discount factor γ, yielding a single unified model across policies and horizons. However, generalizing across many horizons is challenging: variance increases with horizon length, reducing per-horizon accuracy and destabilizing training (Petrik & Scherrer, 2008).

To address this challenge, we propose a generalization of TD-FLOW that enforces consistency across horizons. Rather than learning each horizon independently, we exploit a Bellman-like relationship between the successor measure at two discount factors $\beta \leq \gamma$ to bootstrap longer-horizon predictions from shorter-horizon ones:

$$\begin{aligned} m_\gamma^\pi(\cdot \mid s, a) \;=\; & (1-\gamma)\, P(\cdot \mid s, a) + \gamma \frac{1-\gamma}{1-\beta}\, \mathbb{E}_{S' \sim P(\cdot|s,a), A' \sim \pi(\cdot|S')} \left[m_\beta^\pi(\cdot \mid S', A') \right] \\ & + \gamma \frac{\gamma - \beta}{1-\beta}\, \mathbb{E}_{\substack{S' \sim P(\cdot|s,a), A' \sim \pi(\cdot|S') \\ S^+ \sim m_\beta^\pi(\cdot|S',A'), A^+ \sim \pi(\cdot|S^+)}} \left[m_\gamma^\pi(\cdot \mid S^+, A^+) \right]. \end{aligned} \tag{4}$$

This follows from Theorem 1 by considering the switching policy $\nu := \pi \xrightarrow{\alpha_1 = 1} \pi \xrightarrow{\alpha_2 = 1-\beta/\gamma} \pi$ and noting that $m_{\gamma(1-\alpha_1)}^\pi = m_0^\pi = P$. Building on this result, we extend the derivation of TD-FLOW from Farebrother et al. (2025) to the new Bellman equation in (4) (full derivation in Appendix C.1), arriving at what we call the *Temporal Difference Horizon Consistency (*TD-HC*)* loss:

$$\begin{aligned} \ell_{\text{TD-HC}}(\theta; \bar{\theta}, \beta, \gamma) = (1-\gamma) \mathbb{E}_{\substack{t \sim \mathcal{U}([0,1]),\, (S,A,S') \sim \mathcal{D} \\ X_0 \sim p_0(\cdot),\, X_t = (1-t)X_0 + tS'}} & \left[\left\| v_t(X_t \mid S, A, \gamma; \theta) - (S' - X_0) \right\|^2 \right] \\ + \gamma \frac{1-\gamma}{1-\beta} \mathbb{E}_{\substack{t \sim \mathcal{U}([0,1]),\, (S,A,S') \sim \mathcal{D}, A' \sim \pi(\cdot|S') \\ X_t \sim \psi_t(\cdot|S', A', \beta; \bar{\theta})_\sharp p_0(\cdot)}} & \left[\left\| v_t(X_t \mid S, A, \gamma; \theta) - v_t(X_t \mid S', A', \beta; \bar{\theta}) \right\|^2 \right] \\ + \gamma \frac{\gamma - \beta}{1-\beta} \mathbb{E}_{\substack{t \sim \mathcal{U}([0,1]),\, (S,A,S') \sim \mathcal{D}, A' \sim \pi(\cdot|S') \\ S^+ \sim \psi_1(\cdot|S', A', \beta; \bar{\theta})_\sharp p_0(\cdot),\, A^+ \sim \pi(\cdot|S^+) \\ X_t \sim \psi_t(\cdot|S^+, A^+, \gamma; \bar{\theta})_\sharp p_0(\cdot)}} & \left[\left\| v_t(X_t \mid S, A, \gamma; \theta) - v_t(X_t \mid S^+, A^+, \gamma; \bar{\theta}) \right\|^2 \right]. \end{aligned} \tag{5}$$

When $\gamma = \beta$, the third term vanishes, and we recover the original TD-FLOW loss. In practice, we sample effective horizons $\tau_\gamma \sim \mathcal{U}[1/1-\gamma_{\min}, 1/1-\gamma_{\max}]$ and $\tau_\beta \sim \mathcal{U}[1/1-\gamma_{\min}, \tau_\gamma]$, and convert them to discount factors via $\gamma = 1 - 1/\tau_\gamma$ and $\beta = 1 - 1/\tau_\beta$. Additionally, we only apply horizon consistency (i.e., $\beta \neq \gamma$) to a small proportion of each mini-batch. This is motivated by the fact that the consistency term requires sampling from the model's own predictions at horizon β and using these samples as conditioning for the longer horizon γ, meaning errors in the model's current predictions can compound and introduce bias. Restricting the consistency term to a small fraction of the batch, we gain the benefits of horizon alignment while ensuring the majority of updates come from TD-FLOW. This design follows standard practice for consistency-like generative modeling (e.g., Frans et al., 2025; Geng et al., 2025; Boffi et al., 2025; Ai et al., 2026).

3.3 Compositional Planning

With the machinery for training policy-conditioned GHMs across multiple timescales now in place, we can turn to the central question: how to use them to solve downstream tasks? Given a reward function $r : \mathsf{S} \to \mathbb{R}$, our goal is to find a sequence of policies that maximizes expected return. Since the learned GHMs already capture the full complexity of how policies evolve in the environment, evaluating Q_γ^ν from Lemma 1 requires only specifying the policy embeddings $z_1, \ldots, z_n$, hence planning reduces to the following optimization problem:

$$\max_{a_1, z_1, \ldots, z_n} Q_\gamma^{\pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}}(s, a_1)\,. \tag{6}$$

The switching probabilities $\{\alpha_i\}$ control how long each policy executes before transitioning to the next, and are treated as hyperparameters. Once the optimal sequence $(a_1^*, z_1^*, \ldots, z_n^*)$ is identified, we execute the first action a_1^* followed by the policy $\pi_{z_1^*}$, replanning at future states as needed.

Notably, this planning objective unifies several existing approaches as special cases. By varying the switching probabilities $\{\alpha_i\}$, one can interpolate between action-level control and planning over policies by:

- Setting $\alpha_1 = \cdots = \alpha_n = 1$, which reduces to optimizing over sequences of primitive actions, equivalent to Model-Predictive Control with horizon n.
- Setting $\alpha_1 = 1$ and $\alpha_2 = \cdots = \alpha_n = 0$, yielding *Generalized Policy Improvement* (*GPI*; Barreto et al., 2017).
- Setting $\alpha_1 = \cdots = \alpha_{n-1} = \alpha$ for some fixed $\alpha \in (0, 1)$, which recovers *Geometric Generalized Policy Improvement* (*GGPI*; Thakoor et al., 2022).

We refer to our approach – which allows distinct switching probabilities $\alpha_1, \ldots, \alpha_{n-1} \in (0, 1)$ – as CompPlan. Likewise, we refer to action-level planning as ActionPlan. Crucially, the same pretrained GHMs power all of these methods. By conditioning on policies and a continuum of timescales, our framework spans one-step world models ($\alpha = 1$) through long-horizon policy composition ($\alpha_i \in (0, 1)$), unifying previously disparate paradigms.

Optimization via random shooting. The maximization in (6) is tractable when policies are indexed by a finite set, but becomes challenging for large or continuous Z. In this paper, we focus on goal-conditioned policies where $\mathsf{Z} = \mathsf{S}$ and $z \in \mathsf{S}$ represents a subgoal. Here, the key difficulty lies in proposing good candidate subgoals without searching over the entire state space. Our solution is to use the GHMs themselves as a proposal distribution. Given a goal $g \in \mathsf{S}$, we generate waypoints from $z_0 := s$ by composing $m_{\beta_i}^{\pi_g}$ over horizons $\{\beta_i\}$ as:

$$a_i \sim \pi_g(\cdot \mid z_i), \quad z_{i+1} \sim m_{\beta_{i+1}}^{\pi_g}(\cdot \mid z_i, a_i), \quad \text{for } i = 0, \ldots, n-1\,.$$

This produces a sequence of subgoals $(z_1, \ldots, z_n)$ that naturally guides progress towards the final objective. Alternatively, we can sample from an unconditional GHM that predicts plausible successor states under the data distribution (i.e., the behavior policy). To enable this, we stochastically mask the policy encoding as $z = \varnothing$ during training, bootstrapping with the dataset action at the next state in (3) when masked.

With a proposal distribution, planning reduces to random shooting (Matyáš, 1965): we (1) sample m candidate sequences $\{(z_1^{(i)}, \ldots, z_n^{(i)})\}_{i=1}^m$ from the proposal distribution; (2) evaluate $Q_\gamma^{\nu^{(i)}}(s, a_1^{(i)})$

for each candidate switching policy $\nu^{(i)} := \pi_{z_1^{(i)}} \xrightarrow{\alpha_1} \pi_{z_2^{(i)}} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n^{(i)}}$ using Lemma 1 where $a_1^{(i)} \sim \pi_{z_1^{(i)}}(\cdot \mid s)$; and (3) select the sequence $(a_1^*, z_1^*, \ldots, z_n^*)$ with the highest value $Q_\gamma^{\nu^*}(s, a_1^*)$. The full method is summarized in Algorithm 2.

4 EXPERIMENTS

Our empirical evaluation tests the core hypothesis of this work: that learning a jumpy world model over a diverse collection of parameterized policies enables effective and efficient compositional planning. We begin by describing the experimental setting and the training procedure for the base policies. We subsequently compare the performance of these policies against that of compositional planning over them. Additionally, we benchmark COMPPLAN against other test-time planning approaches and hierarchical methods. Finally, we examine the effect of the proposed consistency loss on both model accuracy and planning performance. Additional ablations on the replanning frequency, planning objective, and proposal distribution are provided in Appendix D.

4.1 EXPERIMENTAL SETUP

Benchmark and Dataset All experiments are conducted using the OGBench benchmark (Park et al., 2025a), which provides challenging long-horizon robotic manipulation and locomotion tasks structured as offline goal-conditioned reinforcement learning problems. We focus on ant navigation tasks across different maze topologies (MEDIUM, LARGE, and GIANT) as well as multi-cube robotic arm manipulation. For both policy and GHM training, we use the *standard* NAVIGATE and PLAY offline datasets for ANTMAZE and CUBE, respectively.

Base Policies The effectiveness of compositional planning depends on the quality and characteristics of the underlying policies. We train five distinct policy types, each exhibiting different tradeoffs between GHM learning and planning performance: 1) Goal-Conditioned TD3 (GC-TD3; Pirotta et al., 2024); 2) Goal-Conditioned 1-Step RL (GC-1S); 3) Contrastive RL (CRL; Eysenbach et al., 2022); 4) Goal-Conditioned Behavior Cloning (GC-BC; Lynch et al., 2019; Ghosh et al., 2021); and 5) Hierarchical Flow Behavior Cloning (HFBC; Park et al., 2025b). Additional details are provided in Appendix F.1.

4.2 HOW DO POLICIES AFFECT COMPOSITIONAL PLANNING?

For each policy family, we train a GHM in an off-policy manner using the TD-HC loss described in §3.2. GHMs are trained for 3M gradient steps using the Adam optimizer (Kingma & Ba, 2015) with a batch size of 256. The model architecture follows a U-Net-style design, similar to Farebrother et al. (2025). Both the timestep t and the discount factor γ are embedded by first applying a sinusoidal embedding to increase dimensionality, followed by a two-layer MLP with mish activations (Misra, 2019). For the discount embedding, we further concatenate the vector $[\gamma, 1-\gamma, -\log(1-\gamma)]$, where $-\log(1-\gamma)$ corresponds to the logarithm of the effective horizon; we find this improves the model's sensitivity to the discount factor. Other conditioning information, such as the state-action pair and policy embedding z, is processed through an additional MLP and added to both the time and discount embeddings. The network incorporates conditioning information via FiLM modulation (Perez et al., 2018). When training each GHM, we apply the

Table 1: Success rate (↑) of base policies π_g (Zero Shot) and compositional planning with GHMs (COMPPLAN; ours) averaged over tasks. We report the mean and standard deviation over 3 seeds. We highlight relative increases and decreases in performance w.r.t. the base policies. Additionally, we **bold** the best performance for each domain.

Domain	CRL		GC-1S		GC-BC		GC-TD3		HFBC	
	Zero Shot	COMPPLAN	Zero Shot	COMPPLAN	Zero Shot	COMPPLAN	Zero Shot	COMPPLAN	Zero Shot	COMPPLAN
ANTMAZE-MEDIUM	0.88	**0.97 (0.02)**	0.56	0.87 (0.05)	0.49	0.85 (0.08)	0.65	0.65 (0.03)	0.94	0.94 (0.01)
ANTMAZE-LARGE	0.84	0.90 (0.00)	0.21	0.61 (0.04)	0.18	0.73 (0.02)	0.23	0.48 (0.05)	0.78	**0.92 (0.02)**
ANTMAZE-GIANT	0.16	0.29 (0.03)	0.00	0.02 (0.00)	0.00	0.03 (0.01)	0.00	0.01 (0.01)	0.42	**0.79 (0.04)**
CUBE-1	0.28	0.86 (0.02)	0.37	0.66 (0.02)	0.90	**0.99 (0.01)**	0.58	0.91 (0.01)	0.80	0.97 (0.01)
CUBE-2	0.02	0.50 (0.03)	0.10	0.57 (0.09)	0.15	**0.97 (0.01)**	0.12	0.82 (0.01)	0.76	0.77 (0.02)
CUBE-3	0.01	0.73 (0.02)	0.01	0.67 (0.02)	0.09	**0.92 (0.01)**	0.12	0.83 (0.04)	0.64	0.83 (0.03)
CUBE-4	0.00	0.39 (0.04)	0.01	0.60 (0.02)	0.00	**0.76 (0.03)**	0.00	0.57 (0.03)	0.24	0.67 (0.03)

horizon-consistency objective (5) (i.e., $\beta \neq \gamma$) to 25% of each mini-batch in ANTMAZE and 12.5% in CUBE. We also train the unconditional model (i.e., $z = \varnothing$) for 10% of each mini-batch; these two proportions do not overlap.

During evaluation, we tailor the proposal distribution to each domain based on their distinct characteristics. In ANTMAZE, states are separated by large temporal distances and physical barriers, so an unconditional proposal would waste most samples on irrelevant regions of the state space; we therefore sample 256 subgoal sequences from the goal-conditioned GHM. In CUBE, tasks consist of short pick-and-place sequences with many viable paths, making the unconditional GHM a natural fit; we sample 1024 sequences to cover the broader proposal. Ablations of these choices are provided in Appendix D.5.

We begin by comparing the zero-shot performance of the base policies against our compositional planning approach. Table 1 details these results (averaged over three seeds), showing that compositional planning consistently improves upon the zero-shot policies. This demonstrates the effectiveness of our method in selecting policy sequences that exceed the performance of the best individual policy. While improvements are evident across all domains, the gains are particularly pronounced in complex, long-horizon tasks such as ANTMAZE-GIANT and CUBE-$\{3,4\}$, where success rates can rise from 10% to 90% in the most extreme cases.

Among the evaluated policy classes, HFBC emerges as the most consistent zero-shot performer. COMPPLAN further improves HFBC, notably in ANTMAZE-GIANT and CUBE-4. CRL, in contrast, is effective in ANTMAZE but underperforms in CUBE. We hypothesize this stems from the inductive bias in CRL's representation learning, which approximates the goal-conditioned value function as $Q(s, a, g) \approx \phi(s, a)^\top \psi(g)$. This factorization effectively captures the ant's spatial position – yielding robust navigation in ANTMAZE – but fails to encode complex object-related features, resulting in weaker CUBE policies. Notably, incorporating planning not only improves upon CRL's strong baseline in ANTMAZE, but also achieves non-trivial success rates in CUBE, demonstrating that our approach can extract utility from base policies otherwise limited by their inductive bias.

Interestingly, the other policies exhibit the opposite trend. When evaluated zero-shot, they show weak performance in medium to long-horizon tasks – ANTMAZE-MEDIUM, ANTMAZE-LARGE, and CUBE tasks all prove challenging, with CUBE success rates falling below $10-15\%$. However, these policies prove remarkably effective when integrated into our compositional planning framework, with success rates climbing above 70% in many tasks. Taken together, these results suggest that zero-shot metrics tell us little about how well a policy will compose; its utility may only become clear when orchestrated with other policies.

4.3 How Does Compositional Planning Compare to Other Planning and Hierarchical Approaches?

Our results in §4.2 demonstrate that compositional planning unlocks capabilities beyond what any single policy achieves alone. We now situate our approach among existing planning and hierarchical methods, revealing that COMPPLAN's advantages stem from its unique combination of temporal abstraction and flexible composition.

Recall from §3.3 that different choices of switching probabilities recover existing methods as special cases. This observation suggests a natural ablation: comparing COMPPLAN against these special cases can disentangle the contribution of its key components. At one extreme lies *Generalized Policy Improvement* (GPI; Barreto et al., 2017) which sets $\alpha_1 = 1$ and commits to a single policy for the remainder of the episode. GPI leverages our GHMs to estimate each policy's value, selecting the best in-class policy at each timestep according to:

$$\max_{z, A \sim \pi_z(\cdot|s)} Q_\gamma^{\pi_z}(s, a) = (1-\gamma)^{-1} \mathbb{E}_{S \sim m_\gamma^{\pi_z}(\cdot|s,a)} \left[r(S) \right] .$$

GPI thus serves as a test of whether composing policies offers an advantage over merely selecting among them.

At the other extreme, we compare against action-level planning (ACTIONPLAN), which sets $\alpha_1 = \cdots = \alpha_n = 1$ and operates at the granularity of individual actions rather than policies. To fairly isolate the effect of temporal abstraction, we train a dedicated one-step world model $\tilde{p}(\cdot|s, a)$ using the same flow-matching framework, network architecture, and training procedure as our GHMs –

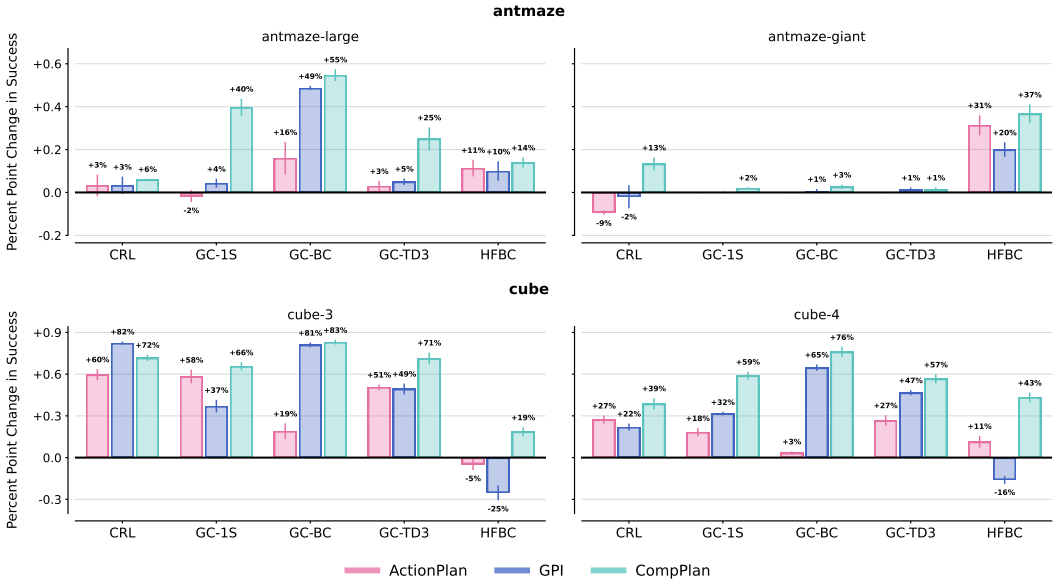

Figure 1: Percentage point change (↑) over zero-shot policies. We compare: action-level planning (ACTIONPLAN) with a world model; generalized policy improvement (GPI) and compositional planning (COMPPLAN; ours) with GHMs.

differing only in the removal of policy and discount conditioning. Given this model, we optimize the following objective:

$$\arg\max_{A_1,\ldots,A_n} \sum_{k=1}^{n} \gamma^k r(S_{k+1}) \,,$$

where $S_{k+1} \sim \tilde{p}(\cdot \mid S_k, A_k)$ and $A_k \sim \pi_g(\cdot \mid S_k)$. This comparison directly tests whether planning over sequences of policies – enabled by jumpy predictions – confers an advantage over action-level planning.

Figure 1 reveals a clear pattern: on long-horizon tasks, COMPPLAN substantially outperforms both alternatives, achieving an 89% relative improvement over GPI and a 201% gain over ACTIONPLAN (averaged across policies and long-horizon domains). These gains indicate that neither policy selection alone nor action-level planning captures the full benefit of our compositional framework. Rather, the combination of planning over sequences of policies at multiple timescales unlocks strong long-horizon capabilities.

Furthermore, we compare against methods that learn hierarchical structure during training: HIQL (Park et al., 2023) and SHARSA (Park et al., 2025b), the current state-of-the-art on OGBench. These approaches train high-level policies to select subgoals or skills, whereas COMPPLAN performs composition at test time using pre-trained GHMs. Table 2 presents this comparison where COMPPLAN employs HFBC base policies similar to SHARSA. Our approach consistently outperforms both hierarchical methods, with the largest margins on the most challenging tasks. In CUBE-4, COMPPLAN achieves 67% success compared to 9% for SHARSA and 0% for HIQL – demonstrating that test-time composition can surpass learned hierarchies when tasks demand flexible, long-horizon reasoning. Notably, COMPPLAN requires no task-specific training, suggesting a promising alternative to task-specific hierarchical methods.

Table 2: Success rate (↑) of hierarchical baselines and compositional planning with HFBC policies (COMPPLAN; ours). For each domain, we highlight the best performance.

Domain	HIQL	SHARSA	HFBC	COMPPLAN
ANTMAZE-MEDIUM	0.96 (0.01)	0.91 (0.03)	0.94	0.94 (0.01)
ANTMAZE-LARGE	0.91 (0.02)	0.88 (0.03)	0.78	0.92 (0.02)
ANTMAZE-GIANT	0.65 (0.05)	0.56 (0.07)	0.42	0.79 (0.04)
CUBE-1	0.15 (0.03)	0.70 (0.03)	0.84	0.97 (0.01)
CUBE-2	0.06 (0.02)	0.60 (0.07)	0.70	0.77 (0.02)
CUBE-3	0.03 (0.01)	0.50 (0.09)	0.54	0.83 (0.03)
CUBE-4	0.00 (0.00)	0.09 (0.04)	0.34	0.67 (0.03)

4.4 How Does Horizon Consistency Affect GHM Learning?

Having established that compositional planning yields strong empirical gains, we now turn inward to examine one of our methodological contributions: the horizon consistency objective from §3.2. We investigate its impact through two complementary lenses – generative fidelity and downstream planning – revealing that consistency plays distinct roles at different stages of the pipeline.

We begin by asking whether enforcing consistency across timescales produces more accurate GHM predictions. To isolate this effect, we train two GHMs – one using TD-FLOW and the other using TD-HC – keeping all else fixed. We obtain ground-truth samples by executing 64 policy rollouts from 256 randomly selected (state, goal) pairs and resampling 2048 visited states according to $t \sim \mathrm{Geom}(1-\gamma)$. We then draw an equal number of samples from each GHM and compute the Earth Mover's Distance (EMD; Rubner et al., 2000) between the two sets. As Table 3 demonstrates, consistency systematically improves accuracy at long horizons. The effect is particularly striking in ANTMAZE, where the maze structure imposes hard constraints on reachability. For example, employing TD-HC with GC-1S policies in ANTMAZE-GIANT results in a 28% reduction in EMD. Qualitatively, Figure 9 shows that TD-HC on ANTMAZE-GIANT leads to fewer samples erroneously traversing walls, a failure mode that compounds over long horizons.

Table 3: Accuracy (EMD; ↓) of GHMs trained with our horizon consistency loss (TD-HC) and without (TD-FLOW) for discount factor $\gamma = 0.995$. We highlight the best performing method.

Domain	CRL		GC-1S	
	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)
ANTMAZE-MEDIUM	4.41 (0.05)	4.22 (0.06)	4.40 (0.02)	4.22 (0.03)
ANTMAZE-LARGE	5.24 (0.07)	4.81 (0.03)	5.12 (0.18)	4.67 (0.04)
ANTMAZE-GIANT	6.77 (0.49)	5.74 (0.06)	7.29 (0.69)	5.25 (0.08)
CUBE-1	1.60 (0.02)	1.57 (0.03)	1.43 (0.00)	1.33 (0.03)
CUBE-2	2.36 (0.03)	2.23 (0.02)	1.86 (0.04)	1.71 (0.01)
CUBE-3	2.15 (0.02)	2.10 (0.02)	1.80 (0.04)	1.71 (0.03)
CUBE-4	2.41 (0.03)	2.34 (0.01)	2.13 (0.03)	2.05 (0.03)

Given these accuracy improvements, one might expect correspondingly large gains in planning. Surprisingly, Table 9 tells a different story: planning success rates are nearly identical with and without consistency, averaging only a 5% relative improvement. These findings are not contradictory but rather illuminate when consistency matters most. Our planning procedure evaluates candidate sequences using effective horizons $\{\beta_i\}$ in the range of $50 - 100$ steps (i.e., $\beta_i \in [0.98, 0.99]$), not the $200+$ step horizons where consistency provides its largest accuracy gains. At these moderate timescales, the base TD-FLOW objective already learns sufficiently accurate models to rank policies correctly. The consistency objective thus provides a margin of safety for long-horizon predictions without being strictly necessary for the planning horizons we employ in OGBench. This suggests that practitioners facing longer-horizon tasks would benefit most from the consistency loss herein.

5 Discussion

This work reframes pre-trained policies not as isolated controllers but as composable primitives – building blocks to be sequenced. Jumpy world models provide the mechanism: by predicting successor states for many policies across a continuum of horizons, they enable planning over behavior rather than primitive actions. Empirically, compositional planning consistently outperforms individual policies, hierarchical methods, and action-level planning, with striking gains at long horizons. Looking ahead, we see many promising directions: learning state-dependent switching probabilities, jointly learning policies and predictive models, employing more sample-efficient model-predictive control methods, and exploring jumpy world models in learned latent spaces.

Acknowledgements

The authors thank Harley Wiltzer, Arnav Jain, Pierluca D'oro, Nate Rahn, Michael Rabbat, Yann Ollivier, Marlos C. Machado, Michael Bowling, Adam White, and Doina Precup for useful discussions that helped improve this work. MGB is supported by the Canada CIFAR AI Chair program and NSERC. Finally, this work was made possible by the Python community, particularly NumPy (Harris et al., 2020), Matplotlib (Hunter, 2007), Seaborn (Waskom, 2021), Einops (Rogozhnikov, 2022), and Mujoco (Todorov et al., 2012).

REFERENCES

Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an RL agent. In *International Conference on Machine Learning (ICML)*, 2025.

Xinyue Ai, Yutong He, Albert Gu, Ruslan Salakhutdinov, J. Zico Kolter, Nicholas M. Boffi, and Max Simchowitz. Joint distillation for fast likelihood evaluation and sampling in flow-based models. In *International Conference on Learning Representations (ICLR)*, 2026.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations (ICLR)*, 2023.

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations, (ICLR)*, 2023.

Lucas N. Alegre, Agon Serifi, Ruben Grandia, David Müller, Espen Knoop, and Moritz Bächer. Amor: Adaptive character control through multi-objective reinforcement learning. In *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2025.

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew J. Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *CoRR*, abs/2506.09985, 2025.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, 2017.

Akhil Bagaria, Jason K. Senthil, and George Konidaris. Skill discovery for exploration and planning using deep skill graphs. In *International Conference on Machine Learning (ICML)*, 2021.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. TD-JEPA: Latent-predictive representations for zero-shot reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor features for transfer in reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2017.

André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Jonathan Hunt, Shibl Mourad, David Silver, and Doina Precup. The option keyboard: Combining skills in reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(4):341–379, 2003.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *CoRR*, abs/2410.24164, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning (CORL)*, 2025.

Léonard Blier. *Some Principled Methods for Deep Reinforcement Learning*. PhD thesis, Université Paris-Saclay, 2022.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *CoRR*, abs/2101.07123, 2021.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Rémi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael S. Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong T. Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023a.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023b.

John C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

Wilka Carvalho, Andre Saraiva, Angelos Filos, Andrew K. Lampinen, Loic Matthey, Richard L. Lewis, Honglak Lee, Satinder Singh, Danilo Jimenez Rezende, and Daniel Zoran. Combining behaviors with the successor features keyboard. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Edoardo Cetin, Ahmed Touati, and Yann Ollivier. Finer behavioral foundation models via auto-regressive features and advantage weighting. In *Reinforcement Learning Conference (RLC)*, 2025.

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

Sydney Chapman. On the brownian displacements and thermal diffusion of grains suspended in a non-uniform fluid. *Proceedings of the Royal Society of London*, 119(781):34–54, 05 1928.

Chang Chen, Hany Hamed, Doojin Baek, Taegu Kang, Yoshua Bengio, and Sungjin Ahn. Extendable long-horizon planning via hierarchical multiscale diffusion. *CoRR*, abs/2503.20102, 2025.

Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *International Conference on Learning Representations (ICLR)*, 2024.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*, 2018.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 1993.

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

Cyrus Derman. *Finite State Markovian Decision Processes*. Academic Press, 1970.

Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, 1998.

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

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Kuan Fang, Patrick Yin, Ashvin Nair, and Sergey Levine. Planning to practice: Efficient online fine-tuning by composing goals in latent space. In *International Conference on Intelligent Robots and Systems (IROS)*, 2022.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Rémi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. In *International Conference on Machine Learning (ICML)*, 2025.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Unsupervised zero-shot reinforcement learning via functional reward encodings. In *International Conference on Machine Learning (ICML)*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations (ICLR)*, 2025.

Dror Freirich, Tzahi Shimkin, Ron Meir, and Aviv Tamar. Distributional multivariate policy evaluation and exploration with the Bellman GAN. In *International Conference on Machine Learning (ICML)*, 2019.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

Michael Gimelfarb, Andre Barreto, Scott Sanner, and Chi-Guhn Lee. Risk-aware transfer in reinforcement learning using successor features. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Google DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025.

Nico Gürtler and Georg Martius. Long-horizon planning with predictable skills. In *Reinforcement Learning Conference (RLC)*, 2025.

Nico Gürtler, Dieter Büchler, and Georg Martius. Hierarchical reinforcement learning with timed subgoals. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Neural Information Processing Systems (NeurIPS)*, 2022.

Charles R. Harris, Millman K. Jarrod, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, and Aaron C. Courville. Riemannian diffusion models. In *Neural Information Processing Systems (NeurIPS)*, 2022.

John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9 (3):90–95, 2007.

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

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

Yuu Jinnai, David Abel, David Hershkowitz, Michael Littman, and George Konidaris. Finding options that minimize planning time. In *International Conference on Machine Learning (ICML)*, 2019.

Leslie Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *International Conference on Machine Learning (ICML)*, 1993a.

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

Kacper Kapusniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael M. Bronstein, Joey Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold. In *Neural Information Processing Systems (NeurIPS)*, 2024.

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

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

Andrei Kolmogoroff. Über die analytischen methoden in der wahrscheinlichkeitsrechnung. *Mathematische Annalen*, 104(1):415–458, 1931.

Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Neural Information Processing Systems (NeurIPS)*, 2016.

Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *CoRR*, abs/2510.21890, 2025.

Kyowoon Lee and Jaesik Choi. State-covering trajectory stitching for diffusion planners. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations (ICLR)*, 2019.

Yitang Li, Zhengyi Luo, Tonghe Zhang, Cunxi Dai, Anssi Kanervisto, Andrea Tirinzoni, Haoyang Weng, Kris Kitani, Mateusz Guzek, Ahmed Touati, Alessandro Lazaric, Matteo Pirotta, and Guanya Shi. BFM-zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Chunlok Lo, Kevin Roice, Parham Mohammad Panahi, Scott M. Jordan, Adam White, Gabor Mihucz, Farzane Aminmansour, and Martha White. Goal-space planning with subgoal models. *Journal of Machine Learning Research (JMLR)*, 25(330):1–57, 2024.

Yunhao Luo, Utkarsh A. Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris M. Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. In *International Conference on Learning Representations (ICLR)*, 2024.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning (CoRL)*, 2019.

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

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations (ICLR)*, 2018.

Marlos C. Machado, André Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research (JMLR)*, 24:80:1–80:69, 2023.

Josef Matyáš. Random optimization. *Automation and Remote control*, 26(2):246–253, 1965.

Meta. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Utkarsh A. Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning (CoRL)*, 2023.

Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *CoRR*, abs/1908.08681, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Kevin Murphy. Reinforcement learning: An overview. *CoRR*, abs/2412.05265, 2024.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations (ICLR)*, 2020.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Neural Information Processing Systems (NeurIPS)*, 2019.

NVIDIA. GR00T N1: an open foundation model for generalist humanoid robots. *CoRR*, abs/2503.14734, 2025.

OpenAI. OpenAI o1 system card. *CoRR*, abs/2412.16720, 2024.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *International Conference on Machine Learning (ICML)*, 2024.

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

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes RL scalable. In *Neural Information Processing Systems (NeurIPS)*, 2025b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025c.

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In *Neural Information Processing Systems (NeurIPS)*, 1997.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40 (4):144:1–144:20, 2021.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4):94:1–94:17, 2022.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Marek Petrik and Bruno Scherrer. Biasing approximate dynamic programming with a lower discount factor. In *Neural Information Processing Systems (NeurIPS)*, 2008.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a VLA that learns from experience. *CoRR*, abs/2511.14759, 2025.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *International Conference on Learning Representations (ICLR)*, 2024.

Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2000.

Doina Precup and Richard S. Sutton. Multi-time models for temporally abstract planning. In *Neural Information Processing Systems (NeurIPS)*, 1997.

Doina Precup, Richard S. Sutton, and Satinder Singh. Planning with closed-loop macro actions. In *AAAI Fall Symposium on Model-directed Autonomous Systems*, 1997.

Doina Precup, Richard S. Sutton, and Satinder Singh. Theoretical results on reinforcement learning with temporally abstract options. In *European conference on machine learning*, 1998.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloé Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross B. Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations (ICLR)*, 2025.

Rafael Rodriguez-Sanchez and George Konidaris. Learning abstract world model for value-preserving planning with options. In *Reinforcement Learning Conference (RLC)*, 2024.

Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations (ICLR)*, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351, pp. 234–241, 2015.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning (ICML)*, 2015.

Jürgen Schmidhuber. Learning to generate sub-goals for action sequences. In *Artificial neural networks*, pp. 967–972, 1991.

Liam Schramm and Abdeslam Boularias. Bellman diffusion models. *CoRR*, abs/2407.12163, 2024.

Lucy Xiaoyang Shi, Joseph J Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2022.

Harshit Sikchi, Siddhant Agarwal, Pranaya Jajoo, Samyak Parajuli, Caleb Chuck, Max Rudolph, Peter Stone, Amy Zhang, and Scott Niekum. RLZero: Direct policy inference from language without in-domain supervision. In *Neural Information Processing Systems (NeurIPS)*, 2025a.

Harshit Sikchi, Andrea Tirinzoni, Ahmed Touati, Yingchen Xu, Anssi Kanervisto, Scott Niekum, Amy Zhang, Alessandro Lazaric, and Matteo Pirotta. Fast adaptation with behavioral foundation models. In *Reinforcement Learning Conference (RLC)*, 2025b.

David Silver and Kamil Ciosek. Compositional planning using optimal option models. In *International Conference on Machine Learning (ICML)*, 2012.

Satinder Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Richard S. Sutton. TD models: Modeling the world at a mixture of time scales. In *International Conference on Machine Learning (ICML)*, 1995.

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.

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.

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. CALM: conditional adversarial latent models for directable virtual characters. In *ACM SIGGRAPH*, 2023.

Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. Maskedmimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics*, 43(6):209:1–209:21, 2024.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition. In *International Conference on Machine Learning (ICML)*, 2022.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *International Conference on Learning Representations (ICLR)*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Manan Tomar, Philippe Hansen-Estruch, Philip Bachman, Alex Lamb, John Langford, Matthew E Taylor, and Sergey Levine. Video occupancy models. *CoRR*, abs/2407.09533, 2024.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *International Conference on Learning Representations (ICLR)*, 2023.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

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 (ICML)*, 2017.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Michael L. Waskom. Seaborn: Statistical data visualization. *Journal of Open Source Software*, 6 (60):3021, 2021.

Harley Wiltzer, Jesse Farebrother, Arthur Gretton, and Mark Rowland. Foundations of multivariate distributional reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2024a.

Harley Wiltzer, Jesse Farebrother, Arthur Gretton, Yunhao Tang, André Barreto, Will Dabney, Marc G. Bellemare, and Mark Rowland. A distributional analogue to the successor representation. In *International Conference on Machine Learning (ICML)*, 2024b.

Runzhe Wu, Masatosh Uehara, and Wen Sun. Distributional offline policy evaluation with predictive error guarantees. In *International Conference on Machine Learning (ICML)*, 2023.

Kevin Xie, Homanga Bharadhwaj, Danijar Hafner, Animesh Garg, and Florian Shkurti. Latent skill planning for exploration and transfer. In *International Conference on Learning Representations (ICLR)*, 2021.

Jaesik Yoon, Hyeonseo Cho, Doojin Baek, Yoshua Bengio, and Sungjin Ahn. Monte carlo tree diffusion for system 2 planning. In *International Conference on Machine Learning (ICML)*, 2025a.

Jaesik Yoon, Hyeonseo Cho, Yoshua Bengio, and Sungjin Ahn. Fast monte carlo tree diffusion: 100× speedup via parallel and sparse planning. In *Neural Information Processing Systems (NeurIPS)*, 2025b.

Jingwei Zhang, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, Abbas Abdolmaleki, Dushyant Rao, Nicolas Heess, and Martin A. Riedmiller. Leveraging jumpy models for planning and fast learning in robotic domains. *CoRR*, abs/2302.12617, 2023.

Pushi Zhang, Xiaoyu Chen, Li Zhao, Wei Xiong, Tao Qin, and Tie-Yan Liu. Distributional reinforcement learning for multi-dimensional reward functions. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Chongyi Zheng, Seohong Park, Sergey Levine, and Benjamin Eysenbach. Intention-conditioned flow occupancy models. In *International Conference on Learning Representations (ICLR)*, 2026.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky T. Q. Chen. Guided flows for generative modeling and decision making. *CoRR*, abs/2311.13443, 2023.

Chuning Zhu, Xinqi Wang, Tyler Han, Simon S. Du, and Abhishek Gupta. Distributional successor features enable zero-shot policy optimization. In *Neural Information Processing Systems (NeurIPS)*, 2024.

APPENDICES

APPENDIX A EXTENDED RELATED WORKS **20**

APPENDIX B ALGORITHMS **22**

ALGORITHM 1: TEMPORAL DIFFERENCE FLOWS WITH HORIZON CONSISTENCY . . 22

ALGORITHM 2: COMPOSITIONAL PLANNING WITH JUMPY WORLD MODELS 23

ALGORITHM 3: GOAL-CONDITIONED PROPOSAL 23

ALGORITHM 4: UNCONDITIONAL PROPOSAL 23

APPENDIX C THEORETICAL RESULTS **24**

C.1 MULTI-TIMESCALE TEMPORAL DIFFERENCE FLOWS WITH HORIZON CONSISTENCY 25

APPENDIX D ADDITIONAL RESULTS **27**

D.1 COMPOSITIONAL PLANNING / ZERO-SHOT RESULTS 27

D.2 COMPOSITIONAL PLANNING / GPI / ACTION PLANNING RESULTS 29

D.3 ABLATION ON THE PLANNING FREQUENCY 31

D.4 ABLATION ON THE PLANNING OBJECTIVE 32

D.5 ABLATION ON THE PROPOSAL DISTRIBUTION 33

D.6 ABLATION ON THE CONSISTENCY OBJECTIVE 34

APPENDIX E QUALITATIVE GEOMETRIC HORIZON MODEL SAMPLES **35**

APPENDIX F EXPERIMENTAL DETAILS **37**

F.1 BASE POLICIES & HYPERPARAMETERS 37

F.2 GEOMETRIC HORIZON MODEL HYPERPARAMETERS 39

F.3 PLANNING HYPERPARAMETERS 40

A EXTENDED RELATED WORKS

Successor Measure Methods that learn (discounted) state occupancies employing temporal difference learning (Sutton, 1995) date back to the successor representation (Dayan, 1993) with more recent extensions like successor features (Barreto et al., 2017) and the successor measure (Blier et al., 2021; Blier, 2022). Janner et al. (2020) was the first to introduce a generative model of the successor measure with γ-models also referred to as geometric horizon models (Thakoor et al., 2022). Wiltzer et al. (2024b) additionally introduced δ-models that learn a distribution over γ-models enabling applications in distributional RL (Bellemare et al., 2017; 2023; Dabney et al., 2018). Many generative modeling techniques have been applied to learn these models, including GANs (Janner et al., 2020; Wiltzer et al., 2024b), normalizing flows (Janner et al., 2020), VAEs (Thakoor et al., 2022; Tomar et al., 2024), flow matching (Farebrother et al., 2025; Zheng et al., 2026), and diffusion (Schramm & Boularias, 2024; Farebrother et al., 2025). Closely related is work on distributional successor features also known as multi-variate distributional RL (Freirich et al., 2019; Gimelfarb et al., 2021; Zhang et al., 2021; Wu et al., 2023; Wiltzer et al., 2024a; Zhu et al., 2024), that involves modeling the distribution over cumulative finite-dimensional features induced by a policy. From a generative modeling perspective, our work generalizes temporal difference flows (Farebrother et al., 2025) and

shows how long-horizon predictions can be improved by training across multiple timescales with a novel horizon consistency objective.

Additionally, our compositional framework can be viewed as a generalization of Geometric Generalized Policy Improvement (GGPI; Thakoor et al., 2022) to arbitrary switching probabilities; however, our concrete formulation and empirical evaluation differ substantially from Thakoor et al. (2022). First, Thakoor et al. (2022) considers only four pre-trained policies in practice, and learn two separate GHMs with effective horizons of 5 and 10 steps respectively. Their experiments also restrict composition to sequences of only two policies. In contrast, we learn GHM models conditioned on a continuous family of policies and timescales with horizons up to $25\times$ longer, and evaluate GSPs with lengths ranging from 3 to 24 policies. As a result, our work not only enables a richer class of geometric switching policies but also performs a more comprehensive empirical validation of these techniques on challenging long-horizon tasks where temporal abstraction is most important.

Planning With Temporal Abstractions Several prior works have explored planning over subgoals or waypoints to solve long-horizon tasks (e.g., Nasiriany et al., 2019; Eysenbach et al., 2019; Nair & Finn, 2020; Chane-Sane et al., 2021; Fang et al., 2022; Hafner et al., 2022; Lo et al., 2024; Gürtler & Martius, 2025). These methods typically involve learning sub-goal conditioned policies together with a high-level dynamics model that predicts the outcomes of reaching these subgoals k steps in the future, then employ model predictive control to select subgoal sequences. In contrast, GHMs model the entire state-occupancy distribution rather than a fixed k-step lookahead, allowing planning over arbitrary reward functions rather than just goal-conditioned tasks.

A parallel line of work employs diffusion models (Vincent, 2011; Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Lai et al., 2025) for trajectory-level planning (e.g., Janner et al., 2022; Ajay et al., 2023; Zheng et al., 2023; Chen et al., 2025; Yoon et al., 2025a;b; Luo et al., 2025; Lee & Choi, 2025). Rather than modeling policy-induced dynamics, these methods train generative models over trajectory segments and perform planning within the denoising process. Execution then relies on inverse dynamics models to extract actions from planned state sequences. While powerful, this paradigm has notable limitations: (i) it requires learning accurate inverse dynamics; (ii) planning quality depends heavily on the trajectory distribution in the training data rather than on the capabilities of any particular policy; and (iii) these methods often assume access to oracle goal representations during the denoising process. In contrast, our work is policy-grounded: it directly composes the behaviors of pre-trained policies by predicting their induced state occupancies, rather than planning over abstract trajectory segments. This makes our approach agnostic to the policy class and avoids inverse dynamics entirely since actions are sampled directly from the base policies.

Closer to our approach, are methods that learn dynamics models over temporally extended behaviors (e.g., Xie et al., 2021; Shi et al., 2022; Zhang et al., 2023; Mishra et al., 2023; Gürtler & Martius, 2025). These methods learn general latent skills together with a high-level dynamics model predicting the outcomes of their execution (more precisely the states reached a fixed number of steps in the future), and use MPC to plan over these skills. However, these approaches suffer from the same limitations as the k-step subgoal methods discussed above. While we only report experiments for goal-based policies, COMPPLAN also enables planning on top of any set of parameterized skills or policies, for example, those learned via unsupervised RL methods (Borsa et al., 2019; Touati & Ollivier, 2021; Touati et al., 2023; Park et al., 2024; Frans et al., 2024; Cetin et al., 2025; Agarwal et al., 2025; Tirinzoni et al., 2025; Sikchi et al., 2025b;a; Bagatella et al., 2026).

Finally, our work connects to both the options framework (Sutton et al., 1999; Precup, 2000) and hierarchical RL (Schmidhuber, 1991; Kaelbling, 1993a; Parr & Russell, 1997; Barto & Mahadevan, 2003; Klissarov et al., 2025), which share a focus on temporal abstraction and multi-level decision making. Prior work has explored planning with options (e.g., Silver & Ciosek, 2012; Jinnai et al., 2019; Barreto et al., 2019; Carvalho et al., 2023; Rodriguez-Sanchez & Konidaris, 2024) or learning policies and value functions at multiple levels of abstraction (e.g., Precup & Sutton, 1997; Precup et al., 1997; 1998; Dayan & Hinton, 1992; Dietterich, 1998; Vezhnevets et al., 2017; Kulkarni et al., 2016; Gürtler et al., 2021; Nachum et al., 2018; Levy et al., 2019; Park et al., 2023; 2025b). Our compositional planning method can be viewed both as planning over options with a simple termination condition – where each policy terminates after a random number of steps – and as a hierarchical RL method that replaces the high-level policy with a test-time planning procedure.

B ALGORITHMS

We present the full TD-HC algorithm along with COMPPLAN and the two proposal distributions outlined herein.

Algorithm 1 Temporal Difference Flows with Horizon Consistency

Inputs: offline dataset $\mathcal{D}$, policy π, batch size K, Polyak coefficient ζ, randomly initialized weights θ, learning rate η, maximum discount factor $\gamma_{\max} \in [0, 1)$, horizon consistency proportion $\tau_c \in [0, 1]$.
for $n = 1, \ldots$ **do**
  Sample mini-batch $\{(S_k, A_k, S'_k, A'_k)\}_{k=1}^{K}$ from $\mathcal{D}$
  for $k = 1, \ldots, K$ **do**
    $t_k \sim \mathcal{U}([0, 1])$
    $\gamma_k \sim \mathcal{U}([0, \gamma_{\max}])$

    # One-Step Term
    $X_0 \sim p_0(\cdot)$
    $\vec{X}_{t_k} \leftarrow (1 - t_k)X_0 + t_k S'_k$
    $\vec{\ell}_k(\theta) = \left\| v_{t_k}(\vec{X}_{t_k} \mid S_k, A_k, \gamma_k; \theta) - (S'_k - X_0) \right\|^2$

    if $k \leq \lceil K \cdot \tau_c \rceil$ **then**
      $\beta_k \sim \mathcal{U}([0, \gamma_k])$
      # β-Bootstrap Term
      $X_0 \sim p_0(\cdot)$
      $\overset{\frown}{X}{}^{\beta}_{t_k} \leftarrow \psi_{t_k}(X_0 \mid S'_k, A'_k, \beta_k; \bar{\theta})$
      $\overset{\frown}{\ell}{}^{\beta}_k(\theta) = \left\| v_{t_k}(\overset{\frown}{X}{}^{\beta}_{t_k} \mid S_k, A_k, \gamma_k; \theta) - v_{t_k}(\overset{\frown}{X}{}^{\beta}_{t_k} \mid S'_k, A'_k, \beta_k; \bar{\theta}) \right\|^2$
      # γ-Bootstrap Term
      $(X_0, X''_0) \sim p_0(\cdot)$
      $S''_k \leftarrow \psi_1(X''_0 \mid S'_k, A'_k, \beta_k; \bar{\theta})$
      $A''_k \sim \pi(\cdot \mid S''_k)$
      $\overset{\frown}{X}{}^{\gamma}_{t_k} \leftarrow \psi_{t_k}(X_0 \mid S''_k, A''_k, \gamma_k; \bar{\theta})$
      $\overset{\frown}{\ell}{}^{\gamma}_k(\theta) = \left\| v_{t_k}(\overset{\frown}{X}{}^{\gamma}_{t_k} \mid S_k, A_k, \gamma_k; \theta) - v_{t_k}(\overset{\frown}{X}{}^{\gamma}_{t_k} \mid S''_k, A''_k, \gamma_k; \bar{\theta}) \right\|^2$
      # Mixture Loss
      $\ell_k(\theta) = (1 - \gamma_k)\vec{\ell}_k(\theta) + \gamma_k \frac{1-\gamma_k}{1-\beta_k} \overset{\frown}{\ell}{}^{\beta}_k(\theta) + \gamma_k \frac{\gamma_k - \beta_k}{1-\beta_k} \overset{\frown}{\ell}{}^{\gamma}_k(\theta)$
    else
      # γ-Bootstrap Term
      $X_0 \sim p_0(\cdot)$
      $\overset{\frown}{X}{}^{\gamma}_{t_k} \leftarrow \psi_{t_k}(X_0 \mid S'_k, A'_k, \gamma_k; \bar{\theta})$
      $\overset{\frown}{\ell}{}^{\gamma}_k(\theta) = \left\| v_{t_k}(\overset{\frown}{X}{}^{\gamma}_{t_k} \mid S_k, A_k, \gamma_k; \theta) - v_{t_k}(\overset{\frown}{X}{}^{\gamma}_{t_k} \mid S'_k, A'_k, \gamma_k; \bar{\theta}) \right\|^2$
      # Mixture Loss
      $\ell_k(\theta) = (1 - \gamma_k)\vec{\ell}_k(\theta) + \gamma_k \overset{\frown}{\ell}{}^{\gamma}_k(\theta)$
    end if
  end for
  # Compute loss
  $\ell(\theta) = \frac{1}{K} \sum_{k=1}^{K} \ell_k(\theta)$
  # Perform gradient step
  $\theta \leftarrow \theta - \eta \nabla_\theta \ell(\theta)$
  # Update parameters of target vector field
  $\bar{\theta} \leftarrow \zeta \bar{\theta} + (1 - \zeta)\theta$
end for

Algorithm 2 Compositional Planning with Jumpy World Models

Inputs: parameterized class of policies $\{\pi_z\}_{z\in\mathsf{Z}}$, geometric horizon model $m_\gamma^{\pi_z}$, policy sequence length K, proposal distribution $\rho : \mathsf{S} \to \mathscr{P}(\mathsf{Z}^K)$, number of proposals M, number of monte-carlo samples N, reward function r, effective discount factors $\{\beta_k\}_{k=1}^K$, mixture weights $\{w_k\}_{k=1}^K$
function COMPPLAN(s)
  for $i = 1, \ldots, M$ **do**
    # Sample policy sequence from proposal distribution
    $(z_1^{(i)}, \ldots, z_K^{(i)}) \sim \rho(\cdot \mid s)$

    # Sample initial action
    $a_1^{(i)} \sim \pi_{z_1^{(i)}}(\cdot \mid s)$

    # Monte Carlo Q-value estimation (Lemma 1)
    for $j = 1, \ldots, N$ **do**
      $(S_0, A_0) \leftarrow (s, a_1^{(i)})$
      for $k = 1, \ldots, K$ **do**
        $S_k \sim m_{\beta_k}^{\pi_{z_k^{(i)}}}(\cdot \mid S_{k-1}, A_{k-1})$
        $A_k \sim \pi_{z_{k+1}^{(i)}}(\cdot \mid S_k)$
      end for
      $\widehat{Q}^{(i,j)} \leftarrow (1-\gamma)^{-1} \sum_{k=1}^K w_k \cdot r(S_k)$
    end for
    $\widehat{Q}^{(i)} \leftarrow \frac{1}{N} \sum_{j=1}^N \widehat{Q}^{(i,j)}$
  end for

  # Select best candidate
  $i^* \leftarrow \arg\max_{i \in [\![M]\!]} \widehat{Q}^{(i)}$

  # Return optimal action and policy
  return $(a_1^{(i^*)}, z_1^{(i^*)})$
end function

Algorithm 3 Goal-Conditioned Proposal

Inputs: geometric horizon model $m_\gamma^{\pi_z}$, policy sequence length K, effective discount factors $\{\beta_k\}_{k=1}^K$
function GOALCONDPROPOSAL(s, g)
  # Chain GHM samples toward goal
  $z_0 \leftarrow s$
  for $k = 1, \ldots, K$ **do**
    $A_{k-1} \sim \pi_g(\cdot \mid z_{k-1})$
    $z_k \sim m_{\beta_k}^{\pi_g}(\cdot \mid z_{k-1}, A_{k-1})$
  end for
  return $(z_1, \ldots, z_K)$
end function

Algorithm 4 Unconditional Proposal

Inputs: unconditional (behavior policy μ) geometric horizon model m_γ^μ, policy sequence length K, effective discount factors $\{\beta_k\}_{k=1}^K$
function UNCONDPROPOSAL(s)
  # Chain unconditional GHM samples
  $z_0 \leftarrow s$
  for $k = 1, \ldots, K$ **do**
    $z_k \sim m_{\beta_k}^{\mu}(\cdot \mid z_{k-1})$
  end for
  return $(z_1, \ldots, z_K)$
end function

Figure 2: Two proposal distributions for goal-conditioned compositional planning. **Left:** GOALCONDPROPOSAL samples subgoal sequences by chaining GHM predictions conditioned on the goal $g \in \mathsf{S}$, guiding the agent toward the target. **Right:** Unconditional proposal samples from the behavior policy's GHM that can be trained alongside $m_\gamma^{\pi_z}$ by periodically setting $z = \varnothing, a = \varnothing$.

C THEORETICAL RESULTS

Theorem 1. *Let $\nu := \pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}$ be a geometric switching policy with global discount factor $\gamma \in (0,1)$, effective discount factors $\{\beta_k\}_{k=1}^n$, and weights $\{w_k\}_{k=1}^n$ from Definition 1. For any state-action pair (s,a), the successor measure of ν decomposes as:*

$$m_\gamma^\nu(\mathrm{d}s^+ \mid s,a) = \sum_{k=1}^n w_k \int_{\substack{s_1,\dots,s_{k-1}\\ a_1,\dots,a_{k-1}}} m_{\beta_1}^{\pi_{z_1}}(\mathrm{d}s_1 \mid s,a)\pi_{z_2}(\mathrm{d}a_1 \mid s_1) \cdots m_{\beta_k}^{\pi_{z_k}}(\mathrm{d}s^+ \mid s_{k-1},a_{k-1}).$$

Proof. Let's denote $\nu_{l:n} = \pi_{z_l} \xrightarrow{\alpha_l} \pi_{z_{l+1}} \dots \xrightarrow{\alpha_{n-1}} \pi_{z_n}$ the geometric switching policy that starts by π_{z_l}. we will proceed by induction over $l \in \{n, n-1, \dots 1\}$ to show that:

$$m_\gamma^{\nu_{l:n}}(\mathrm{d}s' \mid s,a) = \sum_{k=l}^n \frac{1-\gamma}{1-\beta_k}\left(\prod_{i=l}^{k-1}\frac{\gamma-\beta_i}{1-\beta_i}\right)\int_{\substack{s_l,\dots,s_{k-1}\\ a_l,\dots,a_{k-1}}} m_{\beta_l}^{\pi_{z_l}}(\mathrm{d}s_l \mid s,a)\pi_{z_{l+1}}(\mathrm{d}a_l \mid s_l)\dots m_{\beta_k}^{\pi_{z_k}}(\mathrm{d}s' \mid s_{k-1},a_{k-1}), \tag{7}$$

where $(s_{l-1},a_{l-1}) = (s,a)$.

For the case $l = n$, it is straightforward to see that the induction hypothesis (7) is satisfied since $m_\gamma^{\nu_{n:n}} = m_\gamma^{\pi_{z_n}}$.

Let us now assume that the induction hypothesis (7) holds for $l+1 \in \{n, n-1, \dots, 2\}$. Our goal is to demonstrate that it also holds for l.

After executing a single step of $\nu_{l:n}$, two outcomes are possible:: with probability $(1-\alpha_l)$, we remain committed to $\nu_{l:n}$, or with probability α_l, we switch to the next policy $\pi_{z_{l+1}}$, thereby continuing the episode with $\nu_{l+1:n}$. This leads to the following Bellman-equation:

$$m^{\nu_{l:n}} = (1-\gamma)P + \gamma(1-\alpha_l)P^{\pi_{z_l}}m^{\nu_{l:n}} + \gamma\alpha_l P^{\pi_{z_{l+1}}}m^{\nu_{l+1:n}}$$

which implies

$$\begin{aligned}
&(I-\gamma\beta_l P^{\pi_{z_l}})m^{\nu_{l:n}} = (1-\gamma)P + \gamma\alpha_l P^{\pi_{z_{l+1}}}m^{\nu_{l+1:n}}\\
&\implies m^{\nu_{l:n}} = (1-\gamma)(I-\gamma\beta_l P^{\pi_{z_l}})^{-1}P + \gamma\alpha_l(I-\gamma\beta_l P^{\pi_{z_l}})^{-1}P^{\pi_{z_{l+1}}}m^{\nu_{l+1:n}}\\
&\implies m^{\nu_{l:n}}(\mathrm{d}s' \mid s,a) = \frac{1-\gamma}{1-\beta_l}m_{\beta_l}^{\pi_l}(\mathrm{d}s' \mid s,a) + \frac{\gamma-\beta_l}{1-\beta_l}\int_{s_l} m_{\beta_l}^{\pi_l}(\mathrm{d}s_l \mid s,a)\pi_{\pi_{z_{l+1}}}(\mathrm{d}a_l \mid s_l)m^{\nu_{l+1:n}}(\mathrm{d}s' \mid s_l,a_l).
\end{aligned}$$

Using the induction hypothesis for $l+1$, we find:

$$\begin{aligned}
&m^{\nu_{l:n}}(\mathrm{d}s' \mid s,a)\\
&= \frac{1-\gamma}{1-\beta_l}m_{\beta_l}^{\pi_l}(\mathrm{d}s' \mid s,a)\\
&+ \frac{\gamma-\beta_l}{1-\beta_l}\int_{s_l} m_{\beta_l}^{\pi_l}(\mathrm{d}s_l \mid s,a)\pi_{\pi_{z_{l+1}}}(\mathrm{d}a_l \mid s_l)\\
&\times \sum_{k=l+1}^n \frac{1-\gamma}{1-\beta_k}\left(\prod_{i=l+1}^{k-1}\frac{\gamma-\beta_i}{1-\beta_i}\right)\int_{\substack{s_{l+1},\dots,s_{k-1}\\ a_{l+1},\dots,a_{k-1}}} m_{\beta_{l+1}}^{\pi_{z_{l+1}}}(\mathrm{d}s_{l+1} \mid s,a)\pi_{z_{l+2}}(\mathrm{d}a_{l+1} \mid s_{l+1})\dots m_{\beta_k}^{\pi_{z_k}}(\mathrm{d}s' \mid s_{k-1},a_{k-1})\\
&= \frac{1-\gamma}{1-\beta_l}m_{\beta_l}^{\pi_l}(\mathrm{d}s' \mid s,a)\\
&+ \sum_{k=l+1}^n \frac{1-\gamma}{1-\beta_k}\left(\prod_{i=l}^{k-1}\frac{\gamma-\beta_i}{1-\beta_i}\right)\int_{\substack{s_l,\dots,s_{k-1}\\ a_l,\dots,a_{k-1}}} m_{\beta_l}^{\pi_{z_l}}(\mathrm{d}s_l \mid s,a)\pi_{z_{l+1}}(\mathrm{d}a_l \mid s_l)\dots m_{\beta_k}^{\pi_{z_k}}(\mathrm{d}s' \mid s_{k-1},a_{k-1})\\
&= \sum_{k=l}^n \frac{1-\gamma}{1-\beta_k}\left(\prod_{i=l}^{k-1}\frac{\gamma-\beta_i}{1-\beta_i}\right)\int_{\substack{s_l,\dots,s_{k-1}\\ a_l,\dots,a_{k-1}}} m_{\beta_l}^{\pi_{z_l}}(\mathrm{d}s_l \mid s,a)\pi_{z_{l+1}}(\mathrm{d}a_l \mid s_l)\dots m_{\beta_k}^{\pi_{z_k}}(\mathrm{d}s' \mid s_{k-1},a_{k-1})
\end{aligned}$$

which shows the desired result. □

C.1 Multi-Timescale Temporal Difference Flows with Horizon Consistency

The results in this section generalize those found in Farebrother et al. (2025) to arbitrary mixture distributions.

Lemma 2. *Let $\{v_t^i\}_{i\in[\![N]\!]}$ a family of $N \in \mathbb{N}$ vector fields that generate the probability paths $\{p_t^i\}_{i\in[\![N]\!]}$. Then, the mixture probability path $p_t = \sum_i \lambda_i p_t^i$, where $\{\lambda_i\}_{i\in[\![N]\!]} \in [0,1]$ and $\sum_i \lambda_i = 1$ is generated by the vector field*

$$v_t := \frac{\sum_i \lambda_i p_t^i v_t^i}{\sum_i \lambda_i p_t^i}. \tag{8}$$

Proof. Since v_i^t generates p_t^i, we know from the continuity equation that:

$$\forall i \in [\![N]\!], \frac{\partial p_t^i}{\partial t} = \text{div}(p_t^i v_t^i)$$

where div denotes the divergence operator. Then, by linearity of div,

$$\begin{aligned}
\frac{\partial p_t}{\partial t} &= \frac{\partial \left(\sum_i \lambda_i p_t^i\right)}{\partial t} \\
&= \sum_i \lambda_i \text{div}(p_t^i v_t^i) \\
&= \text{div}\left(\sum_i \lambda_i p_t^i v_t^i\right) \\
&= \text{div}\left(\frac{\sum_i \lambda_i p_t^i v_t^i}{\sum_i \lambda_i p_t^i} \sum_i \lambda_i p_t^i\right) \\
&= \text{div}(v_t p_t).
\end{aligned}$$

Hence, (v_t, p_t) satisfies the continuity equation, which implies that v_t generates p_t. □

Lemma 3. *Let $\{v_t^i\}_{i\in[\![N]\!]}$ a family of $N \in \mathbb{N}$ vector fields that generate the probability paths $\{p_t^i\}_{i\in[\![N]\!]}$. For $\lambda_i \in [0,1]$ such that $\sum_i \lambda_i = 1$, the vector field $v_t = \frac{\sum_i \lambda_i p_t^i v_t^i}{\sum_i \lambda_i p_t^i}$ satisfies*

$$v_t = \underset{v:\mathbb{R}^d \to \mathbb{R}^d}{\arg\min} \left\{ \sum_i \lambda_i \, \mathbb{E}_{x_t \sim p_t^1} \left[\|v_t(x_t) - v_t^i(x_t)\|^2 \right] \right\}.$$

Proof. Let $\ell_t(v) := \sum_i \lambda_i \, \mathbb{E}_{x_t \sim p_t^i} \left[\|v_t(x_t) - v_t^i(x_t)\|^2 \right]$. The functional derivative of this quantity wrt v evaluated at some point x is

$$\nabla_v \ell_t(v)(x) = \sum_i \lambda_i p_i^t(x)(v_t(x) - v_t^i(x)).$$

Setting this to zero and solving for $v_t(x)$ yields the result. □

The consistency operator in equation (4) combines three different distributions. Lemmas 2 and 3 indicate that we can construct distinct probability paths for each distribution as follows

1. For the first distribution *i.e* $P(\cdot \mid s, a)$, We apply the standard Conditional Flow Matching (CFM) approach, where the probability path is defined as the marginal over a simple conditional path (specifically, we use the Optimal Transport (OT) path):

$$q_t(x|s,a) = \mathbb{E}_{S' \sim P(\cdot|s,a)} \left[\mathcal{N}(x; tS', (1-t)^2) \right] \tag{9}$$

where $\mathcal{N}(x; tS', (1-t)^2)$ is the gaussian distribution of mean tS' and variance $(1-t)^2$. This leads to the standard CFM objective:

$$\mathbb{E}_{\substack{t,(S,A,S') \\ X_0 \sim p_0, X_t = tS' + (1-t)X_0}} \left[\left\| v_t(X_t \mid S, A; \theta, \gamma) - (S' - X_0) \right\|^2 \right] \tag{10}$$

2. For the second distribution, *i.e.,* $\mathbb{E}_{S'\sim P(\cdot|s,a),A'\sim\pi(\cdot|S')}\left[m^{\pi}_{\beta}(\cdot \mid S', A')\right] = (P^{\pi}m^{\pi}_{\beta})(\cdot \mid s, a)$, we leverage that m^{π}_{β} is parametrized by a flow matching model to define the probability path:

$$q_t(x \mid s, a) = \mathbb{E}_{S'\sim P(\cdot|s,a),A'\sim\pi(\cdot|S')}\left[p_{0\#}\psi_t(\cdot \mid S', A'; \theta, \beta)\right] \tag{11}$$

q_t is a valid probability path, satisfying the boundary conditions: $q_0(x \mid s, a) = \mathbb{E}_{S'\sim P(\cdot|s,a),A'\sim\pi(\cdot|S')}\left[p_0(x)\right] = p_0(x)$ and $q_1 = P^{\pi}m^{\pi}_{\beta}$. q_t can be interpreted as aggregation of conditional paths $p_{0\#}\psi_t(\cdot \mid S', A', \beta, \theta)$, for which we have access to their vector field $v_t(X_t \mid S', A'; \theta, \beta)$. Using the equivalence between marginal flow matching and conditional flow matching (Lipman et al., 2023), we arrive at the following objective.

$$\mathbb{E}_{\substack{t,(S,A,S'),A'\sim\pi(\cdot|S') \\ X_t\sim p_{0\#}\psi_t(\cdot|S',A',\beta;\bar{\theta})}}\left[\left\|v_t(X_t \mid S, A, \gamma; \theta) - v_t(X_t \mid S', A', \beta; \bar{\theta})\right\|^2\right] \tag{12}$$

3. Similarly, for the third distribution $\mathbb{E}_{\substack{S'\sim P(\cdot|s,a),A'\sim\pi(\cdot|S') \\ S''\sim m^{\pi}_{\beta}(\cdot|S',A'),A''\sim\pi(\cdot|S'')}}\left[m^{\pi}_{\gamma}(\cdot \mid S'', A'')\right]$, we again leverage that m^{π}_{γ} is parametrized by flow matching model to define the following probability path:

$$q_t(x \mid s, a) = \mathbb{E}_{\substack{S'\sim P(\cdot|s,a),A'\sim\pi(\cdot|S') \\ S''\sim p_{0\#}\psi_1(\cdot|S',A',\beta;\bar{\theta}),A''\sim\pi(\cdot|S'')}}\left[p_{0\#}\psi_t(\cdot \mid S', A'; \theta, \gamma)\right] \tag{13}$$

q_t can be interpreted as aggregation of conditional paths $p_{0\#}\psi_t(\cdot \mid S', A', \gamma, \theta)$, for which we have access to their vector field $v_t(X_t \mid S', A', \gamma; \theta)$. Using the equivalence between marginal flow matching and conditional flow matching (Lipman et al., 2023), we arrive at the following objective.

$$\mathbb{E}_{\substack{t,(S,A,S'),A'\sim\pi(\cdot|S') \\ S''\sim p_{0\#}\psi_1(\cdot|S',A',\beta;\bar{\theta}) \\ X_t\sim p_{0\#}\psi_t(\cdot|S'',A'',\gamma;\bar{\theta})}}\left[\left\|v_t(X_t \mid S, A, \gamma; \theta) - v_t(X_t \mid S'', A'', \gamma; \bar{\theta})\right\|^2\right] \tag{14}$$

D ADDITIONAL RESULTS

D.1 COMPOSITIONAL PLANNING / ZERO-SHOT RESULTS

We report the full planning results here. Table 4 summarizes the success rate for each task. We evaluate 7 domains with 5 tasks per domain, for a total of 35 tasks. For each task we evaluate the base policy and COMPPLAN by rolling out 10 trajectories. We report standard deviation over the 3 seeds used for GHM training, for the base policy we do not have multiple seeds. We see that COMPPLAN is better than the base policy in almost all the tasks. Figure 3 shows a bar plot comparing the domain-averaged success rates of COMPPLAN and the zero-shot baseline.

Table 4: Success rate (↑) per task for base policies π_g (Zero Shot) and COMPPLAN. Mean and standard deviation reported over 3 seeds. Blue and red denote an increase and decrease w.r.t. zero-shot with gray indicating no significant difference.

Domain	Task	CRL		GC-1S		GC-BC		GC-TD3		HFBC	
		Zero Shot	COMPPLAN	Zero Shot	COMPPLAN	Zero Shot	COMPPLAN	Zero Shot	COMPPLAN	Zero Shot	COMPPLAN
ANTMAZE-MEDIUM	1	0.950	0.970 (0.030)	0.400	0.930 (0.070)	0.400	0.870 (0.090)	0.650	0.800 (0.060)	0.900	0.870 (0.030)
	2	0.950	0.900 (0.100)	0.800	1.000 (0.000)	0.550	0.830 (0.090)	0.900	0.530 (0.120)	0.900	0.970 (0.030)
	3	0.650	0.970 (0.030)	0.100	0.500 (0.120)	0.600	0.900 (0.060)	0.450	0.770 (0.030)	1.000	0.970 (0.030)
	4	0.900	1.000 (0.000)	0.650	0.930 (0.030)	0.100	0.770 (0.090)	0.550	0.600 (0.100)	1.000	0.970 (0.030)
	5	0.950	1.000 (0.000)	0.850	0.970 (0.030)	0.800	0.870 (0.090)	0.700	0.570 (0.070)	0.900	0.930 (0.030)
ANTMAZE-LARGE	1	0.800	0.830 (0.030)	0.100	0.770 (0.070)	0.150	0.570 (0.070)	0.300	0.500 (0.120)	0.800	0.970 (0.030)
	2	0.600	0.770 (0.030)	0.150	0.630 (0.120)	0.100	0.730 (0.030)	0.100	0.530 (0.090)	0.600	0.800 (0.000)
	3	0.850	0.930 (0.030)	0.800	0.830 (0.090)	0.550	0.900 (0.060)	0.750	0.500 (0.100)	1.000	0.970 (0.030)
	4	0.950	1.000 (0.000)	0.000	0.430 (0.030)	0.000	0.670 (0.090)	0.000	0.400 (0.060)	0.900	0.900 (0.060)
	5	1.000	0.970 (0.030)	0.000	0.370 (0.120)	0.100	0.770 (0.090)	0.000	0.470 (0.090)	0.600	0.970 (0.030)
ANTMAZE-GIANT	1	0.000	0.070 (0.030)	0.000	0.000 (0.000)	0.000	0.000 (0.000)	0.000	0.000 (0.000)	0.400	0.730 (0.090)
	2	0.000	0.670 (0.070)	0.000	0.000 (0.000)	0.000	0.000 (0.000)	0.000	0.030 (0.030)	0.300	0.830 (0.030)
	3	0.000	0.100 (0.060)	0.000	0.000 (0.000)	0.000	0.000 (0.000)	0.000	0.000 (0.000)	0.200	0.800 (0.060)
	4	0.500	0.230 (0.070)	0.000	0.000 (0.000)	0.000	0.000 (0.000)	0.000	0.000 (0.000)	0.500	0.730 (0.030)
	5	0.300	0.400 (0.120)	0.000	0.100 (0.000)	0.000	0.130 (0.030)	0.000	0.030 (0.030)	0.700	0.830 (0.030)
CUBE-1	1	0.200	0.930 (0.030)	0.300	0.470 (0.030)	1.000	1.000 (0.000)	0.500	0.900 (0.060)	0.800	0.970 (0.030)
	2	0.100	0.830 (0.090)	0.400	0.470 (0.030)	0.950	1.000 (0.000)	0.850	1.000 (0.000)	0.700	1.000 (0.000)
	3	0.500	0.900 (0.000)	0.300	0.900 (0.060)	0.950	1.000 (0.000)	0.600	0.900 (0.000)	0.900	1.000 (0.000)
	4	0.250	0.930 (0.030)	0.500	0.770 (0.030)	1.000	1.000 (0.000)	0.550	0.900 (0.060)	0.700	0.970 (0.030)
	5	0.350	0.700 (0.120)	0.350	0.700 (0.100)	0.600	0.970 (0.030)	0.400	0.870 (0.070)	0.900	0.900 (0.060)
CUBE-2	1	0.100	0.870 (0.030)	0.200	0.970 (0.030)	0.750	1.000 (0.000)	0.300	0.970 (0.030)	1.000	1.000 (0.000)
	2	0.000	0.500 (0.060)	0.200	0.500 (0.170)	0.000	1.000 (0.000)	0.150	0.930 (0.070)	0.900	0.930 (0.030)
	3	0.000	0.570 (0.030)	0.050	0.600 (0.100)	0.000	1.000 (0.000)	0.100	0.870 (0.030)	0.900	1.000 (0.000)
	4	0.000	0.070 (0.030)	0.000	0.400 (0.170)	0.000	0.870 (0.070)	0.000	0.430 (0.130)	0.300	0.200 (0.060)
	5	0.000	0.500 (0.100)	0.050	0.370 (0.070)	0.000	0.970 (0.030)	0.050	0.900 (0.060)	0.700	0.700 (0.100)
CUBE-3	1	0.050	1.000 (0.000)	0.050	0.930 (0.030)	0.450	1.000 (0.000)	0.400	1.000 (0.000)	0.900	1.000 (0.000)
	2	0.000	0.970 (0.030)	0.000	1.000 (0.000)	0.000	1.000 (0.000)	0.050	1.000 (0.000)	0.900	1.000 (0.000)
	3	0.000	0.930 (0.030)	0.000	0.870 (0.090)	0.000	1.000 (0.000)	0.150	0.930 (0.070)	0.800	0.970 (0.030)
	4	0.000	0.270 (0.090)	0.000	0.300 (0.060)	0.000	0.700 (0.060)	0.000	0.700 (0.060)	0.300	0.370 (0.120)
	5	0.000	0.470 (0.030)	0.000	0.230 (0.030)	0.000	0.900 (0.000)	0.000	0.530 (0.170)	0.300	0.800 (0.000)
CUBE-4	1	0.000	0.530 (0.070)	0.050	0.970 (0.030)	0.000	1.000 (0.000)	0.000	1.000 (0.000)	0.600	1.000 (0.000)
	2	0.000	0.870 (0.090)	0.000	0.900 (0.060)	0.000	1.000 (0.000)	0.000	0.970 (0.030)	0.400	0.830 (0.120)
	3	0.000	0.270 (0.120)	0.000	0.800 (0.000)	0.000	1.000 (0.000)	0.000	0.470 (0.030)	0.200	0.770 (0.030)
	4	0.000	0.030 (0.030)	0.000	0.100 (0.060)	0.000	0.170 (0.170)	0.000	0.170 (0.120)	0.000	0.200 (0.100)
	5	0.000	0.230 (0.130)	0.000	0.230 (0.070)	0.000	0.630 (0.130)	0.000	0.230 (0.030)	0.000	0.570 (0.130)

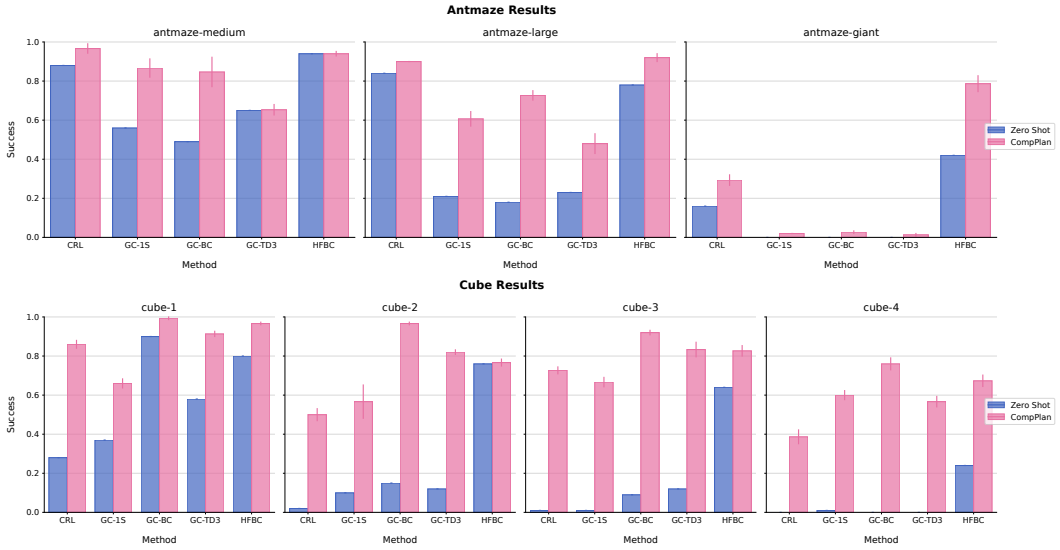

Figure 3: Success rate (↑) of base policies π_g (Zero Shot) and compositional planning as in Equation (6) with GHMs (COMPPLAN, ours) averaged over tasks.

D.2 Compositional Planning / GPI / Action Planning Results

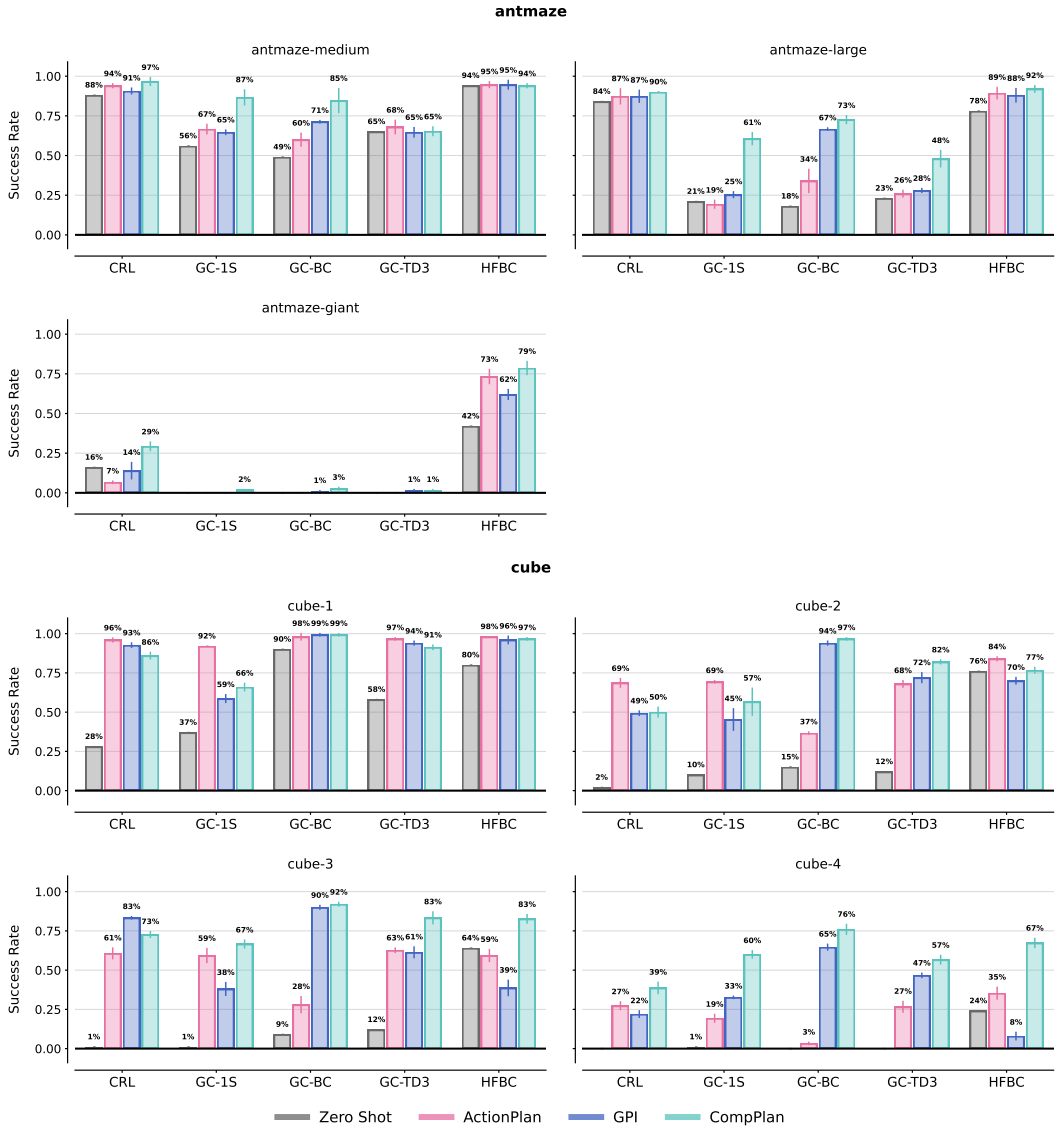

Figure 4: Success rate (↑) of planning vs. zero shot. We consider action-level planning (ActionPlan) with a world model; generalized policy improvement (GPI) and compositional planning (CompPlan; ours) with GHMs.

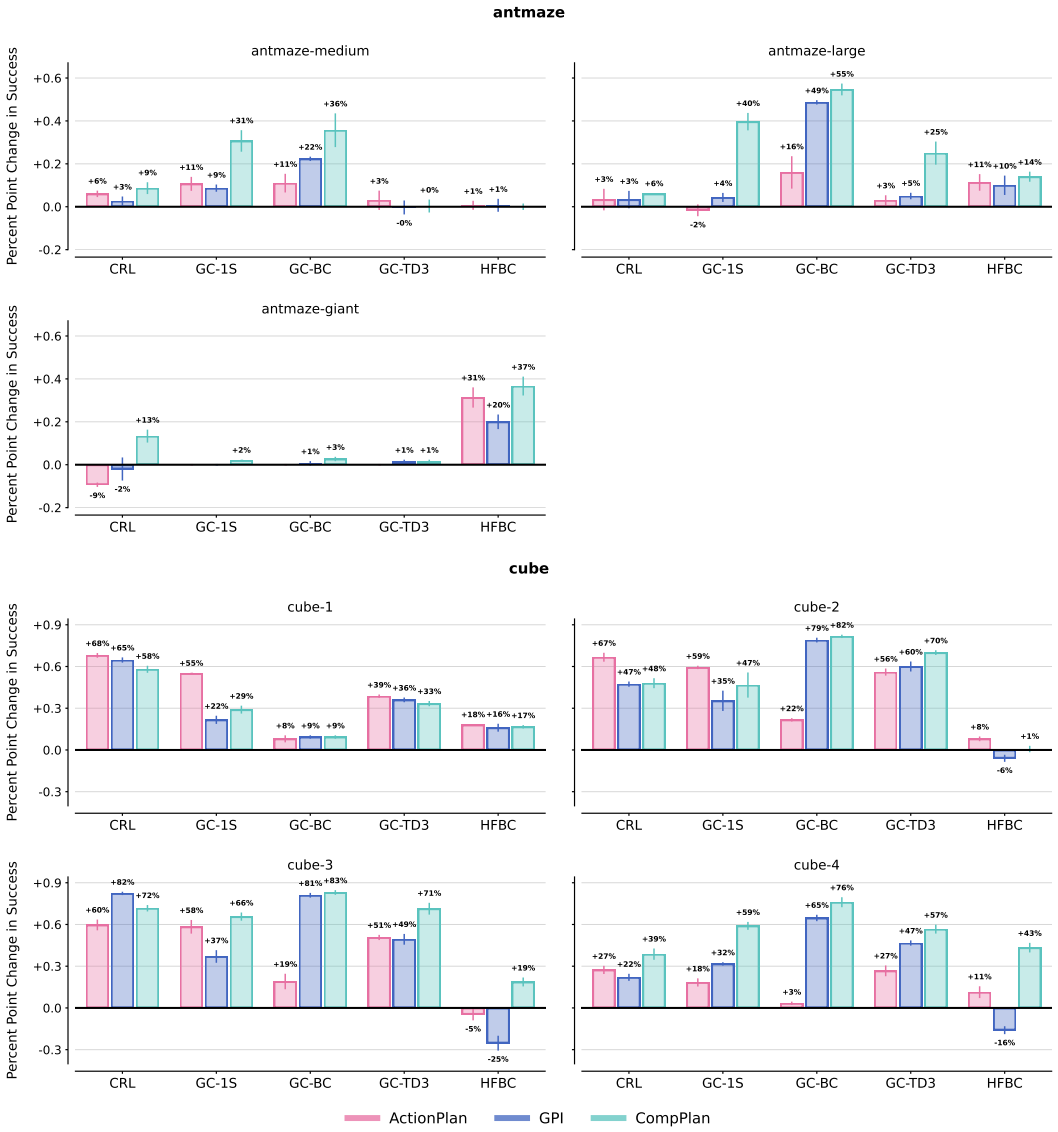

Figure 5: Percent point change (↑) of planning over zero shot. We consider action-level planning (ACTIONPLAN) with a world model; generalized policy improvement (GPI) and compositional planning (COMPPLAN; ours) with GHMs.

D.3 Ablation on the Planning Frequency

In this section, we investigate whether the planning cost can be amortized over time. Specifically, we compare the performance of planning at each time step with planning every N steps. In the latter case, we execute the action maximizing the CompPlan objective for the first step and then follow the policy π_{z_1} (i.e., the trajectory is $s, a^\star, s_1, \pi_{z_1}(s_1), s_2, \ldots, s_{N-1}, \pi_{z_1}(s_{N-1}), s_N$). Figure 6 reports the average success rate for each domain and method. On average, planning at every step leads to about a 20% improvement compared to planning every 5 steps. This is mostly due to a few cases—most notably, CRL policies in Cube—in which planning every 5 steps results in a large performance drop or complete unlearning. In the majority of experiments, planning every 5 steps does not substantially degrade overall performance, while significantly reducing planning time. Overall, this is a lever that can be used to trade off speed and performance. Finally, Table 5 provides a tabular summary of these results.

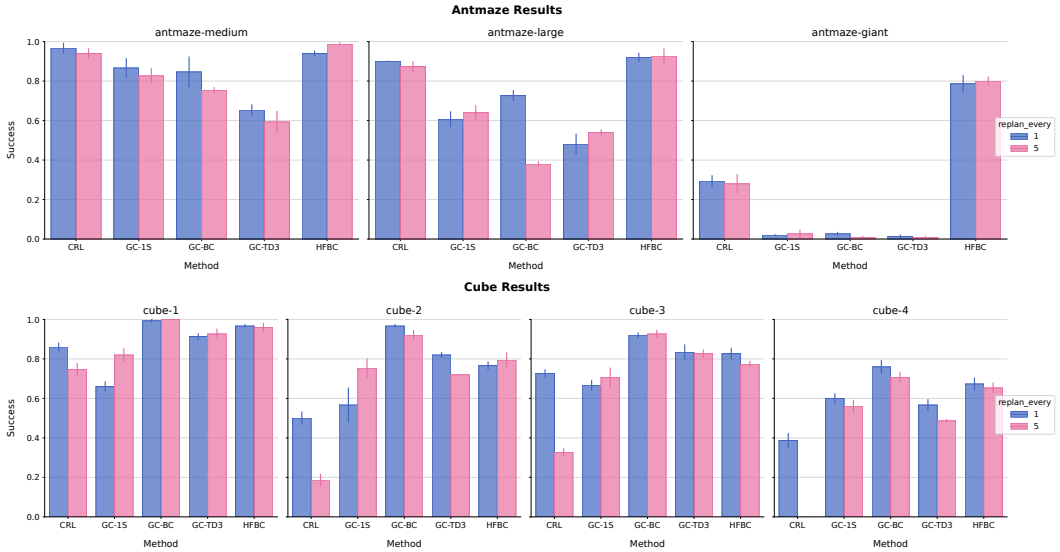

Figure 6: Success rate (↑) of CompPlan with different base policies when planning at every step of every 5 steps. We report mean and standard deviation 3 seeds for GHM training.

Table 5: Success rate (↑) of CompPlan with different base policies when replanning every 1 or 5 steps. We report mean and standard deviation over 3 seeds. Best method highlighted in blue; gray indicates no significant difference.

Domain	CRL		GC-1S		GC-BC		GC-TD3		HFBC	
	1 Step	5 Steps	1 Step	5 Steps	1 Step	5 Steps	1 Step	5 Steps	1 Step	5 Steps
ANTMAZE-MEDIUM	0.97 (0.02)	0.94 (0.02)	0.87 (0.05)	0.83 (0.04)	0.85 (0.08)	0.75 (0.01)	0.65 (0.03)	0.59 (0.05)	0.94(0.01)	0.99 (0.01)
ANTMAZE-LARGE	0.90 (0.00)	0.87 (0.02)	0.61(0.04)	0.64 (0.03)	0.73 (0.02)	0.38 (0.01)	0.48(0.05)	0.54 (0.01)	0.92(0.02)	0.93 (0.04)
ANTMAZE-GIANT	0.29 (0.03)	0.28 (0.05)	0.02(0.00)	0.03 (0.02)	0.03 (0.01)	0.01 (0.01)	0.01 (0.01)	0.01 (0.01)	0.79(0.04)	0.80 (0.02)
CUBE-1	0.86 (0.02)	0.75 (0.03)	0.66(0.02)	0.82 (0.03)	0.99(0.01)	1.00 (0.00)	0.91(0.01)	0.93 (0.02)	0.97 (0.01)	0.96 (0.02)
CUBE-2	0.50 (0.03)	0.19 (0.03)	0.57(0.09)	0.75 (0.05)	0.97 (0.01)	0.92 (0.02)	0.82 (0.01)	0.72 (0.00)	0.77(0.02)	0.79 (0.04)
CUBE-3	0.73 (0.02)	0.33 (0.02)	0.67(0.02)	0.71 (0.05)	0.92(0.01)	0.93 (0.02)	0.83 (0.02)	0.83 (0.02)	0.83 (0.03)	0.77 (0.01)
CUBE-4	0.39 (0.04)	0.00 (0.00)	0.60 (0.02)	0.56 (0.03)	0.76 (0.03)	0.71 (0.02)	0.57 (0.03)	0.49 (0.01)	0.67 (0.03)	0.65 (0.02)

D.4 Ablation on the Planning Objective

As mentioned in the main paper, we can leverage the GHM in several different planning approaches. In this section, we compare two strategies. The first is the approach presented in the paper, in which we optimize both the first action and the policy sequence $(z_1, \ldots, z_n)$:

$$\max_{a_1, z_1, \ldots, z_n} Q_\gamma^{\pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}}(s, a_1) \tag{15}$$

while the second does not involve action optimization

$$\max_{z_1, \ldots, z_n} V_\gamma^{\pi_{z_1} \xrightarrow{\alpha_1} \pi_{z_2} \cdots \xrightarrow{\alpha_{n-1}} \pi_{z_n}}(s) \tag{16}$$

The difference is that, when using (15), the first action is deterministic, whereas in (16) it is sampled according to π_{z_1}. Figure 7 shows that maximizing over the first action yields more consistent performance overall, with an average improvement of about 70%.[4] In contrast, planning over the z sequence can fail when the base policy is diffuse (i.e., highly stochastic). This occurs, for example, for CRL policies in the Cube domains and for GC-BC policies in AntMaze. Finally, Table 6 provides a tabular summary of these results.

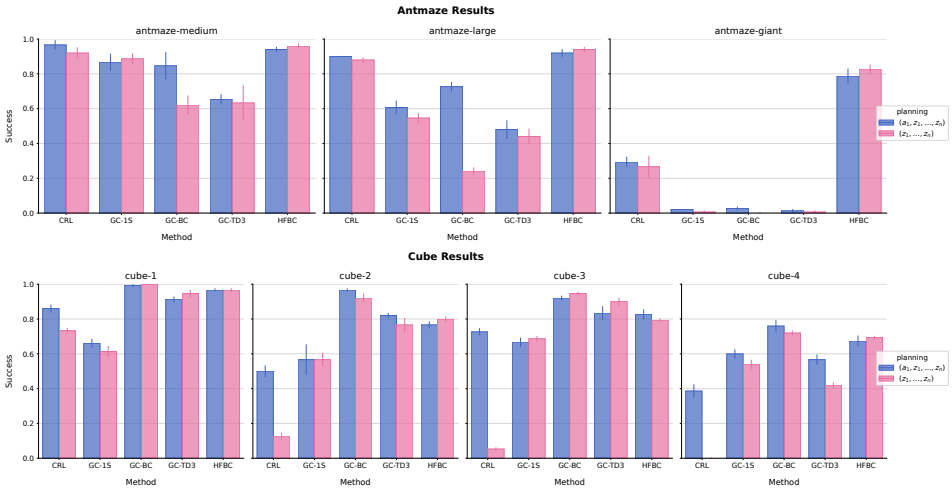

Figure 7: Success rate (↑) of CompPlan with different base policies when maximizing over $(a_1, z_1, \ldots z_n)$ (Eq. 15) or $(z_1, \ldots, z_n)$ (Eq. 16). We report mean and standard deviation 3 seeds for GHM training.

Table 6: Success rate (↑) of CompPlan with (Eq. 15) and without (Eq. 16) action maximization. We report mean and standard deviation over 3 seeds. Best method highlighted in blue; gray indicates no significant difference.

Domain	CRL		GC-1S		GC-BC		GC-TD3		HFBC	
	max Q	max V	max Q	max V	max Q	max V	max Q	max V	max Q	max V
ANTMAZE-MEDIUM	0.97 (0.02)	0.92 (0.03)	0.87 (0.05)	0.89 (0.03)	0.85 (0.08)	0.62 (0.05)	0.65 (0.03)	0.63 (0.10)	0.94 (0.01)	0.96 (0.01)
ANTMAZE-LARGE	0.90 (0.00)	0.88 (0.01)	0.61 (0.04)	0.55 (0.03)	0.73 (0.02)	0.24 (0.02)	0.48 (0.05)	0.44 (0.04)	0.92 (0.02)	0.94 (0.01)
ANTMAZE-GIANT	0.29 (0.03)	0.27 (0.06)	0.02 (0.00)	0.01 (0.01)	0.03 (0.01)	0.00 (0.00)	0.01 (0.01)	0.01 (0.01)	0.79 (0.04)	0.83 (0.02)
CUBE-1	0.86 (0.02)	0.73 (0.01)	0.66 (0.02)	0.61 (0.03)	0.99 (0.01)	1.00 (0.00)	0.91 (0.01)	0.95 (0.02)	0.97 (0.01)	0.97 (0.01)
CUBE-2	0.50 (0.03)	0.13 (0.02)	0.57 (0.04)	0.57 (0.04)	0.97 (0.01)	0.92 (0.02)	0.82 (0.01)	0.77 (0.04)	0.77 (0.02)	0.80 (0.01)
CUBE-3	0.73 (0.02)	0.05 (0.01)	0.67 (0.02)	0.69 (0.01)	0.92 (0.01)	0.95 (0.01)	0.83 (0.04)	0.90 (0.02)	0.83 (0.03)	0.79 (0.01)
CUBE-4	0.39 (0.04)	0.00 (0.00)	0.60 (0.02)	0.54 (0.02)	0.76 (0.03)	0.72 (0.01)	0.57 (0.03)	0.42 (0.01)	0.67 (0.03)	0.69 (0.01)

[4]This is mostly due to unsuccessful outcomes (i.e., near-zero performance) on certain tasks when Equation 16.

D.5 Ablation on the Proposal Distribution

In this section, we study the effect of the sampling distribution on the planning procedure. As mentioned in the main paper, GHMs are trained either with the policy condition z or with a learnable token $\varnothing$. In the latter case, the resulting GHM corresponds to the behavioral policy. We compare two planning strategies: (1) sampling from the GHM conditioned on the policy associated with the goal (*conditional proposal*); and (2) sampling from the GHM of the behavioral policy (*unconditional proposal*). Figure 8 shows that, in AntMaze, sampling from the unconditional distribution performs only marginally worse than the conditional proposal, highlighting the robustness of our planning procedure. Table 7 provides a tabular summary of these results.

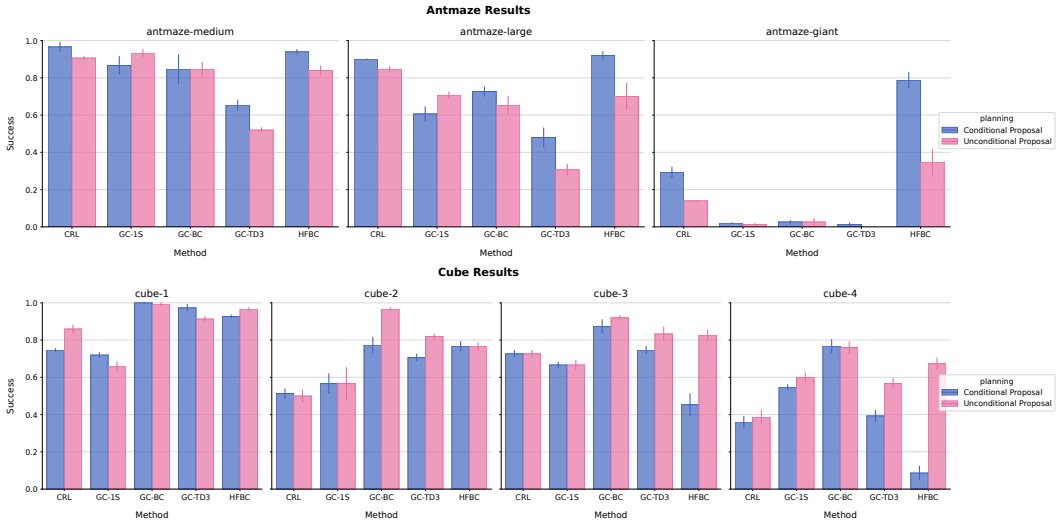

Figure 8: Success rate (↑) of CompPlan with different base policies when using the conditional or unconditional GHMs to propose subgoals. We report mean and standard deviation 3 seeds for GHM training.

Table 7: Success rate (↑) of CompPlan with conditional and unconditional proposal. We report mean and standard deviation 3 seeds. Best method highlighted in blue; gray indicates no significant difference.

Domain	CRL		GC-1S		GC-BC		GC-TD3		HFBC	
	Cond	Uncond	Cond	Uncond	Cond	Uncond	Cond	Uncond	Cond	Uncond
ANTMAZE-MEDIUM	0.97 (0.02)	0.91 (0.01)	0.87 (0.05)	0.93 (0.02)	0.85 (0.04)	0.85 (0.04)	0.65 (0.03)	0.52 (0.01)	0.94 (0.01)	0.84 (0.02)
ANTMAZE-LARGE	0.90 (0.00)	0.85 (0.01)	0.61 (0.04)	0.71 (0.02)	0.73 (0.02)	0.65 (0.05)	0.48 (0.05)	0.31 (0.03)	0.92 (0.02)	0.70 (0.07)
ANTMAZE-GIANT	0.29 (0.03)	0.14 (0.00)	0.02 (0.00)	0.01 (0.01)	0.03 (0.02)	0.03 (0.02)	0.01 (0.01)	0.00 (0.00)	0.79 (0.04)	0.35 (0.07)
CUBE-1	0.75 (0.01)	0.86 (0.02)	0.72 (0.01)	0.66 (0.02)	1.00 (0.00)	0.99 (0.01)	0.97 (0.02)	0.91 (0.01)	0.93 (0.01)	0.97 (0.01)
CUBE-2	0.51 (0.02)	0.50 (0.03)	0.57 (0.09)	0.57 (0.09)	0.77 (0.04)	0.97 (0.01)	0.71 (0.02)	0.82 (0.01)	0.77 (0.02)	0.77 (0.02)
CUBE-3	0.73 (0.02)	0.73 (0.02)	0.67 (0.02)	0.67 (0.02)	0.87 (0.04)	0.92 (0.01)	0.75 (0.02)	0.83 (0.04)	0.45 (0.06)	0.83 (0.03)
CUBE-4	0.36 (0.03)	0.39 (0.04)	0.55 (0.01)	0.60 (0.02)	0.77 (0.04)	0.76 (0.03)	0.39 (0.03)	0.57 (0.03)	0.09 (0.04)	0.67 (0.03)

D.6 Ablation on the Consistency Objective

This section expands upon §4.4, providing the complete empirical results for the Temporal Difference Horizon Consistency (TD-HC) objective across generative fidelity (Table 8), qualitative predictions (Figure 9), and downstream planning performance (Table 9).

Generative Fidelity Table 8 shows that TD-HC systematically improves the generative accuracy of the TD-FLOW baseline as measured by the Earth Mover's Distance (EMD; Rubner et al., 2000). These gains are especially pronounced in complex domains like ANTMAZE-GIANT where bootstrapping errors easily compound. Figure 9 visually confirms this effect: while both models perform comparably at shorter horizons ($\gamma = 0.99$), TD-FLOW suffers from severe compounding errors at longer horizons ($\gamma = 0.998$), and fails to capture the tail of the distribution. By anchoring its predictions with shorter-horizons, TD-HC successfully respects the topological constraints and properly captures the tail of the true successor state distribution.

Planning Performance Despite the generative advantages of TD-HC at extreme horizons, Table 9 reveals that downstream planning success rates remain broadly similar between the two methods. Because our planning procedure evaluates candidate sequences using moderate effective horizons ($\beta_i \in [0.98, 0.99]$, or roughly $50 - 100$ steps), it does not query the extreme timescales where TD-FLOW breaks down. We expect TD-HC to enable planning over more extreme horizons in the future as benchmarks evolve towards more complex tasks.

Table 8: Accuracy (EMD, ↓) of GHMs trained with (TD-HC) and without (TD-FLOW) our horizon consistency loss (§3.2). Best method is highlighted in blue.

Domain	CRL		GC-1S		GC-BC		GC-TD3		HFBC	
	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)
ANTMAZE-MEDIUM	4.41 (0.05)	4.22 (0.06)	4.40 (0.02)	4.22 (0.03)	4.56 (0.05)	4.36 (0.03)	4.68 (0.05)	4.44 (0.05)	3.38 (0.02)	3.22 (0.02)
ANTMAZE-LARGE	5.24 (0.07)	4.81 (0.03)	5.12 (0.18)	4.67 (0.04)	5.32 (0.03)	4.73 (0.02)	5.33 (0.07)	4.83 (0.04)	3.50 (0.05)	3.18 (0.01)
ANTMAZE-GIANT	6.77 (0.49)	5.74 (0.06)	7.29 (0.69)	5.25 (0.08)	6.46 (0.10)	4.95 (0.12)	6.51 (0.14)	5.24 (0.11)	3.78 (0.03)	3.00 (0.09)
CUBE-1	1.60 (0.02)	1.57 (0.03)	1.43 (0.00)	1.33 (0.03)	1.03 (0.01)	0.98 (0.01)	1.12 (0.01)	1.06 (0.01)	1.82 (0.06)	1.27 (0.01)
CUBE-2	2.36 (0.03)	2.23 (0.02)	1.86 (0.04)	1.71 (0.01)	1.47 (0.00)	1.42 (0.01)	1.89 (0.04)	1.80 (0.03)	2.29 (0.09)	1.54 (0.04)
CUBE-3	2.15 (0.02)	2.10 (0.02)	1.80 (0.04)	1.71 (0.03)	1.89 (0.02)	1.85 (0.02)	1.91 (0.06)	1.84 (0.04)	1.99 (0.09)	1.55 (0.02)
CUBE-4	2.41 (0.03)	2.34 (0.01)	2.13 (0.03)	2.05 (0.03)	2.33 (0.03)	2.22 (0.02)	2.21 (0.02)	2.15 (0.02)	2.05 (0.05)	1.61 (0.03)

Table 9: Success rate (↑) of CompPlan with consistency (TD-HC) and without consistency (TD-FLOW). Mean and standard deviation over 3 seeds. Best method highlighted in blue; gray indicates no significant difference.

Domain	CRL		GC-1S		GC-BC		GC-TD3		HFBC	
	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)	TD-FLOW (✗)	TD-HC (✓)
ANTMAZE-MEDIUM	0.95 (0.01)	0.97 (0.02)	0.85 (0.01)	0.87 (0.05)	0.91 (0.01)	0.85 (0.08)	0.71 (0.03)	0.65 (0.03)	0.95 (0.02)	0.94 (0.01)
ANTMAZE-LARGE	0.91 (0.03)	0.90 (0.00)	0.53 (0.09)	0.61 (0.04)	0.79 (0.01)	0.73 (0.02)	0.51 (0.03)	0.48 (0.05)	0.91 (0.03)	0.92 (0.02)
ANTMAZE-GIANT	0.31 (0.05)	0.29 (0.03)	0.01 (0.01)	0.02 (0.00)	0.02 (0.00)	0.03 (0.01)	0.01 (0.01)	0.01 (0.01)	0.81 (0.02)	0.79 (0.04)
CUBE-1	0.89 (0.03)	0.86 (0.02)	0.55 (0.02)	0.66 (0.02)	1.00 (0.00)	0.99 (0.01)	0.97 (0.01)	0.91 (0.01)	0.97 (0.01)	0.97 (0.01)
CUBE-2	0.41 (0.02)	0.50 (0.03)	0.57 (0.09)	0.57 (0.09)	0.95 (0.02)	0.97 (0.01)	0.85 (0.02)	0.82 (0.01)	0.84 (0.03)	0.77 (0.02)
CUBE-3	0.72 (0.02)	0.73 (0.02)	0.72 (0.01)	0.67 (0.02)	0.91 (0.02)	0.92 (0.01)	0.83 (0.04)	0.83 (0.04)	0.83 (0.03)	0.83 (0.03)
CUBE-4	0.36 (0.00)	0.39 (0.04)	0.56 (0.02)	0.60 (0.02)	0.75 (0.02)	0.76 (0.03)	0.56 (0.03)	0.57 (0.03)	0.64 (0.04)	0.67 (0.03)

E Qualitative Geometric Horizon Model Samples

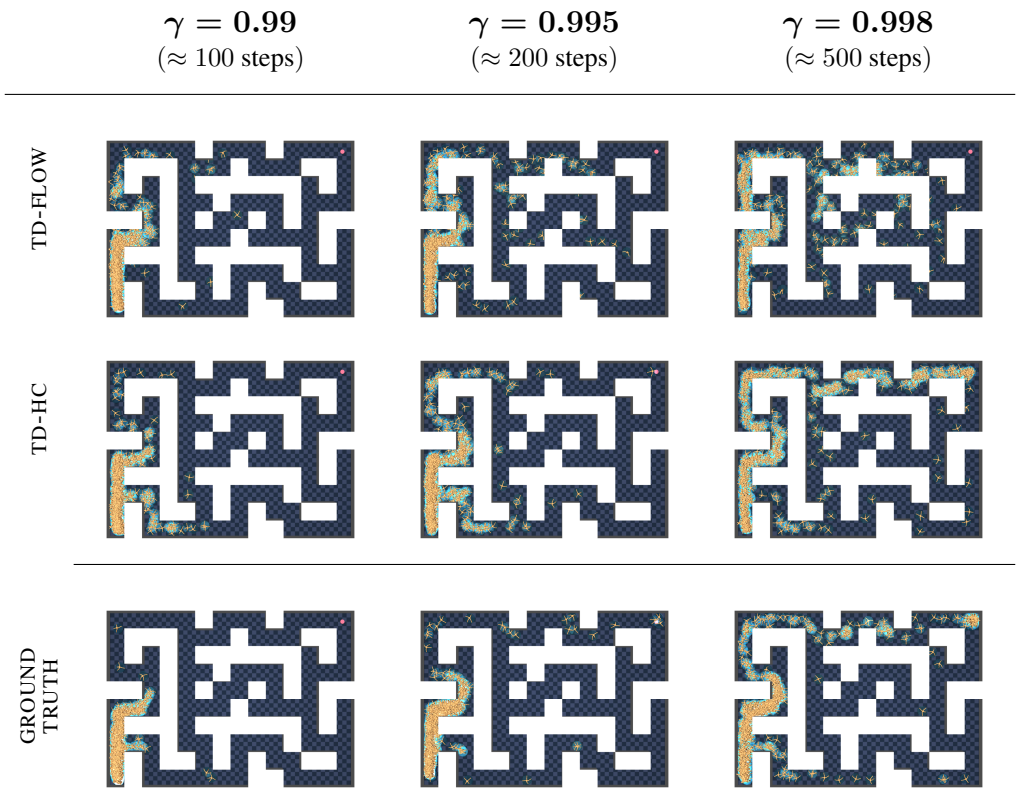

Figure 9: Qualitative plots of GHM samples on ANTMAZE-GIANT task 1 at different horizons $(0.99, 0.995, 0.998)$ from TD-FLOW and TD-HC with the last row depicting the ground truth discounted occupancy. As can be seen in the figure, performance is comparable at smaller horizons with TD-HC doing a much better job at capturing the true distribution as the horizon increases.

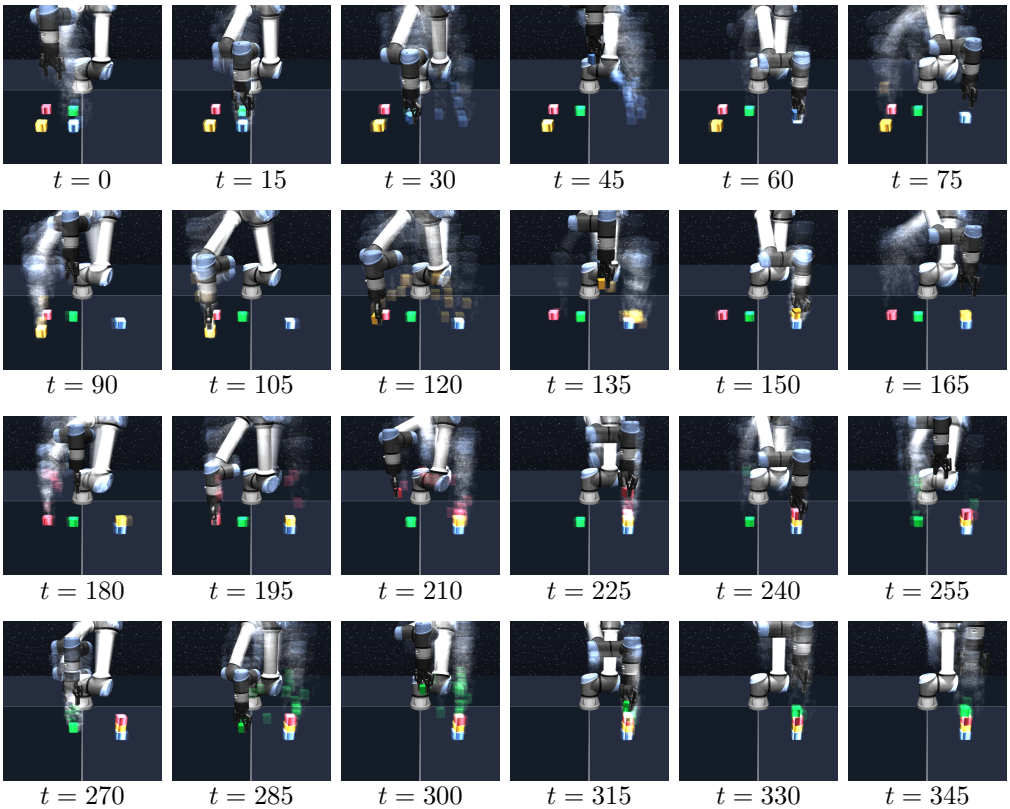

Figure 10: Qualitative visualization for an episode of CUBE-4-TASK-5. The robots and cubes shown with transparency indicated GHM samples from the first policy in the GSP.

F Experimental Details

F.1 Base Policies & Hyperparameters

Each policy class is chosen to highlight a different aspect of the pipeline (i.e., GHM learning and skill planning).

1. **Goal-Conditioned TD3 (GC-TD3; Pirotta et al., 2024)**: A standard goal-conditioned offline RL algorithm where we employ Flow Q-learning (FQL; Park et al., 2025c), a flow-based variant of TD3-BC (Fujimoto & Gu, 2021) for policy extraction. Due to bootstrapping on its own learned policy, actions may drift out-of-distribution (OOD) from the dataset, posing a challenge for off-policy predictive modeling (Levine et al., 2020). In particular, we use the standard TD3 critic loss (Fujimoto et al., 2018) to learn a goal-conditioned action-value function $Q_\phi(S, A, G)$ (Schaul et al., 2015):

$$\ell(\phi) = \mathbb{E}_{\substack{(S,A,S',G) \\ A' \sim \pi_G(\cdot|S')}} \left[\left(Q_\phi(S, A, G) - \mathbb{I}\{S' = G\} - \gamma Q_{\bar{\phi}}(S', A', G) \right)^2 \right] ,$$

   where (S, A, S') are transition sampled uniformly from the dataset, and G is a goal state drawn from the following mixture distribution: with probability 0.2, set $G = S'$, with probability 0.3 sample G uniformly from the dataset and with 0.5 set G to a randomly selected future state along the trajectory starting from (S, A), where the selection time step is drawn from a γ-geometric distribution.

   For policy training, we use the flow Q-learning procedure. We learn jointly a behavior policy $\pi(\cdot \mid S)$ and goal-conditioned policy $\pi_G(\cdot \mid S)$ parametrized by flow-matching vector field. The behavioral policy is trained by a standard conditional flow-matching objective (Lipman et al., 2023) on state-action pairs uniformly sampled from the dataset. The goal-conditioned policy is modeled as a one-step flow map. We denote by $\mu_\psi(S, X_0)$ and $\mu_\omega(S, G, X_0)$ the flow-maps of π and π_G, respectively.

$$\ell(\psi) = -\mathbb{E}_{\substack{(S,A,G) \\ X_0 \sim \mathcal{N}(0,I)}} \left[Q(S, \mu_\psi(S, G, X_0), G) + \lambda \|\mu_\psi(S, G, X_0) - \mu_\omega(S, X_0)\|^2 \right] , \quad (17)$$

   where λ is the distillation coefficient that controls the behavior cloning regularization. Here G is a goal state drawn from the following mixture distribution: with probability 0.5 sample G uniformly from the dataset, and with 0.5 set G to a randomly selected future state along the trajectory starting from (S, A).

2. **Goal-Conditioned 1-Step RL (GC-1S)**: A more conservative variant of GC-TD3 that bootstraps using the behavior policy via the dataset's actions. We expect this to yield easier-to-model occupancies as we no longer query the learned value function with OOD actions. Policy extraction on the resulting value function is also performed using FQL (17). In particular, we follow the same training as explained for TD3, and we change only the critic objective:

$$\ell(\phi) = \mathbb{E}_{(S,A,S',A',G)} \left[\left(Q_\phi(S, A, G) - \mathbb{I}\{S' = G\} - \gamma Q_{\bar{\phi}}(S', A', G) \right)^2 \right] ,$$

   where A' is now sampled from the dataset rather than from the learned policy.

3. **Contrastive RL (CRL; Eysenbach et al., 2022)**: An alternative value-based approach that uses contrastive learning to approximate the successor measure of the behavior policy (Eysenbach et al., 2022). In particular CRL learns a state-action $\phi(s, a)$ and goal encoder $\psi(g)$ by the following Monte-Carlo InfoNCE (van den Oord et al., 2018) contrastive loss:

$$\ell(\phi, \psi) = \mathbb{E}_{(S,A,G)} \left[-\phi(S, A)^\top \psi(G) \right] - \left[\log \sum_{(S,A,G')} \exp \left(\phi(S, A)^\top \psi(G') \right) \right] \quad (18)$$

   where G is γ-distributed sampled future state along the trajectory starting from (S, A) and G' is a state distribued uniformly from the dataset. Moreover, policy extraction is performed using FQL (17) with Q-function $Q(S, A, G)$ estimated as $\phi(S, A)^\top \psi(G)$.

4. **Goal-Conditioned Behavior Cloning (GC-BC; Lynch et al., 2019; Ghosh et al., 2021)**: A purely imitative policy trained via flow matching (Lipman et al., 2023) to mimic the high-quality trajectories in the dataset to reach specific goals using hindsight relabeling (Andrychowicz et al., 2017). Specially, we parametrize the policy by vector field $v_\phi(t, S, G)$

$$\ell(\phi) = \mathbb{E}_{\substack{t,S,A,G \\ X_0 \sim \mathcal{N}(0,I)}} \left[\|v_\phi(t, S, G) - (A - X_0)\|^2 \right], \tag{19}$$

   where G is γ-distributed sampled future state along the trajectory starting from (S, A). A key consequence of this value-free approach is that the policy struggles to generalize to distant goals, which can limit the effectiveness of proposing goal-directed "waypoints" during planning.

5. **Hierarchical Flow Behavior Cloning (HFBC; Park et al., 2025b)**: an imitative approach that trains two policies: a high-level policy is trained to predict subgoals that are h steps away from the current state for a fixed lookahead h, and the low-level policy is trained to predict actions to reach the given subgoal. Specially we parametrize both high-level and low-level policy by two vector fields $v_\phi(t, S, G)$ and $\nu_\psi(t, S, G)$ respectively.

$$\ell(\phi) = \mathbb{E}_{\substack{t,S_n,S_{n+h},G \\ X_0 \sim \mathcal{N}(0,I)}} \left[\|v_\phi(t, S_n, G) - (S_{n+h} - X_0)\|^2 \right],$$
$$\ell(\psi) = \mathbb{E}_{\substack{t,S_n,A_n,S_{n+h} \\ X_0 \sim \mathcal{N}(0,I)}} \left[\|v_\phi(t, S_n, S_{n+h}) - (A_n - X_0)\|^2 \right],$$

   where S_{n+h} is the h-steps way from the current state S_n and G is is γ-distributed sampled future state along the trajectory starting from S_n Consequently, the high-level policy can be used as a proposal for sub-goal distribution at planning time.

Table 10: Base policy hyperparameters. Parameters in { } denote sweeps performed over the values inside brackets.

Method	Hyperparameter	Antmaze	Cube
CRL **GC-TD3** **GC-1S**	FQL Distillation coefficient	$\{0.1, 0.15, 0.2, 0.3, 0.4\}$	$\{0.7, 0.8, 0.9, 1, 3\}$
	Discount factor	$\{0.995, 0.997\}$ for giant 0.99 otherwise	$\{0.99, 0.995\}$ for CUBE-4 0.99 otherwise
	Gradient steps	$\{1\text{M}, 3\text{M}\}$	500k for cube-$\{1, 2\}$ 1M for CUBE-3 3M for CUBE-4
GC-BC	Discount factor	$\{0.99, 0.995\}$ for giant $\{0.98, 0.99\}$ for medium $\{0.98, 0.99\}$ for large	$\{0.95, 0.96\}$ for CUBE-1 $\{0.96, 0.97\}$ for CUBE-2 $\{0.96, 0.97, 0.98\}$ for CUBE-3 $\{0.96, 0.98, 0.99\}$ for CUBE-4
	Gradient steps	125k	500k for CUBE-1 1M for CUBE-2 2M for CUBE-3 3M for CUBE-4
HFBC **SHARSA**	Lookahead	$[25, 50]$	$[25, 50]$
	Discount factor	$\{0.99, 0.995\}$	$\{0.95, 0.99, 0.995\}$
	Gradient steps	3M	1M for cube-$\{1, 2\}$ 3M for cube-$\{3, 4\}$

F.2 Geometric Horizon Model Hyperparameters

Table 11: Hyperparameters for Geometric Horizon Model pre-training.

	Hyperparameter	**Value**
Flow Matching (Lipman et al., 2023)	Probability Path	Conditional OT (σ=0)
	Time Sampler	$\mathcal{U}([0, 1])$
	ODE Solver	Euler
	ODE $\mathrm{d}t$ (train) / steps	0.1 / 10
	ODE $\mathrm{d}t$ (eval) / steps	0.05 / 20
Network (U-Net) (Ronneberger et al., 2015)	t-Positional Embedding Dim.	256
	t-Positional Embedding MLP	$(1024, 1024)$
	Hidden Activation	mish (Misra, 2019)
	Blocks per Stage	1
	Block Dimensions	$(1024, 1024, 1024)$
Conditional Encoder	Encoder MLP	$(1024, 1024, 1024)$
	Encoder Activation	mish (Misra, 2019)
	Conditioning Mixing	additive
Optimizer Adam (Kingma & Ba, 2015)	Learning Rate	10^{-4}
	Weight Decay	0
	Gradient Norm Clip	—
Training	Max Discount $\gamma_{\max}$	0.996
	Target Network EMA	5×10^{-4}
	Gradient Steps	3M
	Batch Size	256
	Context Drop Probability ($z = \varnothing$)	0.1
	Consistency Proportion	$\begin{cases} 0.25 & \text{for ANTMAZE} \\ 0.15 & \text{for CUBE} \end{cases}$
GCRL Goal Sampling	p(trajectory goal)	0.5
	p(random goal)	0.5
	Geometric Trajectory Discount	0.995

F.3 PLANNING HYPERPARAMETERS

Table 12: COMPPLAN hyperparameters for the main results of the paper.

Method	Domain	Candidates	Effective horizons	Proposal Distribution	Eval samples	Replan every	Discount
CRL	ANTMAZE-MEDIUM	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-LARGE	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-GIANT	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	CUBE-1	1024	$[20, 80]$	Unconditional	128	1	0.99
	CUBE-2	1024	$[20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-3	1024	$[20, 20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-4	1024	$[20, 20, 20, 20, 80]$	Unconditional	128	1	0.99
GC-TD3	ANTMAZE-MEDIUM	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-LARGE	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-GIANT	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	CUBE-1	1024	$[20, 80]$	Unconditional	128	1	0.99
	CUBE-2	1024	$[20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-3	1024	$[20, 20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-4	1024	$[20, 20, 20, 20, 80]$	Unconditional	128	1	0.99
GC-BC	ANTMAZE-MEDIUM	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-LARGE	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-GIANT	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	CUBE-1	1024	$[20, 80]$	Unconditional	128	1	0.99
	CUBE-2	1024	$[20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-3	1024	$[20, 20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-4	1024	$[20, 20, 20, 20, 80]$	Unconditional	128	1	0.99
GC-1S	ANTMAZE-MEDIUM	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-LARGE	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	ANTMAZE-GIANT	256	$[50, 50, 100, 100, 200]$	Conditional	256	1	0.999
	CUBE-1	1024	$[20, 80]$	Unconditional	128	1	0.99
	CUBE-2	1024	$[20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-3	1024	$[20, 20, 20, 80]$	Unconditional	128	1	0.99
	CUBE-4	1024	$[20, 20, 20, 20, 80]$	Unconditional	128	1	0.99
HFBC	ANTMAZE-MEDIUM	32	$[25] * 24$	Conditional	128	1	0.999
	ANTMAZE-LARGE	32	$[25] * 24$	Conditional	128	1	0.999
	ANTMAZE-GIANT	32	$[25] * 24$	Conditional	128	1	0.999
	CUBE-1	32	$[25] * 4$	Unconditional	128	1	0.99
	CUBE-2	32	$[25] * 5$	Unconditional	128	1	0.99
	CUBE-3	32	$[25] * 6$	Unconditional	128	1	0.99
	CUBE-4	32	$[25] * 7$	Unconditional	128	1	0.99