Learning Temporally Abstract World Models without Online Experimentation

Benjamin Freed [1] **Siddarth Venkatraman** [1] **Guillaume Sartoretti** [2] **Jeff Schneider** [1] **Howie Choset** [1]

Abstract

Agents that can build temporally abstract representations of their environment are better able to understand their world and make plans on extended time scales, with limited computational power and modeling capacity. However, existing methods for automatically learning temporally abstract world models usually require millions of online environmental interactions and incentivize agents to reach every accessible environmental state, which is infeasible for most real-world robots both in terms of data efficiency and hardware safety. In this paper, we present an approach for simultaneously learning sets of skills and temporally abstract, skill-conditioned world models purely from offline data, enabling agents to perform zero-shot online planning of skill sequences for new tasks. We show that our approach performs comparably to or better than a wide array of state-of-the-art offline RL algorithms on a number of simulated robotics locomotion and manipulation benchmarks, while offering a higher degree of adaptability to new goals. Finally, we show that our approach offers a much higher degree of robustness to perturbations in environmental dynamics, compared to policy-based methods.

1. Introduction

Reinforcement learning, despite its success on many complex sequential decision-making problems, still struggles with long-horizon tasks. As a potential solution, recent deep RL methods have focused on *temporal abstraction*, wherein the agent's original action space is replaced by a set of temporally extended behavioral primitives, *i.e.*, *skills*, which allow the agent to make decisions on a coarser time granularity (Sutton et al., 1999; Ajay et al., 2020; Pertsch et al., 2021; Sharma et al., 2020b;a; Achiam et al., 2018; Baumli et al., 2021; Eysenbach et al., 2018; Shankar & Gupta, 2020). Most approaches to skill learning are based on model-free reinforcement learning, in which a skill-selection policy is directly learned from data, without explicitly forming a model of how skills influence the state of the environment (Ajay et al., 2020; Pertsch et al., 2021; Zhou et al., 2021; Achiam et al., 2018; Baumli et al., 2021; Eysenbach et al., 2018; Shankar & Gupta, 2020). However, without an explicit model of the effect of skills, there is no straightforward way for agents to rapidly transfer their policy to new tasks, without a lengthy retraining process. On the other hand, existing model-based approaches to temporal abstraction require skills to be run repeatedly online in the environment (Sharma et al., 2020b;a). This lengthy trial-and-error process is particularly ill-suited to learning in physical environments, where training online on hardware may result in unreasonable wear and tear, and can be hazardous. In this work, we introduce (OPOSM), a novel approach to simultaneously decompose a given dataset of demonstrations into a space of parameterized skills, while also extracting a temporally abstract world model that enables online planning of skill sequences for downstream tasks without retraining (Fig. 1). OPOSM outperforms or performs comparably to all relevant model-based and model-free offline RL baselines on a number of standard benchmarks, including simulated robot navigation and manipulation tasks, while allowing zero-shot transfer to new goals. Finally, we show that our iterative, model-based re-planning approach provides natural robustness to perturbations in environmental dynamics, which to our knowledge is unique in the space of offline RL algorithms.

Similar to our approach, previous model-free offline skill learning algorithms, *e.g.,* OPAL (Ajay et al., 2020), PLAS (Zhou et al., 2021), and SPiRL (Pertsch et al., 2021), extract a set of skills from an offline dataset using a variational inference (VI)-based approach (Blei et al., 2017). In all three approaches, skills are treated as latent variables, and an evidence lower bound (ELBO) is maximized to learn a skill-conditioned low-level policy, describing the mapping from skills (and possibly states) to actions. Unlike our approach, in these offline model-free skill learning algorithms, a policy was subsequently learned to select among skills given the state of the environment using a model-

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA [2]Mechanical Engineering Department, National University of Singapore, Singapore. Correspondence to: Benjamin Freed <bfreed@cs.cmu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

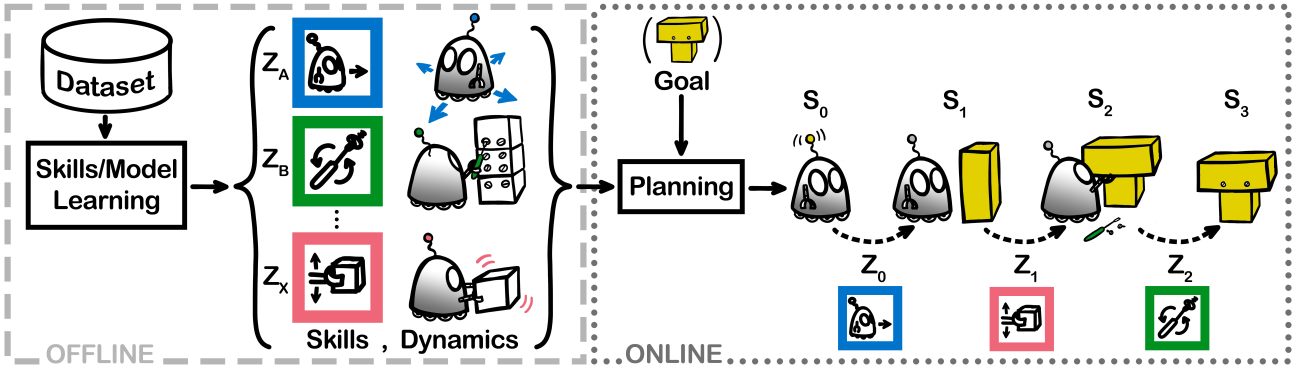


Figure 1. **Overview of Online Planning with Offline Skill Models (OPOSM).** Our approach simultaneously extracts a set of skills, and a skill-conditioned temporally abstract world model (TAWM) from purely offline data. The TAWM predicts the state transitions resulting from executing a given skill, enabling the agent to extend its effective planning horizon by planning sequences of temporally extended skills, rather than base-level actions.

free RL algorithm. This policy is suitable for one task, and requires re-training to handle new tasks. This work also shares similarities with online model-based skill learning approaches, namely DADS (Sharma et al., 2020b) and Off-DADS (Sharma et al., 2020a), which simultaneously extract a set of skills *and* a dynamics model that predicts the long-term state transition, resulting from the execution of a particular skill. This dynamics model was then used for zero-shot planning on downstream tasks. Unlike our approach, the model was built by repeatedly running skills online in the environment, and training a model to predict the long-term state transitions they induced. The set of skills was trained to enable the agent to reach the most diverse set of states possible.

Our proposed approach, which we call online planning with offline skill Models (OPOSM), simultaneously extracts a set of skills, and a skill-conditioned temporally abstract world model (TAWM), purely from offline data. The primary challenge that our approach overcomes is in learning a *causal* skill-conditioned world model, when online experimentation is not possible. In the offline setting, deducing the true causal effect of skills on the state of the environment is challenging because one cannot repeatedly deploy skills in the environment to observe their effect (Pearl, 2009). In fact, we show that a naive VI-based approach similar to that taken by OPAL, PLAS, and SPiRL is not guaranteed to correctly model these causal relationships, instead leading the agent to overestimate its influence on the environmental state. We propose a principled method to fix this issue with a modified VI approach, in which the approximate posterior is constrained to match the true causal structure of the environment. We show that the performance of OPOSM far exceeds that of a standard model-based planning approach, while making planning up to 60x faster, on a number of challenging long-horizon D4RL benchmarks. Additionally, we show that we perform comparably to or outperform a wide

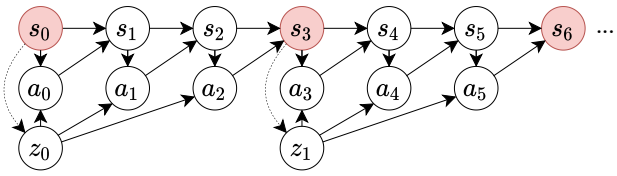


Figure 2. **Environmental Causal Structure.**. State transitions depend on the previous state and action, according to the state-transition distribution $p(s_{t+1}|s_t, a_t)$. Actions depend on the current state, and the current skill z that is being executed for a given time horizon, according to the skill-conditioned low-level policy $\pi_\theta(a_t|s_t, z)$. Dotted arrows indicate that the skill selected at a given timestep may depend on the state at that timestep. States in which skills begin/terminate execution are highlighted in red; these are also the states predicted by the temporally abstract dynamics model.

range of offline RL algorithms on these D4RL benchmarks. While typical offline RL algorithms essentially "overfit" to one specific goal on D4RL tasks, we show that OPOSM is able to solve D4RL tasks with *randomly selected* goals, without any retraining. Finally, we show that OPOSM provides a far higher degree of robustness to perturbations in environmental dynamics compared to a model-free expert, indicating that our approach is better able to generalize to novel environments.

2. Model-Based Temporal Abstraction

The primary contributions of this work are 1) an approach to simultaneously learning a skill-conditioned low-level policy *and* a skill-conditioned temporally abstract world model (TAWM) purely from offline data, and 2) the use of this TAWM for zero-shot planning on a downstream task.

Similar to past works, skills are represented by a *skill-conditioned low-level policy* $\pi_\theta(a_t|s_t, z)$, parameterized by θ, where $a_t \in \mathcal{A}$ is the current action selected by the agent, $s_t \in \mathcal{S}$ is the current state, and each $z \in \mathcal{Z}$ is an

abstract *skill variable* that encodes a specific skill. We assume skills execute for a fixed time horizon H; however, in principle these fixed termination conditions can be replaced with learned termination conditions, similar to the approach taken by Shankar & Gupta (2020). The induced causal structure of the environment is depicted in Fig. 2.

In addition to learning a set of skills, we also seek to learn a *skill-conditioned temporally abstract world model* (TAWM) $p_\psi(s'|s,z)$, parameterized by ψ, which models the distribution of states the agent will find itself in upon termination of the current skill z, given that it began executing z in state s. We do not train on rewards; instead, the TAWM is used downstream for planning, for an arbitrary reward function provided at execution time.

2.1. Learning Causal Temporally Abstract World Models from Offline Data

The primary challenge of learning a TAWM from offline data is ensuring that the TAWM accurately captures the true causal influence of skills on long-term state transitions. A naive approach to learning π_θ and p_ψ would be to treat skills as latent variables, and optimize the following evidence lower bound (ELBO), derived in Sec. A of the appendix:

$$\mathcal{L}(\theta,\psi,\phi,\omega) = \mathbb{E}_{\tau_T\sim\mathcal{D}}\Big[\mathbb{E}_{q_\phi(z|\tau_T)}\big[\log \pi_\theta(\vec{a}|\vec{s},z) + \log p_\psi(s_T|s_0,z)\big] - D_{KL}(q_\phi(z|\tau_T)||p_\omega(z|s_0))\Big], \tag{1}$$

where τ_T is an T-length sub-trajectory uniformly sampled from the offline dataset $\mathcal{D}$, $\vec{s}$ and $\vec{a}$ are the state and action sequences that comprise τ_T, q_ϕ represents our posterior over z given τ_T and parameterized by ϕ, and p_ω represents our prior over z, given s_0, and parameterized by ω. In this work, we take T to be a fixed hyperperameter that is shorter than the overall episode length, as has been done in most related works (Ajay et al., 2020; Pertsch et al., 2021; Eysenbach et al., 2018; Sharma et al., 2020b; Baumli et al., 2021; Achiam et al., 2018). The corresponding architecture is shown in Fig. 3, and additional details are provided in the supplemental material. The first two terms in Eq. (1) correspond to the log-likelihood of demonstrator actions and long-term state-transitions, respectively. The final term represents the KL divergence between our skill posterior and prior, and encourages learning a compressed representation of skills. The ELBO in Eq. (1) is similar to those used by Ajay et al. (2020); Pertsch et al. (2021); Shankar & Gupta (2020); Zhou et al. (2021) for offline skill learning, except that it contains a term corresponding to the log likelihood of long time-horizon state transitions according to p_ψ. However, naively optimizing $\mathcal{L}$ with respect to θ,ψ,ϕ, and ω jointly is not guaranteed to learn the correct *causal* relationship between skills and long-term state transitions. Instead, our analysis indicates that the model may undesirably learn to use the latent variable (z) to explain variations in the long-term state transitions that are not under the control of the agent. We demonstrate this phenomenon in the supplemental material with a simple example, and connect it to causal theory (Pearl, 2009). We show that this issue can cause the agent to be overly confident when predicting state transitions, which in turn leads to overestimation of reward and suboptimal decision-making.

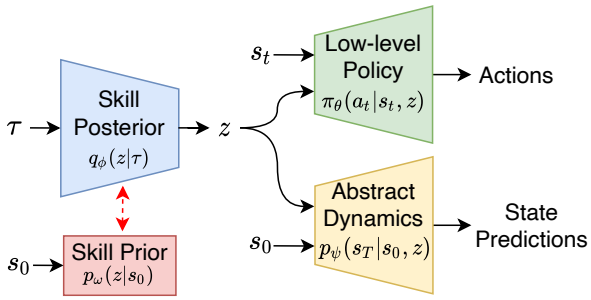


Figure 3. **Skill Model Architecture**. During offline training, sub-trajectories τ_T are sampled from the dataset and fed into a skill posterior q_ϕ, which learns to infer the skill z. This z is then used to condition the predictions of the low-level skill-conditioned policy π_θ, and the temporally abstract dynamics p_ψ. The skill prior p_ω infers z from the initial state of the sub-trajectory only.

The fundamental issue with the naive VI approach described above is that the approximate posterior over skills q_ϕ does not match the true causal structure of the environment, which is depicted in Fig. 2. Based on this structure, for a given low-level policy and skill prior, the *true* posterior over skills is given by

$$p(z|\vec{s},\vec{a}) = \frac{1}{\eta}\pi_\theta(\vec{a}|\vec{s},z)p_\omega(z|s_0), \tag{2}$$

which we derive in the supplemental material, where η is a normalization constant. We therefore propose the following incremental EM-style algorithm, which trains q_ϕ to match the true posterior, for learning *causal* TAWMs from offline data:

E Step: ϕ is updated with a gradient descent step so as to minimize the expected KL divergence between q_ϕ and the true posterior, which is equivalent to minimizing $\mathbb{E}_{\tau_T\sim\mathcal{D}}\left[\mathbb{E}_{z\sim q_\phi}\left[\log \frac{q_\phi(z|\vec{s},\vec{a})}{\pi_\theta(\vec{a}|\vec{s},z)p_\omega(z|s_0)}\right]\right]$.

M Step: θ, ψ, and ω are updated with a gradient ascent step to maximize the ELBO from Eq. (1).

Due to the tightness of the ELBO, and because the approximate posterior is now trained to match the true posterior, using this EM style approach should result in a TAWM that correctly captures the causal influence of skills on state transitions.

3. Planning for Downstream Tasks Using Temporally Abstract World Models

Once a world model has been learned and a reward function has been specified for a given task, our agent can plan a sequence of skills to maximize its predicted cumulative reward. Unlike past offline RL approaches (Kidambi et al., 2020; Yu et al., 2020), we found it unnecessary to constrain the agent behavior to be close to the dataset distribution, because our agent plans over behaviors that are represented in the dataset (Zhou et al., 2021). We use a cross entropy method (CEM) planner (Rubinstein, 1999) to optimize the skill sequence. However, rather than directly optimizing a sequence of z vectors, we plan in a "whitened" version of $\mathcal{Z}$-space. Specifically, we optimize a sequence of ϵ *vectors*, where the ith element of the plan $\epsilon_i \in \mathcal{E}$ is related to z_i according to a shifting and scaling by the learned skill prior mean $\mu_0(s_i)$ and standard deviation $\sigma_0(s_i)$, *i.e.*, $z_i = \mu_0(s_i) + \sigma_0(s_i) \cdot \epsilon_i$.

Planning in $\mathcal{E}$-space essentially allows the agent to search in the space of *normalized biases* away from the demonstration policy's mean action. This provides the agent with a well-conditioned search space, where reasonable plan values lie within a (roughly-)unit ball around the origin, regardless of the state, rather than having different state-dependent means and scales. Additionally, planning in $\mathcal{E}$-space allows us to easily warm-start the planning procedure, since our initial plan can easily be sampled from the demonstrator policy, i.e., $\epsilon_0, ..., \epsilon_K \sim \mathcal{N}(\vec{0}, I)$.

We take a model predictive control approach, wherein the agent iteratively executes the first skill in the plan, and subsequently re-optimizes a new plan. This iterative re-planning improves robustness to errors in its world model. The agent executes each skill in its plan by running the low-level policy $\pi(a_t|s_t, z) = \mathcal{N}(\mu_a(s_t, z), \sigma_a(s_t, z))$, where s_t and a_t refer to the current state and action respectively, and z refers to the current skill.

4. Experimental Results

4.1. Visualization of Model Predictions

We hypothesize that our temporally abstract world model should be more accurate over long time horizons, compared to sequential predictions using a single-timestep model. We assess this qualitatively in the antmaze-large environment by visualizing the set of states that the agent predicts it will visit, given an optimized skill sequence, according to both our TAWM and a single-timestep dynamics model (Fig. 4). We also visualize the distribution of trajectories actually followed by the agent, during repeated executions of the plan in the real environment, without re-planning. We notice that in most cases, the predictions made by the TAWM agree well with the true trajectories, as indicated by the fact that the true trajectories typically intersect the high-density regions predicted by the TAWM. This enables the agent to reach a location close to the goal, despite not doing any re-planning. On the other hand, the low-level dynamics predictions tend to suffer from compounding error and diverge rapidly from the true trajectories. This often leads to the low-level dynamics predicting that the agent will travel through walls, which is physically impossible.

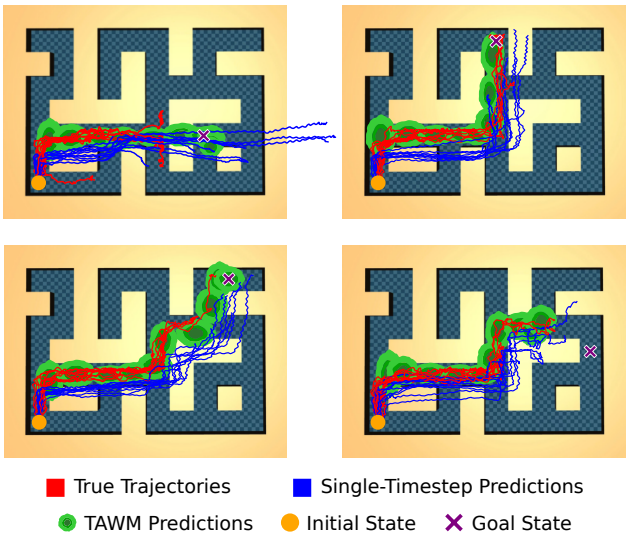


Figure 4. **Visual Comparison of Temporally Abstract and Single-Timestep Model Prediction Accuracy in Antmaze-large Environment**. The planner optimizes a skill sequence using the TAWM. We then repeatedly execute the plan, and visualize the resulting set of trajectories (shown above in red). We also visualize the predictions made by the TAWM (green), and the single-timestep low-level dynamics model (blue). We find that the predictions made by the TAWM agree well with the true trajectories, as they typically intersect the high-density regions predicted by the TAWM. On the other hand, the low-level dynamics predictions tend to suffer from compounding errors and rapidly diverge from the true trajectories.

4.2. D4RL Benchmark Tasks

We compare the performance of OPOSM with that of several other offline RL algorithms (Ajay et al., 2020; Kumar et al., 2019; Ghasemipour et al., 2021; Kumar et al., 2020; Zhou et al., 2021; Sharma et al., 2020b; Yu et al., 2021; Sharma et al., 2020a) on multiple tasks from the D4RL benchmark suite (Fu et al., 2020). The algorithms we compare to are summarized below:

Conservative Q-Learning+OPAL (CQL+OPAL): Proposed in (Ajay et al., 2020), CQL+OPAL is a skill-based model-free offline RL algorithm. Similar to OPOSM, CQL+OPAL extracts temporally extended skills from an offline dataset. Unlike OPOSM, CQL+OPAL uses CQL

to learn a high-level policy to select skills given the current state, rather than using a dynamics model for online planning.

Offline Behavior Cloning+OPAL (BC+OPAL): Similar to CQL+OPAL, BC+OPAL starts by extracting temporally extended skills from an offline dataset. Behavior cloning is them used to learn a high-level skill policy, rather than CQL. This is done by partitioning the offline dataset into successful and unsuccessful episodes, and training a high-level policy to predict the skills associated with subtrajectories from successful episodes. This algorithm is very similar to the BC+OPAL algorithm proposed in (Ajay et al., 2020), although we do not provide additional examples of successful trajectories, but instead extract successful trajectories from the offline dataset.

Behavior Cloning (BC): A policy is trained to predict the actions conditioned on states from successful episodes (Pomerleau, 1988).

Bootstrapping Error Accumulation Reduction (BEAR): introduced in (Kumar et al., 2019), BEAR is a Q-learning algorithm that attempts to reduce the bootstrapping error from actions that lie outside of the training data distribution. BEAR reduces bootstrapping error by incorporating a *distribution-constrained backup operator*, which constrains the policy search to policies with the same support as the offline dataset (Kumar et al., 2019).

Conservative Q-Learning (CQL) : introduced in (Kumar et al., 2020), CQL mitigates the problem of overestimation of values induced by the distributional shift between the dataset and the learned policy, common in off-policy RL methods. CQL attempts to overcome value overestimation error by learning a conservative Q-function that lower-bounds the true values.

Expected-Max Q-Learning (EMaQ): introduced in (Ghasemipour et al., 2021), EMaQ derives a novel backup operator that smoothly interpolates between performing policy evaluation on the data-collection policy, and a typical Q-learning backup. EMaQ explicitly considers the support of the offline dataset, and reduces the number of function approximators and hyperparameters necessary compared to previous related approaches.

CQL+Off-DADS: An offline variant of the off-policy dynamics-aware discovery of skills (off-DADS) (Sharma et al., 2020a) algorithm, used as a baseline for comparison in (Ajay et al., 2020). Similar to OPOSM, CQL+Off-DADS extracts temporally-extended skills from an offline dataset. It does so by learning a latent representation of behaviors (*i.e.,* the skills) that maximizes mutual information between skills and trajectories. CQL+Off-DADS differs from OPOSM in that it uses a model-free offline RL algorithm (CQL) to learn a high-level policy over skills, whereas OPOSM performs online planning over skills.

Policy in Latent Action Space (PLAS): Similar to OPOSM, PLAS (Zhou et al., 2021) learns a latent space that models the distribution of actions seen in the offline dataset, similar to our skill space. Unlike OPOSM, PLAS does not perform temporal abstraction (skills are single-timestep behaviors); nor does PLAS learn a dynamics model or perform planning. Instead, PLAS learns a policy over latent actions using Q-learning.

Conservative Offline Model-Based Policy Optimization (COMBO): proposed in (Yu et al., 2021), COMBO is an offline model-based RL algorithm that computes conservative value estimates for state-action pairs that are out of the support of the dataset. COMBO accomplishes this by training a value function using both the offline dataset, and simulated rollouts from a model, while additionally regularizing the value function on out-of-support state-actions generated by the model. Similar to OPOSM, COMBO is model-based; however, COMBO does not perform temporal abstraction or online planning.

Low-Level Planning (LLP): a simple model-based planning approach, in which a traditional single-timestep action-conditioned model is learned and used to plan sequences of actions, while constraining actions to be similar to an imitation-learned policy.

We chose to test our approach on the maze2d, antmaze, and kitchen tasks due to their long horizons, which causes traditional model-based planning to fail, as demonstrated by our results. Additionally, these tasks seem to possess a natural hierarchy wherein low-level primitives (such as navigating to a specific location in the case of maze2d and antmaze, or picking and placing objects in kitchen) can be identified and leveraged to solve the task, making them an ideal testbed for temporal abstraction. In all experiments, independent episodes of OPOSM and LLP were run until confidence intervals were smaller than 10 %, typically requiring 250-300 episodes for Maze and Antmaze tasks, and 50-100 episodes for Kitchen tasks.

Maze2d: We test on two variants of the maze2D task: medium and large. In both variants, a point-mass agent moves around a 2-dimensional maze. The goal of the task is for the agent to navigate from one corner of the maze to the opposite corner. The size of the maze differs between the two environments, with large using the larger of the two. In both cases, our agent was trained on the corresponding task dataset provided by D4RL, which contain a single long

state-action sequence of the agent moving about the maze, towards randomly-chosen waypoints. An episode is considered a success if the agent comes within a 0.5-unit radius of the goal. The cost assigned to a predicted trajectory during planning is the negative $\mathcal{L}_2$ distance between the goal and the nearest predicted state to the goal. Rewards are therefore sparse in the sense that an entire trajectory is assigned one scalar reward value; no per-timestep rewards are used. We find that both OPOSM and LLP are able to perfectly solve both tasks; however, OPOSM requires up to 60x less time to find a plan[1], requiring on average only 0.051 ± 0.0014s, while LLP required 3.02 ± 0.245s. Visualizations of the plans generated by OPOSM and LLP are included in the supplemental material. We leave it to future work to determine what fraction of this speed increase is due to the decreased search-space size enabled by temporal abstraction, versus planning in the better-conditioned $\mathcal{E}$-space.

Antmaze: a Mujoco-simulated quadruped robot (the "ant") is tasked with navigating from one corner to the opposite corner of a fixed, 2-dimensional maze. Antmaze is a nice testbed for skill learning and temporal abstraction because the problem offers a natural control hierarchy, wherein skills corresponding to low-level locomotion controllers, e.g., walking in a straight line or turning around a corner, can be sequenced to solve a higher-level navigational task, e.g., navigate to the upper right-hand corner of the maze. Antmaze is challenging because in addition to the nontrivial task of learning to locomote from offline data, the agent must also string together these locomotion behaviors over hundreds of timesteps to navigate the maze. We consider two variants of the antmaze environment: antmaze-medium-diverse and antmaze-large-diverse, which differ in maze size. Similar to the maze2d environments, the antmaze environments are associated with a dataset of the ant robot moving about the maze in an undirected fashion. Our results are reported in Table 1. Boxes filled with "-" indicate experiments we were unable to run due to lack of publically available code. To the best of our knowledge, OPOSM is the first model-based offline RL algorithm to achieve this level of performance in the antmaze environments. We find that our approach performs very similarly to OPAL, and exceeds the performance of the other baselines we consider by a wide margin, including an offline version of DADS (CQL+Off-DADS), described in (Ajay et al., 2020).

Kitchen: a robotic manipulation task in which a simulated Franka robotic arm interacts with items in a kitchen. The goal of the task is to place the items in a specific configuration. We train our skill model on three different datasets, referred to as *mixed*, *partial*, and *complete*. The mixed dataset consists of non-task-directed demonstrations, and is typically considered the hardest to learn from. The partial dataset consists of partially task-directed demonstrations, while the complete dataset contains fully task-directed demonstrations.

Our results are reported in Table 2. To the best of our knowledge, OPOSM is the first model-based offline RL algorithm to perform well across all three variants. OPOSM also significantly outperforms PLAS (Zhou et al., 2021), which is similar to our approach in the sense that it extracts skills from an offline dataset using VI. However, PLAS does not use a temporally abstract world model for planning, highlighting how this contribution yields significant improvement over a highly-related approach. Additionally, OPOSM achieves comparable rewards to SPiRL [2], despite the fact that SPiRL has the benefit of training online in addition to using offline data.

On the other hand, we find that OPOSM is outperformed by EMaQ and CQL+OPAL on the kitchen-mixed and kitchen-partial tasks. Further work is required to determine why this is the case. Surprisingly, OPOSM is outperformed by LLP on the kitchen-complete task, but outperforms the other baselines, suggesting that perhaps temporal abstraction is not necessary for this particular task.

4.3. Comparison to Naive Variational Inference

In Sec. 2.1, we argued that a naive VI approach may learn inaccurate skill models, because the model may predict that skills have an erroneously high degree of influence over state transitions. To determine whether this issue degrades performance in online planning, we compare planning success rates using the naive VI method to our proposed EM algorithm in the antmaze-medium and -large tasks (Table 3). The results agree with our analysis, as we find that our proposed EM algorithm outperforms the naive VI method on both tasks.

4.4. D4RL Tasks with Randomized Goals

In the standard usage of the maze2d, antmaze, and kitchen tasks, the goals for each of these tasks are fixed across episodes, or vary only very slightly, such that the same learned policy can be effective for every episode. However, this is unrealistic for many real-world robotic tasks, where goals may change rapidly; for example, in an autonomous vehicle or warehouse automation scenario, where a robot may be required to navigate to arbitrary locations which it has never visited before. One of the strengths of OPOSM is that our TAWM is not specific to any particular reward function, allowing it to handle a near-arbitrary range of goals,

[1] Planning in both OPOSM and LLP was terminated when the predicted plan reward exceeded a threshold.

[2] The results for SPiRL are not shown in this table because exact numeric results were not available. Additionally, it is an online algorithm and therefore not directly comparable to OPOSM.

Table 1. **Success Rates for Antmaze Environments**. Success rate is measured as the fraction of episodes in which the agent reaches its goal. OPOSM performs comparably to CQL+OPAL, while outperforming all other baseline algorithms.

Task	**OPOSM**	LLP	CQL+OPAL	BC+OPAL	BC	BEAR	EMaQ	CQL	CQL+Off-DADS	COMBO
Medium	$\mathbf{78.29 \pm 4.32}$	31.2 ± 5.74	$\mathbf{81.1 \pm 3.1}$	24.0 ± 4.8	0.0	8.0	0.0	53.7 ± 6.1	59.6 ± 2.9	17.3 ± 4.3
Large	$\mathbf{70.2 \pm 4.6}$	14.4 ± 4.35	$\mathbf{70.3 \pm 2.9}$	53.0 ± 5.65	0.0	0.0	0.0	14.9 ± 3.2	-	5.0 ± 2.5

Table 2. **Average Rewards for Kitchen Environments.** In kitchen-complete, OPOSM outperforms all model-free baselines, and is outperformed only by low-level planning. In kitchen-mixed and kitchen-partial, OPOSM is outperformed by CQL+OPAL and EMaQ, and outperforms the other baselines.

Task	**OPOSM**	LLP	CQL+OPAL	BC	BEAR	EMaQ	CQL	PLAS	COMBO
Mixed	54.5 ± 4.125	46.0 ± 3.8	$\mathbf{69.3 \pm 2.7}$	47.5	47.2	$\mathbf{70.8 \pm 2.3}$	52.4 ± 2.5	40	2.3 ± 0.82
Partial	65.25 ± 2.85	39.00 ± 4.57	$\mathbf{80.2 \pm 2.4}$	33.8	13.1	74.6 ± 0.6	50.1 ± 1.0	45	24.3 ± 2.04
Complete	47.325 ± 2.725	$\mathbf{66.68 \pm 5.03}$	-	27.2 ± 3.2	0.0	36.9 ± 3.7	43.8	34.8	0.0 ± 0.0

Table 3. **Comparison of Proposed EM Skill Learning to Naive VI.** We report success rate in antmaze, and reward in kitchen environments. We find that, in agreement with our analysis, our proposed EM skill learning algorithm outperforms a naive VI-based approach.

	EM (ours)	Naive VI
Antmaze-medium	$\mathbf{78.29 \pm 4.32}$	$\mathbf{74.4 \pm 4.4}$
Antmaze-large	$\mathbf{70.2 \pm 4.6}$	58.3 ± 5.6
Kitchen-partial	$\mathbf{65.25 \pm 2.85}$	21.4 ± 5.7
Kitchen-complete	$\mathbf{47.325 \pm 2.725}$	2.0 ± 2.725
Kitchen-mixed	$\mathbf{54.5 \pm 4.125}$	$\mathbf{51.4 \pm 5.7}$

while model-free methods require re-training to adapt to new goals (reward functions). While multi-task learning may enable existing model-free methods to deal with changing goals, this approach is still limited in that it requires the algorithm designers to prescribe in advance the compete set of tasks for which the policy will be trained. Multi-task learning typically only enables generalization to tasks within this pre-specified set, and likely requires larger-capacity policies for larger task distributions. We believe that a model-based approach is a far more natural way to flexibly and rapidly transfer knowledge from past experience to new tasks while keeping the model size relatively small.

For this reason, we test our approach on a new variant of the D4RL tasks that we consider above, by randomizing the goal at the beginning of each episode. Specifically, for the maze2d and antmaze tasks, a goal location is selected uniformly at random from the set of locations present in the training set. In the case of the kitchen environments, we randomly select a set of 4 objects to reconfigure out of all of the 7 objects present in the environment. We compare OPOSM to LLP, as well as a goal-conditioned variant of OPAL with behavior cloning (GC-BC+OPAL). Specifically, we randomly sampled many 2D goal locations from the set of locations visited in the dataset, and learned a policy over OPAL skills by performing behavior cloning on sub-trajectories from the dataset that were able to reach these goals. We did not compare to a multi-goal version

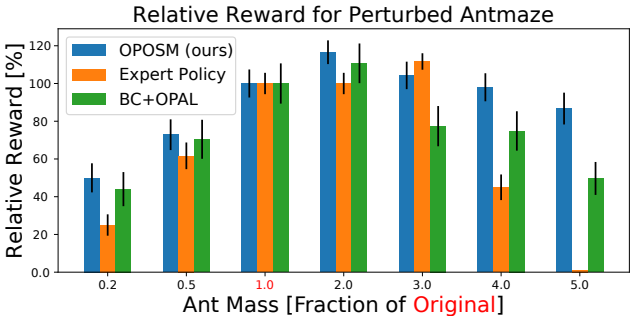


Figure 5. **Robustness to Perturbed Dynamics**. We evaluate the robustness of OPOSM, BC+OPAL, and a model-free expert policy to changes in robot mass in the antmaze-large environment. Plotted above is relative change in success rate, compared to the original environment, as robot mass is varied. We find that the performance of OPOSM diminishes far less compared to the baselines, for masses other than the original, indicating that OPOSM is more robust to changes in environmental dynamics.

of CQL+OPAL because we were unable to reproduce the reproduce the results obtained reported in (Ajay et al., 2020) or to obtain a working implementation from the authors. We report our results in Table 4. In general, we find that the performance of OPOSM degrades very little on random goal tasks, compared to its single-goal performance (reported in tables 1 and 2). On the other hand, GC-BC+OPAL suffers a large degradation in performance, compared to its single-goal counterpart. These findings indicate that the zero-shot planning capabilities OPOSM provides a more natural mechanism for transferring skills across tasks, compared to policy-based approaches.

4.5. Robustness to Perturbed Environmental Dynamics

We hypothesize that OPOSM's iterative replanning approach may provide greater robustness, compared to approaches that deploy a static policy, because iterative replanning allows the agent to compute plans that are optimal (according to the learned dynamics model) from *any* state within the support of the dataset. While it is true that plans will be suboptimal if the dynamics model is inaccurate, at least planning *attempts* to generate a high-performing plan

Table 4. **Results on D4RL Tasks with Random Goals**. Unlike model-free RL algorithms, OPOSM enables zero-shot model-based planning for new tasks. To demonstrate this capability, we test OPOSM and our low-level planning (LLP) baseline on variants of D4RL tasks with randomized goals. Our results show that OPOSM retains high performance, and continues to outperform LLP.

Task	**OPOSM (Ours)**	LLP	GC-BC+OPAL
Maze2d-medium	$\mathbf{100 \pm 0}$	95.67 ± 2.55	-
Maze2d-large	$\mathbf{99.4 \pm 1.1}$	86.67 ± 4.97	-
Antmaze-medium	$\mathbf{72.6 \pm 4.7}$	32.8 ± 5.82	52.0 ± 5.67
Antmaze-large	$\mathbf{65.07 \pm 4.96}$	30.0 ± 5.68	36.33 ± 5.44
Kitchen-partial	$\mathbf{56.25 \pm 4.90}$	43.07 ± 4.50	-
Kitchen-mixed	$\mathbf{53.125 \pm 4.00}$	35.5 ± 4.475	-

from any state the agent may find itself in. Past work has shown that iterative replanning can impart a high degree of robustness to changes in environmental dynamics (Pereira et al., 2018). This is in contrast to a static policy, which has no guarantee of optimality outside of its (potentially very narrow) state visitation distribution, and in practice tend to generalize poorly to unseen states or dynamics (Cobbe et al., 2019). We therefore hypothesize that OPOSM may be better equipped to deal with perturbations in the dynamics of the environment, which will change the state visitation distribution of an optimal policy.

To test this hypothesis, we run our approach on a range of modified versions of the antmaze-large-diverse environment, in which the mass of the agent is varied greatly. No attempt is made to adapt our agent to these differing environmental dynamics; we use the same model on all variants, trained on the original unaltered dataset provided by D4RL. We compare our approach to an expert model-free policy, taken from the Deep Offline Policy Evaluation benchmark suite (Fu et al., 2021). We found that this policy outperformed all the offline approaches we considered on the unperturbed antmaze-large task. We find that our method is far less sensitive to changes in the mass of the agent, and still achieves a relatively high success rate, even when the mass of the agent is increased by a factor of 5, for which the expert policy fails completely (Fig. 5). Typical offline RL algorithms struggle with subtle shifts in state-visitation distribution between offline datasets and online execution, even when executed in environments with the same dynamics as they were trained on. To the best of our knowledge, this is the first offline RL algorithm to demonstrate a high degree of natural robustness to changes in environmental dynamics between training dataset and online execution.

5. Related Work

OPOSM builds on top of the *options* framework proposed by Sutton et al. (1999). The options framework mathematically formalizes the notion of temporal abstraction in MDPs, by extending the action space to include *options*, which are essentially skills in our terminology. An option is composed of a policy, a termination condition, and an initiation set that defines what state an option can be initiated from. A set of options defined over an MDP constitute a semi-MDP, wherein actions take variable lengths of time. Similar to many past works (Ajay et al., 2020; Sharma et al., 2020b;a; Pertsch et al., 2021; Baumli et al., 2021; Achiam et al., 2018; Eysenbach et al., 2018), our work can be viewed as an options discovery algorithm, wherein the low-level policy conditioned on a particular z can be viewed as the option policy, the skill prior can be viewed as a soft initiation set, and options have a fixed-length termination condition. In addition to discovering a set of options, our method also uses the TAWM to represent the dynamics of the semi-MDP induced by those options, purely from offline data.

Of the many existing learning-based approaches to options discovery, the approaches most relevant to our work are the so-called *variational options discovery* algorithms, which treat skills as latent variables, and attempt to learn skills by maximizing a variational objective. These algorithms include Diversity Is All You Need (Eysenbach et al., 2018), Variational Intrinsic Control (VIC) (Baumli et al., 2021), Variational Autoencoding Learning of Options by Reinforcement (VALOR), Dynamics Aware Discovery of Skills (DADS, Off-DADS) (Sharma et al., 2020b;a), and Skill-Prior RL (SPiRL).

DIAYN, VIC, and VALOR all frame skill learning as an online, model-free RL problem, wherein the reward function encourages learning a diverse set of skills. These approaches then learn high-level policies that select among skills (rather than base-level actions), again using online model-free RL, to accomplish downstream tasks. Unlike these approaches, OPOSM requires no online interaction with the environment, and is model-based, thus allowing it to plan zero-shot for downstream tasks (provided a reward function) with no re-training.

DADS is an online, on-policy, model-based skill learning algorithm that uses repeated online trial and error to build up a set of skills, and a dynamics model over the skills, with a reward function that incentives a high mutual information between skills and future states. Similar to our approach, DADS uses its skill-conditioned dynamics model for zero-shot planning on downstream tasks. The primary difference between our work and DADS is that we do not require online interaction to discover skills or learn our dynamics model, which we instead learn from purely offline data. Off-DADS is similar to DADS, but makes use of off-policy (online) RL to improve data efficiency, and has been shown to effectively learn skills on real robots. Off-DADS still requires online interaction with the environment.

Similar to our approach, SPiRL uses offline data to learn a set of skills, and a prior over those skills. It then uses that skill prior to help guide an online RL algorithm to learn a

policy over skills for a downstream task. Unlike SPiRL, our approach requires no online training for downstream tasks.

Offline Primitives for Accelerating offline reinforcement Learning (OPAL), like our approach, extracts skills and learns how to use those skills to solve a task, purely from offline data. The principle difference between OPAL and OPOSM is that OPOSM is model-based, and relies on model-based planning to find sequences of skills, rather than using model-free RL to learn a policy over skills, to solve a downstream task. Therefore, OPOSM enables online zero-shot planning for *any* objective, whereas OPAL would require a lengthy retraining process should the downstream objective change.

This work was developed concurrently with Skill-Based Model-Based RL (SkiMo) (Shi et al., 2022), which, similar to our work, extracts skills and temporally abstract dynamics models from offline data. The primary difference between SkiMo and our work is that SkiMo uses skills strictly for online learning on downstream tasks, whereas we focus on offline learning. As we point out in Sec. 2.1, pure offline learning creates difficulties in inferring the causal effect of skills, which a naive VI approach fails to address. An additional difference between our work and SkiMo is that SkiMo's skills are not reactive to the current state, essentially making them action sequences rather than policies.

6. Limitations

One limitation of the current definition of OPOSM is our assumption of fixed-length skills. This is limiting because for many tasks, the most natural decomposition may result in subtasks (and thus skills) that require differing amounts of time to complete. Additionally, depending on random events may cause a subtask to take an unpredictable amount of time to complete. One potential solution would be to learn *termination conditions* for skills, similar to the approach taken by Shankar & Gupta (2020), thus allowing skills to execute until a natural stopping point is reached.

An additional limitation of our current approach is the assumption of a reward function that depends only on states in which skills begin or terminate. This assumption is made because these are the only states predicted by our TAWM. However, not all reward functions may be expressed this way. One potential solution is to learn an *abstract reward function*, which predicts the cumulative reward obtained during the execution of a skill. This way, cumulative episode reward can always be predicted, assuming the abstract reward function is well approximated.

Another limitation of OPOSM (and other offline skill-learning approaches) is that the quality of learned skills depends on the quality of the demonstrations provided. We currently have no way of biasing the learned skills towards *useful* skills (*e.g.*, skills that can guarantee a high reward on a particular task). A potential solution to this limitation would be to use an approach such as joint model-policy optimization (Eysenbach et al., 2021) instead of VI, wherein a lower bound on reward is maximized as opposed to a lower bound on log likelihood of the data, thus biasing the model to learn high-reward skills. However, this solution comes with a trade-off, as skills over-specialized to one task may not generalize as well to a new task.

One final limitation of our approach is that currently, we do nothing to explicitly constrain the skill plans generated by OPOSM to remain within the support of the dataset. This is a common feature of many offline RL algorithms, as it prevents the agent from straying into parts of the state-space where the dynamics are unknown. We hypothesize that this limitation contributes to the relative weakness of our approach on the kitchen-mixed and kitchen-partial tasks.

7. Conclusions and Future Work

We have proposed an approach for simultaneously learning a set of skills, and a temporally abstract world model capable of predicting the long-term state transitions caused by those skills, purely from offline data. Our approach exhibits high performance on a wide range of long time-horizon benchmarks, while enabling zero-shot generalization to new downstream tasks and robustness to changes in environmental dynamics. While our performance may not exceed all existing offline RL algorithms at this point, we believe that OPOSM opens exciting avenues for future advances and robotics applications, mainly due to its high versatility. In particular, we believe that the model-based nature of OPOSM will naturally yield additional advantages in future work, such as increased data efficiency in an online setting, and the ability to represent and utilize uncertainty estimates in environmental dynamics (epistemic uncertainty).

Ongoing work includes expanding our approach to include variable-length skills, as we hypothesize this will both improve the performance of skills and our ability to predict their outcomes. Additionally, we are investigating an online variant of our approach, in which the data buffer is augmented with online data to iteratively refine skills and our TAWM. Finally, we are considering ways in which the performance of our offline algorithm can be improved, for example by incorporating epistemic uncertainty estimation in our TAWM, enabling the agent to avoid regions of the state space in which it is uncertain of the dynamics. Epistemic uncertainty estimation could additionally improve safety on real robotic hardware by enabling our agent to be risk-sensitive and avoid behaviors that result in uncertain outcomes.

References

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

Baumli, K., Warde-Farley, D., Hansen, S., and Mnih, V. Relative variational intrinsic control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6732–6740, 2021.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, 2019.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

Eysenbach, B., Khazatsky, A., Levine, S., and Salakhutdinov, R. Mismatched no more: Joint model-policy optimization for model-based rl. *arXiv preprint arXiv:2110.02758*, 2021.

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

Fu, J., Norouzi, M., Nachum, O., Tucker, G., ziyu wang, Novikov, A., Yang, M., Zhang, M. R., Chen, Y., Kumar, A., Paduraru, C., Levine, S., and Paine, T. Benchmarks for deep off-policy evaluation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=kWSeGEeHvF8.

Ghasemipour, S. K. S., Schuurmans, D., and Gu, S. S. Emaq: Expected-max Q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy Q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Pearl, J. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

Pereira, M., Fan, D. D., An, G. N., and Theodorou, E. Mpc-inspired neural network policies for sequential decision making. *arXiv preprint arXiv:1802.05803*, 2018.

Pertsch, K., Lee, Y., and Lim, J. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning*, pp. 188–204. PMLR, 2021.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Rubinstein, R. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

Shankar, T. and Gupta, A. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pp. 8624–8633. PMLR, 2020.

Sharma, A., Ahn, M., Levine, S., Kumar, V., Hausman, K., and Gu, S. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. *arXiv preprint arXiv:2004.12974*, 2020a.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=HJgLZR4KvH.

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

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

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Zhou, W., Bajracharya, S., and Held, D. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pp. 1719–1735. PMLR, 2021.

A. Derivation of Naive Variational Method and Modified EM Algorithm

In order to learn a temporally abstract world model, one must assume that the state at some point in the future s_T can be directly predicted from an initial state s_0 and a skill, without requiring the prediction of intermediate states and actions. We therefore hypothesize the existence of a set of skills that causes s_T to be conditionally independent of $s_1, ..., s_{T-1}$ and $a_0, ..., a_T$, given s_0 and z, leading to the graphical model shown in Fig. 6b. Our goal is to discover this set of skills (that is, learn the set of skill-conditioned policies $\pi(a|s, z)$) that make our conditional independence assumption hold, and the associated distribution $p(s_T|s_0, z)$.

Taking a simple VI approach to derive the evidence lower bound corresponding to our skill model, we first consider the joint density over latent and observed variables:

$$\mathbb{E}_{\tau_T \sim \mathcal{D}} \log p(s_T, a_{0:T-1}|s_{0:T-1}) = \mathbb{E}_{\tau_T \sim \mathcal{D}} \log \int p(s_T, a_{0:T-1}, z|s_{0:T-1}; \psi, \omega, \theta) dz, \tag{3}$$

$$= \mathbb{E}_{\tau_T \sim \mathcal{D}} \log \int p_\psi(s_T|s_0, z) \pi_\theta(\vec{a}|\vec{s}, z) p_\omega(z|s_0) dz, \tag{4}$$

$$\tag{5}$$

Next, we introduce an approximate posterior (encoder network) over skills:

$$= \mathbb{E}_{\tau_T \sim \mathcal{D}} \log \int \frac{q_\phi(z|\tau_T)}{q_\phi(z|\tau_T)} p_\psi(s_T|s_0, z) \pi_\theta(\vec{a}|\vec{s}, z) p_\omega(z|s_0) dz, \tag{6}$$

$$= \mathbb{E}_{\tau_T \sim \mathcal{D}} \log \mathbb{E}_q \frac{p_\psi(s_T|s_0, z) \pi_\theta(\vec{a}|\vec{s}, z) p_\omega(z|s_0)}{q_\phi(z|\tau_T)}, \tag{7}$$

$$\tag{8}$$

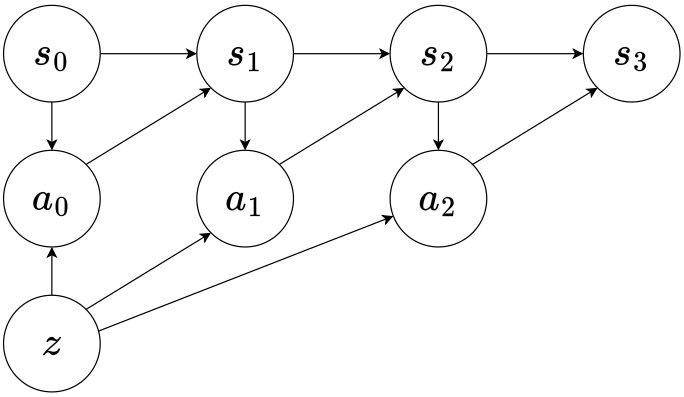
Finally, we arrive at the ELBO by applying Jensen's inequality:

$$\geq \mathbb{E}_{\tau_T \sim \mathcal{D}} \Big[\mathbb{E}_{q_\phi(z|\tau_T)} \left[\log \pi_\theta(\vec{a}|\vec{s}, z) + \log p_\psi(s_T|s_0, z) \right], \tag{9}$$

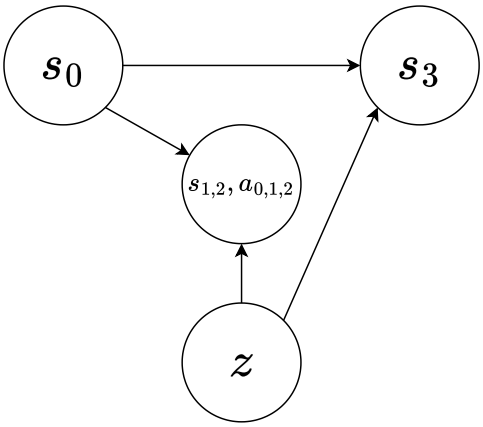
$$- D_{KL}(q_\phi(z|\tau_T)||p_\omega(z|s_0)) \Big] \tag{10}$$

$$= \mathcal{L}(\theta, \psi, \phi, \omega). \tag{11}$$

This ELBO can be jointly maximized with respect to θ, ψ, ϕ, and ω. However, this leads to an issue in which the agent may overestimate its causal influence over state transitions. A detailed example of how this may negatively impact decision-making is given in Sec. B. The issue arises from the fact that the approximate posterior q, which we have so far assumed to be any distribution over z conditioned on τ_T, does not reflect what we know to be true about the environment (namely, that skills can only influence state transitions indirectly through the actions as depicted in Fig. 6a). Instead, the model may learn that skills have a *direct* influence over state transitions. To better explain this, we consider a 1 timestep skill model in which z influences the selection of a_0, which in turn influences the state transition from s_0 to s_1. Assume also that some external noise ϵ also influences s_1 (Fig. 7a). In Pearl's *do* notation from causal theory, we would like to ascertain $p(s_1|s_0, do(z))$, *i.e.,* the effect of the agent intervening on the environment by executing skill z from state s_0. In the method we've described thus far, the model may learn to lump ϵ into z (that is, use z to represent environmental stochasticity that we do not have control over, such as an external disturbance force). In this case, the causal structure assumed by the agent is given in Fig. 7b. Let z_a be the component of z that impacts a_0, and let ϵ be the component of z that has a *direct* effect on s_1. The model may learn a *statistically* correct distribution $p(s_1|s_0, z) = p(s_1|s_0, z_a, \epsilon)$, by virtue of the fact that the ELBO in Eq. (9) is minimized. However, if the agent uses this model to predict the causal influence of z on s_1, the agent will compute $p(s_1|s_0, do(z_a), do(\epsilon))$, or in other words, the effect of executing the skill corresponding to z_a, *and* setting the environmental noise to ϵ. Because the agent cannot in reality set the value of ϵ, it will predict that it has a higher degree of influence over s_1 than it actually does.



(a) True causal structure of the environment. Actions influence state transitions, which are turn influenced by skills (z), with no direct influence of skills on state transitions.



(b) Graphical model of environment given skills that cause long-term state transitions to be conditionally independent of intermediate states and actions, given the initial state and skill.

Figure 6.

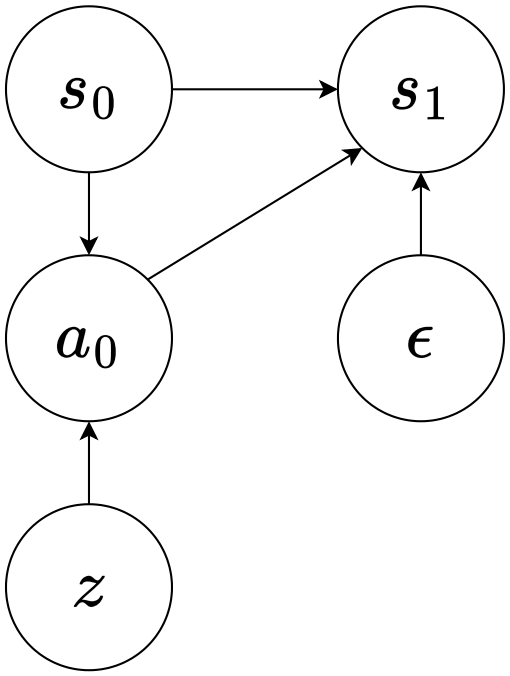
We avoid this issue by restricting the optimization over approximate posteriors to the set that matches the true causal structure of the environment, shown in Fig. 6a, yielding our modified EM approach. Here, at each training step, q is regressed towards the *true* posterior $p(z|\tau_T) \propto \pi_\theta(\vec{a}|\vec{s}, z)p_\omega(z|s_0)$, derived using the true causal structure of the environment depicted in Fig. 6a. This is done by minimizing the KL divergence $D_{KL}(q(z|\tau_T)||p(z|\tau_T)) = \mathbb{E}_{\tau_T \sim \mathcal{D}}\left[\mathbb{E}_{z\sim q_\phi}\left[\log \frac{q_\phi(z|\vec{s},\vec{a})}{\pi_\theta(\vec{a}|\vec{s},z)p_\omega(z|s_0)}\right]\right] + c$, where $c = \log \eta$ is a constant offset that does not affect the optimization.

B. Causality Example

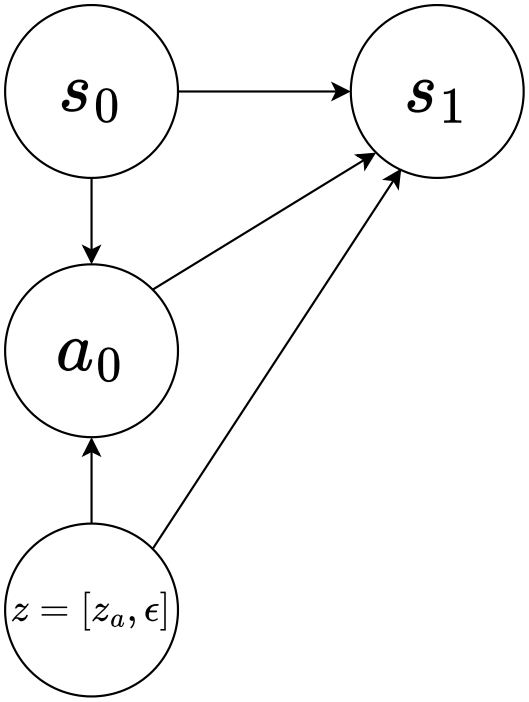
Here we provide an example of how a naive VI approach to skill learning may harm planning performance. Consider the following 4-state, single-timestep MDP: an agent is initialized in state $s = s_0$. It can then select between three actions, a_1, a_2, and a_3. If the agent selects a_1, it transitions to state $s' = s_1$ with probability 1, and earns a reward of $+10$. If the agent selects a_2, then it either transitions to state $s' = s_2$ and earns reward $+100$ with probability 0.1, or transitions to state $s' = s_3$ and earns reward -100 with probability 0.9. Finally, if the agent selects a_3, then it transitions to state s_3 with probability 1, and earns a reward of -100. This MDP is depicted schematically in Fig. 8. Clearly, the optimal action is a_1, because this yields $+10$ reward, while a_2 and a_3 in expectation yield -89 and -100 reward, respectively.

In this MDP, there is no action the agent can select that will cause it to reliably transition to state s_2 and achieve $+100$ reward. However, if the agent learns a set of skills and a skill-conditioned world model simply by optimizing the ELBO derived in Sec. A, the agent may *believe* that it possesses a skill enabling it to transition to state s_2. To illustrate this point, we consider fitting tabular models with a discrete, finite skill set containing 3 skills, $z \in \{1, 2, 3\}$, to a large dataset of state transitions resulting from the agent selecting random actions uniformly from its action-space. One possible skill model the agent may learn is a skill-conditioned policy that maps skills directly to actions, and a TAWM that matches the true dynamics of the environment, given the mapping from skills to actions. We will refer to this model as the "correct" mode, as it correctly models the causal relationship between skills and state transitions. On the other hand, the agent may learn a set of skills and a world model such that the model predicts that state transitions are deterministic, given a skill, causing the agent to predict that it can freely choose to transition to states s_1, s_2, or s_3 *deterministically*. We refer to this model as the "incorrect" model, because this is not a physically realizable set of skills the agent could employ. Both models yield the correct distribution $p(s_T, a_{0:T-1}|s_{0:T-1}$ when skills are marginalized over, and both achieve the same value of the ELBO, which we computed using the python script `compute_elbo.py` (included in the supplemantal material) to be -1.20697, meaning that the ELBO does not prefer the correct model over the incorrect model.

The distributions $q_\phi(z|s, a, s')$, $p_\psi(s'|s, z)$, $\pi(a|s, z)$ and $p_\omega(z|s)$ for the correct model are summarized in the tables below. Dependency on s is omitted, because s does not vary across episodes. Additionally, dependence of q_ϕ on s' is omitted, because it only depends on a:



(a) Causal structure of environment, depicting influence of noise ϵ that is independent of actions.



(b) Incorrect skill model, in which ϵ is part of z. Here the agent learns that it has influence over the environmental noise ϵ, which it in fact does not.

Figure 7.

The distributions for the incorrect model are summarized in the tables below. Here, dependence of q_ϕ on a is omitted, because it only depends on s':

Using the correct skill model, the agent can correctly reason that executing skill $z = 1$ is optimal (which corresponds exactly to selecting action a_1), and correctly estimates its expected future reward to be $+10$. However, if the agent learns the incorrect skill model instead, it will falsely believe that executing skill $z = 2$ is optimal, and that this will yield an expected reward of $+100$, because the agent believes that selecting skill $z = 2$ will cause the state to deterministically transition to $s' = s_2$. However, in reality, selecting skill $z = 2$ corresponds to to selecting action a_1, which yields an expected reward of -89. We can therefore conclude that learning a skill model according to the ELBO derived in Sec. A leads to suboptimal decision-making during planning.

This situation results from the fact that the approximate posterior does not respect the true causal structure of the environment (Fig. 2 in main text), allowing it to essentially encode s' into z. This wouldn't be a problem if our goal is correctly modeling $p(s_T, a_{0:T-1}|s_{0:T-1}$, as both the correct and incorrect models accurately represent this distribution. However, we seek to learn latent variables that have the correct *causal* influence on state transitions, meaning that our model should learn a posterior that that respects the true causal structure of the environment. By Bayes rule, we can derive the true posterior over z, $p(z|\vec{s}, \vec{a})$, given π_θ, p_ψ, and p_ω and the causal structure in Fig. 2 in the main text:

$$p(z|\vec{s}, \vec{a}) = \frac{p(s_{1:T}, a_{0:T-1}|z, s_0)p_\omega(z|s_0)}{p(s_{1:T}, a_{0:T-1}|s_0))} \tag{12}$$

$$= \frac{(\prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t, z))p_\omega(z|s_0)}{p(s_{1:T}, a_{0:T-1}|s_0)}. \tag{13}$$

$p(s_{t+1}|s_t, a_t)$ and $p(s_{1:T}, a_{0:T-1}|s_0)$ do not depend on z, and can therefore be lumped into a normalization constant η, leaving us with

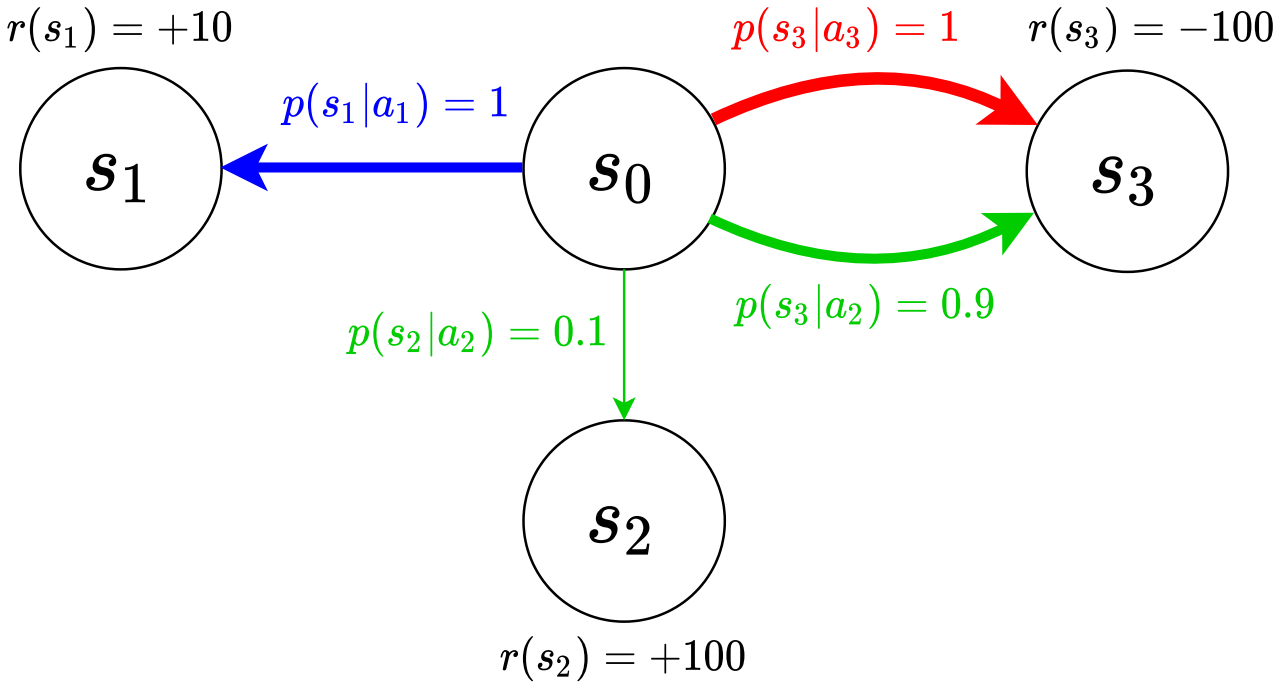


Figure 8. **Causality Example MDP**.

$$p(z|\vec{s},\vec{a}) = \frac{1}{\eta}\left(\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t,z)\right)p_\omega(z|s_0). \tag{14}$$

In our skill learning approach, we therefore train the approximate posterior to minimize the KL divergence between itself and the true posterior, while training π_θ, p_ψ, and p_ω to maximize the ELBO. Because the ELBO is maximized when the the skill-conditioned policy and TAWM imply the same posterior over z as our approximate posterior q_ϕ, the TAWM learned using an approximate posterior that matches the environmental causal structure should also match the structure.

C. Skill Model Implementation Details

Below we detail the implementation details of our model architecture (Fig. 9), learning procedure, and planning algorithm, used to generate our results. Unless otherwise stated, all layers contain 256 neurons, and the dimension of z is 256.

C.1. Skill Posteriors

In all environments, skill posterior networks take as input a sequence of states and actions, and output a single mean vector $\mu_q(\tau_T)$ and standard deviation vector $\sigma_q(\tau_T)$ representing the inferred posterior over z, *i.e.*, $q_\phi(z|\tau_T) = \mathcal{N}(\mu_q(\tau_T), \sigma_q(\tau_T))$. The posterior network consist of a linear layer with a ReLU activation through which the states are passed, followed by a single-layer bidirectional GRU, through which embedded states concatenated with actions are passed. The output of the GRU is then passed through two networks, one for computing the mean and one for computing the standard deviation over z. Both networks consists of 2 linear layers, with a ReLU activation after the first layer. The mean layer has no final activation on its output, while the standard deviation layer has a softplus activation on its final output to make its output strictly positive.

Table 5. $q_\phi(z|a)$ for correct model.

	$z=1$	$z=2$	$z=3$
$a=a_1$	1	0	0
$a=a_2$	0	1	0
$a=a_3$	0	0	1

Table 6. $\pi_\theta(a|z)$ for correct model.

	$a=a_1$	$a=a_2$	$a=a_3$
$z=1$	1	0	0
$z=2$	0	1	0
$z=3$	0	0	1

Table 7. $p_\psi(s'|z)$.

	$s'=s_0$	$s'=s_1$	$s'=s_2$	$s'=s_3$
$z=1$	0	1	0	0
$z=2$	0	0	0.1	0.9
$z=3$	0	0	0	0.1

Table 8. $p_\omega(z)$ for correct model.

$z=1$	$z=2$	$z=3$
$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

C.2. Low-Level Skill-Conditioned Policies

The low-level skill-conditioned policy takes as input the current state of the environment and the skill, and outputs a mean $\mu_\pi(s,z)$ and standard deviation $\sigma_\pi(s,z)$ over the current action, *i.e.*, $\pi_\theta(a|s,z) = \mathcal{N}(\mu_\pi(s,z), \sigma_\pi(s,z))$. The low-level policy used for the maze2d and antmaze environments consists of a 2-layer shared network with ReLU activations, and is followed by a separate mean and standard deviation network, both of which use the same architecture as those described in Sec. C.1. However, for the franka tasks, we use an auto-regressive policy network similar to that described in (Ghasemipour et al., 2021) and (Ajay et al., 2020), in which each element of the action vector has its own network, taking as input the current state and skill, and all previously-sampled action elements. An action vector is sampled element-by-element, with the most recently sampled element becoming an input to the network for the next element.

C.3. Temporally Abstract World Models

Temporally abstract world models take as input the initial state, concatenated with z, and return a mean $\mu_{s_T}(s_0,z)$ and standard deviation $\sigma_{s_T}(s_0,z)$ over predicted terminal states, *i.e.*, $p_\psi(s_T|s_0,z) = \mathcal{N}(\mu_{s_T}(s_0,z), \sigma_{s_T}(s_0,z))$. They consists of a 2-layer shared network with ReLU activations, followed by a mean network and a standard deviation network. The mean network is the same as that described in Sec. C.1. For maze2d and antmaze environments, the standard deviation is also the same as that described in Sec. C.1. However, in the case of the kitchen environments, the standard deviation outputs a single scalar standard deviation that is shared for all elements of the predicted terminal state.

C.4. Skill Priors

The skill prior takes as input the initial state in a subtrajectory, and outputs a mean $\mu_0(s_0)$ and standard deviation $\sigma_0(s_0)$ over z (*i.e.*, $p_\omega(z|s_0) = \mathcal{N}(\mu_0(s_0), \sigma_0(s_0))$). The skill prior consists of a 2-layer shared network with ReLU activations, followed by mean and standard deviation networks with identical architectures to those described in Sec. C.1.

Table 9. $q_\phi(z|a)$ for incorrect model.

	$z=1$	$z=2$	$z=3$
$s'=s_1$	1	0	0
$s'=s_2$	0	1	0
$s'=s_3$	0	0	1

Table 10. $\pi_\theta(a|z)$ for incorrect model.

	$a=a_1$	$a=a_2$	$a=a_3$
$z=1$	1	0	0
$z=2$	0	1	0
$z=3$	0	$\frac{9}{19}$	$\frac{11}{19}$

Table 11. $p_\psi(s'|z)$ for incorrect model.

	$s'=s_0$	$s'=s_1$	$s'=s_2$	$s'=s_3$
$z=1$	0	1	0	0
$z=2$	0	0	1	0
$z=3$	0	0	0	1

Table 12. $p_\omega(z)$ for incorrect model.

$z=1$	$z=2$	$z=3$
$\frac{1}{3}$	$\frac{1}{30}$	$\frac{19}{30}$

C.5. Skill Model Training

To train a skill model, we alternate between the E step, in which one optimization step on the skill posterior is performed according to the E loss, and the M step, in which one optimization step on the skill prior, low-level policy, and TAWM is performed according to the M loss. The Adam (Kingma & Ba, 2014) optimizer was used in both the E and M steps. Training hyperparameters are provided in Table 13.

E Loss Computation: To compute the E loss, a batch of B subtrajectories $\tau_T^{(i)} = [s_{t,i}, a_{t,i}, ..., s_{t+T-1,i}, a_{t+T-1,i}]$ for $i = 1, ..., B$, are passed into the encoder. Each subtrajectory is of length T and is uniformly sampled from the dataset. For each datapoint, and a z is sampled from the posterior using the reparameterization trick. That is, $z_i = \mu_q(\tau_i^{(T)}) + \sigma_q(\tau_i^{(T)}) \cdot \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(\vec{0}, I)$ (Kingma & Welling, 2013). Then, the loss is computed according to

$$\mathcal{L}_E = -\frac{1}{B}\sum_{i=1}^{B}\left[\log \pi_\theta(\vec{a}_i|\vec{s}_i, z_i) + \log p_\omega(z_i|s_{0,i}) - \log q_\phi(z_i|\tau_i^{(T)})\right]. \tag{15}$$

M Loss Computation: To compute the M loss, the same procedure is followed for sampling z vectors from the encoder. However, the M loss is computed according to

$$\mathcal{L}_M = -\frac{1}{B}\sum_{i=1}^{B}\left[\log \pi_\theta(\vec{a}_i|\vec{s}_i, z_i) + \log p_\psi(s_{T,i}|s_{0,i}, z_i) + \log p_\omega(z_i|s_{0,i})\right]. \tag{16}$$

C.6. Planning Details

Skill plans are optimized using a CEM planner that plans in a "whitened" $\mathcal{E}$-space. At each iteration of planning, a batch of K L-length sequences of ϵ vectors is sampled (initially from a zero-mean, unit-variance Gaussian). Each sequence in the

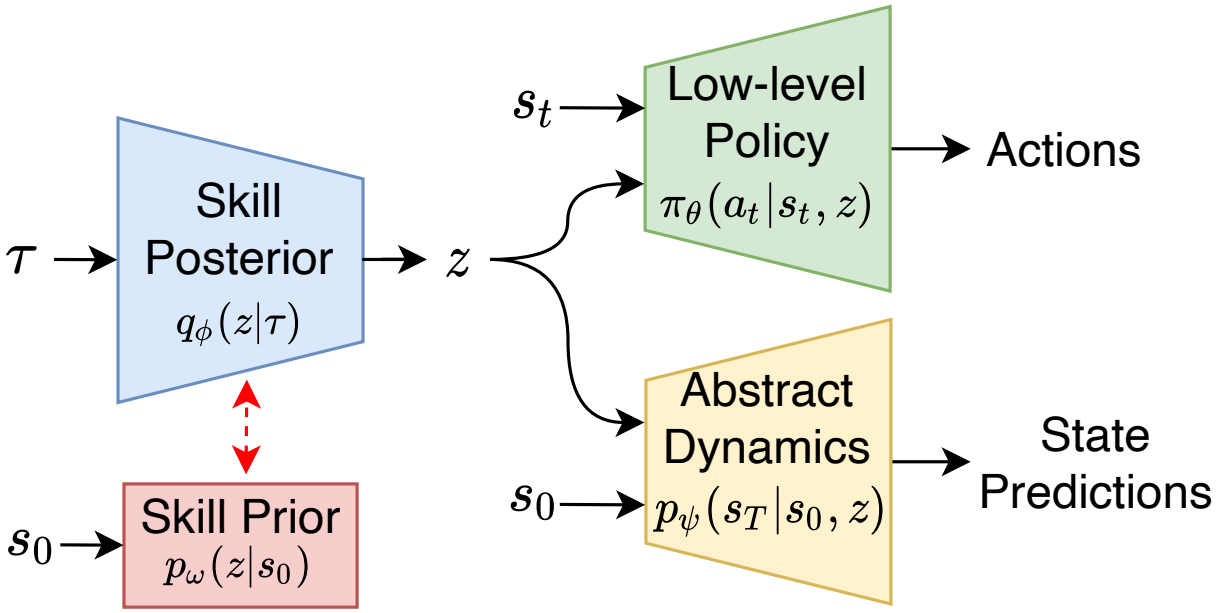


Figure 9. **Skill model architecture**. During offline training, sub-trajectories τ_T are sampled from the dataset and fed into a skill posterior q_ϕ, which learns to infer the skill z. This z is then used to condition the predictions of the low-level skill-conditioned policy π_θ, and the temporally abstract dynamics p_ψ. The skill prior p_ω infers z from the initial state of the sub-trajectory only.

Task	B	$T+1$	lr_E	lr_M
Mazd2d-medium	100	40	$5e-5$	$5e-5$
Maze2d-large	100	40	$5e-5$	$5e-5$
Antmaze-medium	100	10	$5e-5$	$5e-5$
Antmaze-large	100	40	$5e-5$	$5e-5$
Kitchen-mixed	100	5	$5e-5$	$5e-5$
Kitchen-partial	100	5	$5e-5$	$5e-5$
Kitchen-complete	100	5	$5e-5$	$5e-5$

Table 13. Training hyperparameters for EM skill learning procedure.

batch is evaluated according to our TAWM and task reward function, given the current state of the environment. A diagonal Gaussian is then fit to the top N_{keep} ϵ-sequences, from which a new batch is sampled. This process is repeated for N_{iters}, at which point the best ϵ sequence is returned. The skill corresponding to the first element in this sequence ϵ_0 is then executed, by first converting it to a z according to $z = \mu_0(s_0) + \sigma_0(s_0) \cdot \epsilon_0$. The skill is then executed for τ timesteps before a new optimal plan is computed. Parameters for the planning procedure are provided in Table 14. For Kichen tasks, skills were replanned initially after 10 timesteps, then after 5 timesteps after the first two tasks were completed.

D. Visualization of Plans for Maze2d

To better understand the relative advantages of planning in skill-space using our TAWM compared to planning in raw action-space using a traditional single-timestep dynamics model, we visualized plans generated by both approaches in Fig. 10. We additionally visualize the trajectory taken by the agent executing the plan open-loop, without iterative replanning. Here we notice that although both planning methods predict that the agent will reach the goal, the trajectories generated by low-level planning are often physically impossible to execute, and tend to cut through walls. On the other hand, the skill-sequence plans do not violate the physical constraints of the environment, and instead always predict that the agent will travel through free space. The result is that the agent is able to consistently reach the goal when executing planned skill sequences open-loop, without any replanning, but is not able to reach the goal when executing low-level action plans (as indicated by the blue line colliding with walls and failing to reach the goal). We also notice that when the agent deviates from

Task	H	K	L	N_{keep}	N_{iters}	τ
Maze2d-medium	40	100	10	50	100	40
Maze2d-large	40	100	10	50	100	40
Antmaze-Medium	40	1000	3	200	10	30
Antmaze-Large	40	100	10	50	100	10
Kitchen-Mixed	5	1000	10	100	10	10,5
Kitchen-Partial	5	1000	10	100	10	10,5
Kitchen-Complete	5	1000	10	100	10	10,5

Table 14. Parameters for Skill-Sequence Planning.

its planned trajectory while executing a skill sequence, it is able to recover, likely due to the fact that skills are essentially policies which have the ability to perform closed-loop feedback. On the other hand, the agent fails to recover after it deviates from its low-level action plan, likely because there is no feedback present in this situation and therefore no ability to correct for deviations.

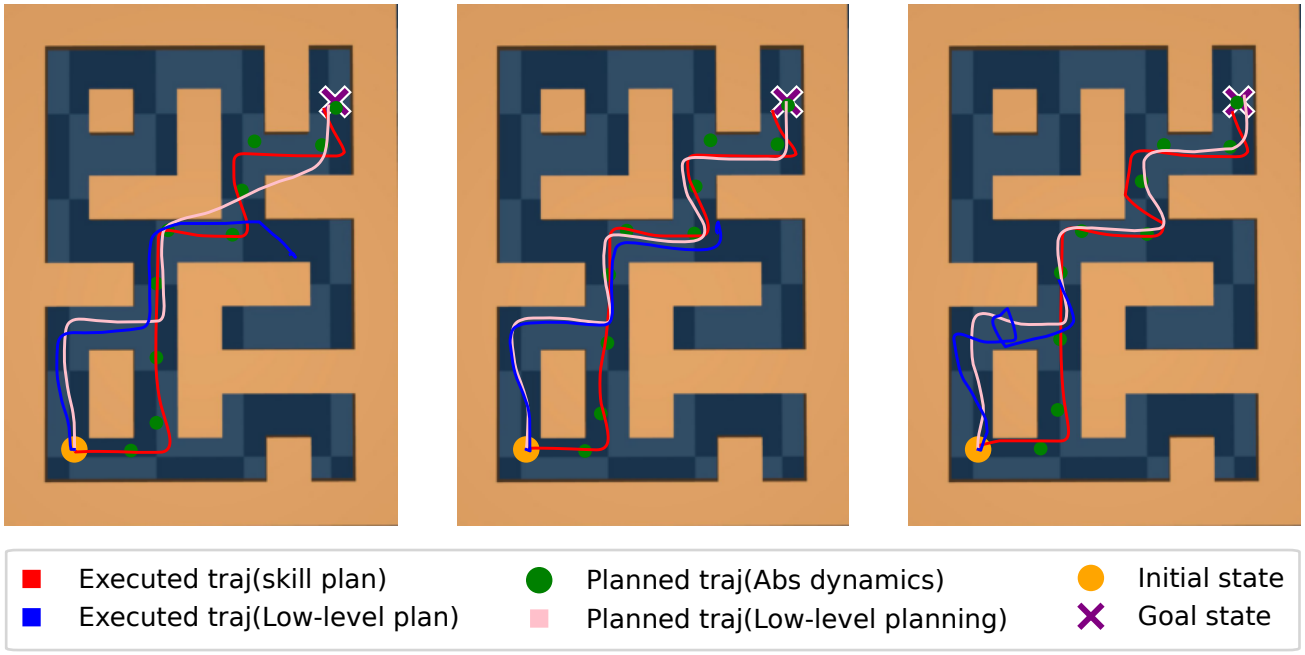


Figure 10. **Visualization of Low-Level vs. Abstract Plans**. Green dots denote the predicted states that the agent will pass through while executing its planned skill sequence, according to the TAWM. Red lines denote the trajectories actually followed while executing the skill sequence. Pink lines denote the predicted trajectories that the agent will follow while executing its low-level action plans, according to the low-level dynamics model. Blue lines denote the trajectory actually followed while executing the low-level action sequence. We notice that the trajectories planned in low-level action space (pink lines) tend to violate physical constraints of the environment, while those planned in skill-space do not. Additionally, the agent can reliably reach its goal by following a planned skill sequence even open-loop, but often fails to reach its goal when following a planned low-level action sequence, when no iterative replanning is used.