Skill-based Model-based Reinforcement Learning

Lucy Xiaoyang Shi[1] **Joseph J. Lim**[2,3*] **Youngwoon Lee**[1]
[1]University of Southern California [2]KAIST [3]NAVER AI Lab

Abstract: Model-based reinforcement learning (RL) is a sample-efficient way of learning complex behaviors by leveraging a learned single-step dynamics model to plan actions in imagination. However, planning every action for long-horizon tasks is not practical, akin to a human planning out every muscle movement. Instead, humans efficiently plan with high-level skills to solve complex tasks. From this intuition, we propose a **Ski**ll-based **Mo**del-based RL framework (**SkiMo**) that enables planning in the skill space using a *skill dynamics model*, which directly predicts the skill outcomes, rather than predicting all small details in the intermediate states, step by step. For accurate and efficient long-term planning, we *jointly* learn the skill dynamics model and a skill repertoire from prior experience. We then harness the learned skill dynamics model to accurately simulate and plan over long horizons in the skill space, which enables efficient downstream learning of long-horizon, sparse reward tasks. Experimental results in navigation and manipulation domains show that SkiMo extends the temporal horizon of model-based approaches and improves the sample efficiency for both model-based RL and skill-based RL. Code and videos are available at https://clvrai.com/skimo.

Keywords: Model-Based Reinforcement Learning, Skill Dynamics Model

1 Introduction

A key trait of human intelligence is the ability to plan abstractly for solving complex tasks [1]. For instance, we perform cooking by imagining outcomes of high-level skills like washing and cutting vegetables, instead of planning every muscle movement involved [2]. This ability to plan with temporally-extended skills helps to scale our internal model to long-horizon tasks by reducing the search space of behaviors. To apply this insight to artificial intelligence agents, we propose a novel skill-based and model-based reinforcement learning (RL) method, which learns a model and a policy in a high-level skill space, enabling accurate long-term prediction and efficient long-term planning.

Typically, model-based RL involves learning a flat single-step dynamics model, which predicts the next state from the current state and action. This model can then be used to simulate "imaginary" trajectories, which significantly improves sample efficiency over their model-free alternatives [3, 4]. However, such model-based RL methods have shown only limited success in long-horizon tasks due to inaccurate long-term prediction [5] and computationally expensive search [6, 7, 8].

Skill-based RL enables agents to solve long-horizon tasks by acting with multi-action subroutines (skills) [9, 10, 11, 12, 13, 14] instead of primitive actions. This temporal abstraction of actions enables systematic long-range exploration and allows RL agents to plan farther into the future, while requiring a shorter horizon for policy optimization, which makes long-horizon downstream tasks more tractable. Yet, on complex long-horizon tasks, skill-based RL still requires a few million to billion environment interactions to learn [13], which is impractical for real-world applications.

To combine the best of both model-based RL and skill-based RL, we propose **Ski**ll-based **Mo**del-based RL (**SkiMo**), which enables effective planning in the skill space using a *skill dynamics model*. Given a state and a skill to execute, the skill dynamics model directly predicts the resultant state after skill execution, without needing to model every intermediate step and low-level action (Figure 1), whereas the flat dynamics model predicts the immediate next state after one action execution. Thus,

*AI Advisor at NAVER AI Lab

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

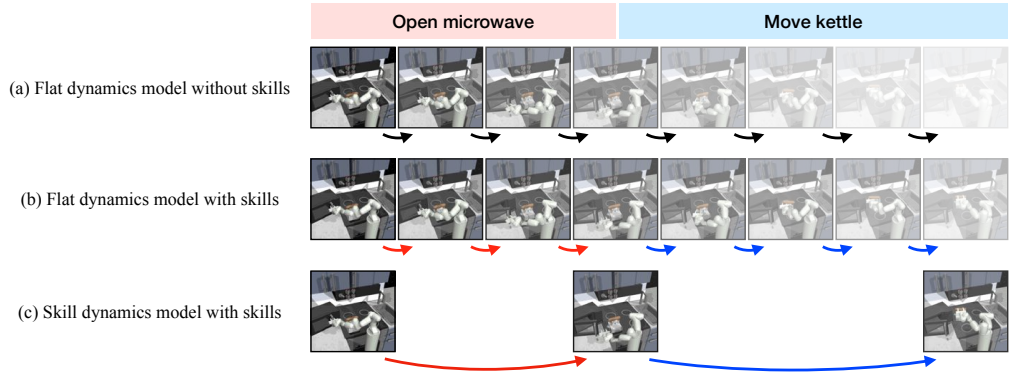


Figure 1: Intelligent agents can use their internal models to imagine potential futures for planning. Instead of planning out every primitive action (black arrows in **a**), they aggregate action sequences into skills (red and blue arrows in **b**). Further, they can leap directly to the predicted outcomes of executing skills in sequence (red and blue arrows in **c**), which leads to better long-term prediction and planning compared to predicting step-by-step (blurriness of images represents the level of error accumulation in prediction).

planning with skill dynamics requires fewer predictions than flat dynamics, resulting in more reliable long-term future predictions and plans.

Concretely, we first jointly learn the skill dynamics model and a skill repertoire from large offline datasets collected across diverse tasks [15, 12, 16]. This joint training shapes the skill embedding space for easy skill dynamics prediction and skill execution. Then, to solve a complex downstream task, we train a hierarchical task policy that acts in the learned skill space. For more efficient policy learning and better planning, we leverage the skill dynamics model to simulate skill trajectories.

The main contribution of this work is to propose *Skill-based Model-based RL (SkiMo)*, a novel sample-efficient model-based hierarchical RL algorithm that leverages task-agnostic data to extract not only a reusable skill set but also a skill dynamics model. The skill dynamics model enables efficient and accurate long-term planning for sample-efficient RL. Our experiments show that our method outperforms the state-of-the-art skill-based and model-based RL algorithms on long-horizon navigation and robotic manipulation tasks with sparse rewards.

2 Related Work

Model-based RL leverages a (learned) dynamics model of the environment to plan a sequence of actions that leads to the desired behavior. The dynamics model predicts the future state of the environment, and optionally the associated reward, after taking a specific action for planning [8, 4] or subsequent policy search [17, 3, 18, 4]. By simulating candidate behaviors in imagination instead of in the physical environment, model-based RL improves the sample efficiency. The imaginary rollouts can be used for planning, e.g., CEM [19] and MPPI [20], as well as for policy optimization [3, 4] Yet, due to the accumulation of prediction error at each step and the increasing search space, finding an optimal, long-horizon plan is inaccurate and computationally expensive [6, 7, 8].

To facilitate learning of long-horizon behaviors, skill-based RL lets the agent act over temporally-extended skills (i.e. options [21] or motion primitives [22]), which can be represented as sub-policies or a coordinated sequence of low-level actions. Temporal abstraction effectively reduces the task horizon for the agent and enables directed exploration [23], a major challenge in RL. The reusable skills can be manually defined [22, 24, 10, 11, 14], extracted from large offline datasets [25, 26, 15, 27, 28], discovered online in an unsupervised manner [29, 30], or acquired in the form of goal-reaching policies [31, 32, 33, 34, 18]. However, skill-based RL is still impractical for real-world applications, requiring a few million to billion environment interactions [13]. In this paper, we use model-based RL to guide the planning of skills to improve the sample efficiency of skill-based approaches.

There have been attempts to plan over skills in model-based RL [29, 35, 5, 36, 37]. However, most of these approaches [29, 5, 36] still utilize the conventional flat (single-step) dynamics model, which struggles at handling long-horizon planning due to error accumulation. Wu et al. [35] proposes to

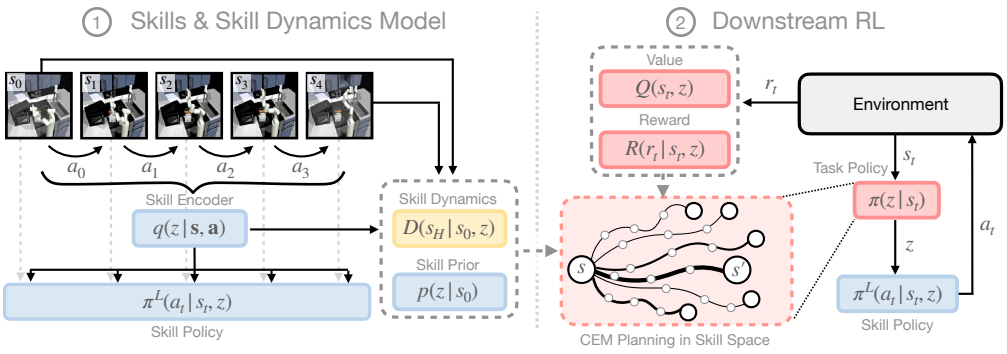


Figure 2: Our approach, SkiMo, combines model-based RL and skill-based RL for sample efficient learning of long-horizon tasks. SkiMo consists of two phases: (1) learn a skill dynamics model and a skill repertoire from offline task-agnostic data, and (2) learn a high-level policy for the downstream task by leveraging the learned model and skills. We omit the encoded latent state $\mathbf{h}$ in the figure and directly write the observation $\mathbf{s}$ for clarity, but most modules take the latent state $\mathbf{h}$ as input.

learn a temporally-extended dynamics model; however, it conditions on low-level actions rather than skills and is only used for low-level planning. A concurrent work, Shah et al. [37], is most similar to our work in that it learns a skill dynamics model, but with a limited set of discrete, manually-defined skills. To fully unleash the potential of temporally abstracted skills, we devise a skill-level dynamics model to provide accurate long-term prediction, which is essential for solving long-horizon tasks. To the best of our knowledge, *SkiMo* is the first work that jointly learns skills and a skill dynamics model from data for model-based RL.

3 Method

In this paper, we aim to improve the long-horizon learning capability of RL agents. To enable accurate long-term prediction and efficient long-horizon planning for RL, we introduce *SkiMo*, a novel skill-based and model-based RL algorithm that combines the benefits of both frameworks. A key change to prior model-based approaches is the use of a *skill dynamics model* that directly predicts the outcome of a chosen skill, which enables efficient and accurate long-term planning. Our approach consists of two phases: (1) learning the skill dynamics model and skills from an offline dataset (Section 3.3) and (2) downstream task learning with the skill dynamics model (Section 3.4), as illustrated in Figure 2.

3.1 Preliminaries

RL We formulate a problem as a Markov decision process [38], which is defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P, \rho_0, \gamma)$ of the state space $\mathcal{S}$, action space $\mathcal{A}$, reward $R(\mathbf{s}, \mathbf{a})$, transition probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, initial state distribution ρ_0, and discounting factor γ. A policy $\pi(\mathbf{a}|\mathbf{s})$ maps from a state $\mathbf{s}$ to an action $\mathbf{a}$. RL aims to find the optimal policy that maximizes the expected discounted return, $\mathbb{E}_{\mathbf{s}_0 \sim \rho_0, (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_{T_i}) \sim \pi} \left[\sum_{t=0}^{T_i - 1} \gamma^t R(\mathbf{s}_t, \mathbf{a}_t) \right]$, where T_i is the variable episode length.

Unlabeled Offline Data We assume access to a reward-free task-agnostic dataset [15, 12], which is a set of N state-action trajectories, $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$. Since it is task-agnostic, this data can be collected from training data for other tasks, unsupervised exploration, or human teleoperation. We do not assume this dataset contains solutions for the downstream task; therefore, tackling the downstream task requires re-composition of skills learned from diverse trajectories.

Skill-based RL We define skills as a sequence of actions $(\mathbf{a}_0, \ldots, \mathbf{a}_{H-1})$ with a fixed horizon H[2] and parameterize skills as a skill latent $\mathbf{z}$ and skill policy, $\pi^L(\mathbf{a}|\mathbf{s}, \mathbf{z})$, that maps a skill latent and state to the corresponding action sequence. The skill latent and skill policy can be trained using variational

[2]It is worth noting that our method is compatible with variable-length skills [26, 39, 27] and goal-conditioned skills [18] with minimal change; however, for simplicity, we adopt fixed-length skills of $H = 10$ in this paper.

auto-encoder (VAE [40]), where a skill encoder $q(\mathbf{z}|(\mathbf{s},\mathbf{a})_{0:H-1})$ embeds a sequence of transitions into a skill latent $\mathbf{z}$, and the skill policy decodes it back to the original action sequence. Following SPiRL [12], we also learn a skill prior $p(\mathbf{z}|\mathbf{s})$, which is the skill distribution in the offline data, to guide the downstream task policy to explore promising skills over the large skill space.

3.2 SkiMo Model Components

SkiMo consists of three major model components: the skill policy (π_θ^L), skill dynamics model (D_ψ), and task policy (π_ϕ), along with auxiliary components for representation learning and value estimation. A state encoder E_ψ first encodes an observation $\mathbf{s}$ into the latent state $\mathbf{h}$. Then, given a skill $\mathbf{z}$, the skill dynamics D_ψ predicts the skill effect in the latent space. The task policy π_ϕ, reward function R_ϕ, and value function Q_ϕ predict a skill, reward, and value on the (imagined) latent state, respectively. The following is a summary of the notations of our model components:

$$
\begin{array}{lll|lll}
\text{State encoder:} & \mathbf{h}_t = E_\psi(\mathbf{s}_t) & & \text{Skill dynamics:} & \hat{\mathbf{h}}_{t+H} = D_\psi(\mathbf{h}_t, \mathbf{z}_t) \\
\text{Observation decoder:} & \hat{\mathbf{s}}_t = O_\theta(\mathbf{h}_t) & & \text{Task policy:} & \hat{\mathbf{z}}_t \sim \pi_\phi(\mathbf{h}_t) \\
\text{Skill prior:} & \hat{\mathbf{z}}_t \sim p_\theta(\mathbf{s}_t) & & \text{Reward:} & \hat{r}_t = R_\phi(\mathbf{h}_t, \mathbf{z}_t) \\
\text{Skill encoder:} & \mathbf{z}_t \sim q_\theta((\mathbf{s},\mathbf{a})_{t:t+H-1}) & & \text{Value:} & \hat{v}_t = Q_\phi(\mathbf{h}_t, \mathbf{z}_t) \\
\text{Skill policy:} & \hat{\mathbf{a}}_t = \pi_\theta^L(\mathbf{s}_t, \mathbf{z}_t) & & &
\end{array} \tag{1}
$$

For convenience, we label the trainable parameters ψ, θ, ϕ of each component according to which phase they are trained on:

1. **Learned from offline data and finetuned in downstream RL** ($\psi = \{\psi_E, \psi_D\}$): The state encoder ($E_\psi$) and the skill dynamics model (D_ψ) are first trained on the offline task-agnostic data and then finetuned in downstream RL to account for unseen states and transitions.
2. **Learned only from offline data** ($\theta = \{\theta_O, \theta_q, \theta_p, \theta_{\pi^L}\}$): The observation decoder (O_θ), skill encoder (q_θ), skill prior (p_θ), and skill policy (π_θ^L) are learned from the offline data.
3. **Learned in downstream RL** ($\phi = \{\phi_Q, \phi_R, \phi_\pi\}$): The value ($Q_\phi$) and the reward ($R_\phi$) function, and the high-level task policy (π_ϕ) are trained for the downstream task using environment interactions.

3.3 Pre-Training Skill Dynamics Model and Skills from Task-agnostic Data

Our method, *SkiMo*, consists of pre-training and downstream RL phases. In pre-training, *SkiMo* leverages offline data to extract (1) skills for temporal abstraction of actions, (2) skill dynamics for skill-level planning on a latent state space, and (3) a skill prior [12] to guide exploration. Specifically, we jointly learn a skill policy and skill dynamics model, instead of learning them separately [35, 5, 36], in a self-supervised manner. The key insight is that this joint training could shape the latent skill space $\mathbb{Z}$ and state embedding in that the skill dynamics model can easily predict the future.

In contrast to prior works that learn models completely online [3, 41, 4], we leverage existing offline task-agnostic datasets to pre-train a skill dynamics model and skill policy. This offers the benefit that the model and skills are agnostic to specific tasks so that they may be used in multiple tasks. Afterwards in the downstream RL phase, the agent continues to finetune the skill dynamics model to accommodate task-specific trajectories.

To learn a low-dimensional skill latent space $\mathbb{Z}$ that encodes action sequences, we train a conditional VAE [40] on the offline dataset that reconstructs the action sequence through a skill embedding given a state-action sequence as in SPiRL [12, 16]. Specifically, given H consecutive states and actions $(\mathbf{s},\mathbf{a})_{0:H-1}$, a skill encoder q_θ predicts a skill embedding $\mathbf{z}$ and a skill decoder π_θ^L (i.e. the low-level skill policy) reconstructs the original action sequence from $\mathbf{z}$:

$$
\mathcal{L}_{\text{VAE}} = \mathbb{E}_{(\mathbf{s},\mathbf{a})_{0:H-1}\sim\mathcal{D}}\left[\frac{\lambda_{\text{BC}}}{H}\sum_{i=0}^{H-1}\underbrace{(\pi_\theta^L(\mathbf{s}_i,\mathbf{z})-\mathbf{a}_i)^2}_{\text{Behavioral cloning}} + \beta\cdot\underbrace{KL\big(q_\theta(\mathbf{z}|(\mathbf{s},\mathbf{a})_{0:H-1})\parallel p(\mathbf{z})\big)}_{\text{Embedding regularization}}\right], \tag{2}
$$

where $\mathbf{z}$ is sampled from q_θ and $\lambda_{\text{BC}}, \beta$ are weighting factors for regularizing the skill latent $\mathbf{z}$ distribution to a prior of a $\tanh$-transformed unit Gaussian distribution, $Z \sim \tanh(\mathcal{N}(0,1))$.

To ensure the latent skill space is suited for long-term prediction, we *jointly* train a skill dynamics model with the VAE above. The skill dynamics model learns to predict $\mathbf{h}_{t+H}$, the latent state H-steps

ahead conditioned on a skill $\mathbf{z}$, for N sequential skill transitions using the latent state consistency loss [4]. To prevent a trivial solution and encode rich information from observations, we additionally train an observation decoder O_θ using the observation reconstruction loss. Altogether, the skill dynamics D_ψ, state encoder E_ψ, and observation decoder O_θ are trained on the following objective:

$$\mathcal{L}_{\text{REC}} = \mathbb{E}_{(\mathbf{s},\mathbf{a})_{0:NH}\sim\mathcal{D}}\Bigg[\sum_{i=0}^{N-1}\Big[\underbrace{\lambda_{\text{O}}\|\mathbf{s}_{iH} - O_\theta(E_\psi(\mathbf{s}_{iH}))\|_2^2}_{\text{Observation reconstruction}} + \underbrace{\lambda_{\text{L}}\|D_\psi(\hat{\mathbf{h}}_{iH}, \mathbf{z}_{iH}) - E_{\psi^-}(\mathbf{s}_{(i+1)H})\|_2^2}_{\text{Latent state consistency}}\Big]\Bigg], \tag{3}$$

where $\lambda_{\text{O}}, \lambda_{\text{L}}$ are weighting factors and $\hat{\mathbf{h}}_0 = E_\psi(\mathbf{s}_0)$ and $\hat{\mathbf{h}}_{(i+1)H} = D_\psi(\hat{\mathbf{h}}_{iH}, \mathbf{z}_{iH})$ such that gradients are back-propagated through time. For stable training, we use a target network whose parameter ψ^- is slowly soft-copied from ψ.

Furthermore, to guide the exploration for downstream RL, we also extract a skill prior [12] from offline data that predicts the skill distribution for any state. The skill prior is trained by minimizing the KL divergence between output distributions of the skill encoder q_θ and the skill prior p_θ:

$$\mathcal{L}_{\text{SP}} = \mathbb{E}_{(\mathbf{s},\mathbf{a})_{0:H-1}\sim\mathcal{D}}\Big[\lambda_{\text{SP}} \cdot KL\Big(\mathbf{sg}(q_\theta(\mathbf{z}|\mathbf{s}_{0:H-1}, \mathbf{a}_{0:H-1})) \parallel p_\theta(\mathbf{z}|\mathbf{s}_0)\Big)\Big], \tag{4}$$

where λ_{SP} is a weighting factor and **sg** denotes the stop gradient operator. Combining the objectives above, we jointly train the policy, model, and prior, which leads to a well-shaped skill latent space that is optimized for both skill reconstruction and long-term prediction:

$$\mathcal{L} = \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{REC}} + \mathcal{L}_{\text{SP}} \tag{5}$$

3.4 Downstream Task Learning with Learned Skill Dynamics Model

To accelerate downstream RL with the learned skill repertoire, *SkiMo* learns a high-level task policy $\pi_\phi(\mathbf{z}_t|\mathbf{h}_t)$ that outputs a latent skill embedding $\mathbf{z}_t$, which is then translated into a sequence of H actions using the pre-trained skill policy π_θ^L to act in the environment [12, 16].

To further improve the sample efficiency, we propose to use model-based RL in the skill space by leveraging the skill dynamics model. The skill dynamics model and task policy can generate imaginary rollouts in the skill space by repeating (1) sampling a skill, $\mathbf{z}_t \sim \pi_\phi(\mathbf{h}_t)$, and (2) predicting H-step future after executing the skill, $\mathbf{h}_{t+H} = D_\psi(\mathbf{h}_t, \mathbf{z}_t)$. Our skill dynamics model requires only $1/H$ dynamics predictions and action selections of the flat model-based RL approaches [3, 4], resulting in more efficient and accurate long-horizon imaginary rollouts (see Appendix, Figure 10).

Following TD-MPC [4], we leverage these imaginary rollouts both for planning (Algorithm 2) and policy optimization (Equation (7)), significantly reducing the number of necessary environment interactions. During rollout, we perform Model Predictive Control (MPC), which re-plans every step using CEM and executes the first skill of the skill plan (see Appendix, Section C for more details).

To evaluate imaginary rollouts, we train a reward function $R_\phi(\mathbf{h}_t, \mathbf{z}_t)$ that predicts the sum of H-step rewards[3], r_t, and a Q-value function $Q_\phi(\mathbf{h}_t, \mathbf{z}_t)$. We also finetune the skill dynamics model D_ψ and state encoder E_ψ on the downstream task to improve the model prediction:

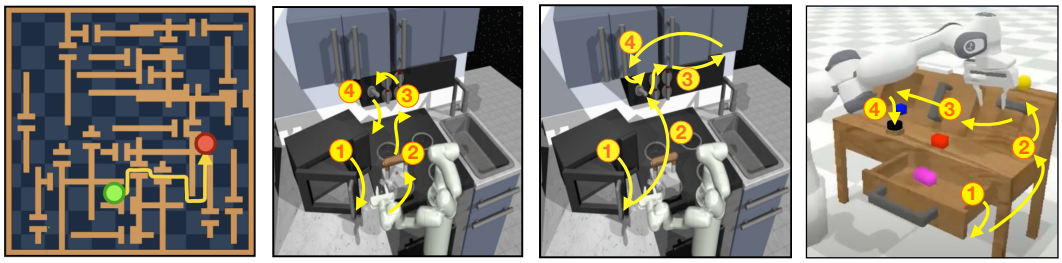
$$\begin{aligned}\mathcal{L}'_{REC} = \mathbb{E}_{\mathbf{s}_t,\mathbf{z}_t,\mathbf{s}_{t+H},r_t\sim\mathcal{D}}\Bigg[&\underbrace{\lambda_{\text{L}}\|D_\psi(\hat{\mathbf{h}}_t, \mathbf{z}_t) - E_{\psi^-}(\mathbf{s}_{t+H})\|_2^2}_{\text{Latent state consistency}} + \underbrace{\lambda_{\text{R}}\|r_t - R_\phi(\hat{\mathbf{h}}_t, \mathbf{z}_t)\|_2^2}_{\text{Reward prediction}} \\ &+ \underbrace{\lambda_{\text{V}}\|r_t + \gamma Q_{\phi^-}(\hat{\mathbf{h}}_{t+H}, \pi_\phi(\hat{\mathbf{h}}_{t+H})) - Q_\phi(\hat{\mathbf{h}}_t, \mathbf{z}_t)\|_2^2}_{\text{Value prediction}}\Bigg].\end{aligned} \tag{6}$$

Finally, we train a high-level task policy π_ϕ to maximize the estimated Q-value while regularizing it to the pre-trained skill prior p_θ [12], which helps the policy output plausible skills:

$$\mathcal{L}_{RL} = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\big[-Q_\phi(\hat{\mathbf{h}}_t, \pi_\phi(\mathbf{sg}(\hat{\mathbf{h}}_t))) + \alpha \cdot KL\big(\pi_\phi(\mathbf{z}_t|\mathbf{sg}(\hat{\mathbf{h}}_t)) \parallel p_\theta(\mathbf{z}_t|\mathbf{s}_t)\big)\big]. \tag{7}$$

For both Equation (6) and Equation (7), consecutive skill-level transitions can be sampled together, so that the models can be trained using backpropagation through time, similar to Equation (3).

[3]For clarity, we use r_t to denote the sum of H-step rewards $\sum_{i=0}^{H-1} r_{t+i}$



(a) Maze (b) Kitchen (c) Mis-aligned Kitchen (d) CALVIN

Figure 3: We evaluate our method on four long-horizon, sparse reward tasks. (a) The green point mass navigates the maze to reach the goal (red). (b, c) The robot arm in the kitchen must complete four tasks in the correct order (*Microwave - Kettle - Bottom Burner - Light* and *Microwave - Light - Slide Cabinet - Hinge Cabinet*). (d) The robot arm needs to complete four tasks in the correct order (*Open Drawer - Turn on Lightbulb - Move Slider Left - Turn on LED*).

4 Experiments

In this paper, we propose a model-based RL approach that can efficiently and accurately plan long-horizon trajectories over the skill space by leveraging the skills and skill dynamics model learned from a task-agnostic dataset. In our experiments, we aim to answer the following questions: (1) Can the skill dynamics model improve the efficiency of RL for long-horizon tasks? and (2) Is the joint training of skills and the skill dynamics model essential for efficient model-based RL?

We compare SkiMo with prior model-based RL and skill-based RL methods (summarized in Table 1) on four long-horizon tasks with sparse rewards: 2D maze navigation, two kitchen manipulation, and tabletop manipulation tasks, as illustrated in Figure 3 (see Appendix, Section C for more details).

4.1 Tasks

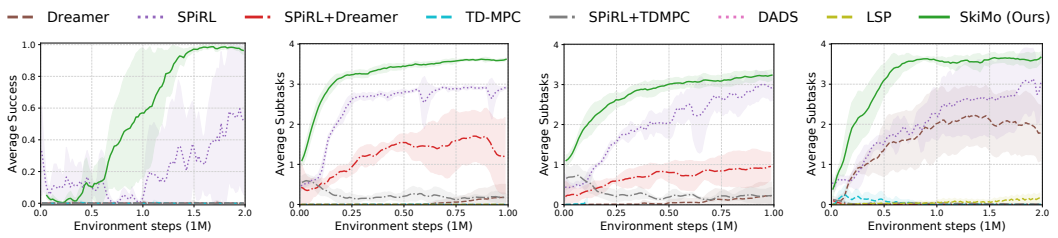
Maze We use the maze navigation task from Pertsch et al. [16], where a point mass agent is randomly initialized near the green region and needs to reach the fixed goal region in red (Figure 3a). The agent observes its 2D position and 2D velocity, and controls its (x, y)-velocity. The agent receives a sparse reward of 100 only when it reaches the goal. The task-agnostic offline dataset from Pertsch et al. [16] consists of 3,046 trajectories between randomly sampled initial and goal positions.

Kitchen We use the FrankaKitchen environment and 603 teleoperation trajectories from D4RL [42]. The 7-DoF Franka Emika Panda arm needs to perform four sequential sub-tasks (*Microwave - Kettle - Bottom Burner - Light*). In **Mis-aligned Kitchen**, we also test another task sequence (*Microwave - Light - Slide Cabinet - Hinge Cabinet*), which has a low sub-task transition probability in the offline data distribution [16]. The agent observes 11D robot state and 19D object state, and uses 9D joint velocity control. The agent receives a reward of 1 for every sub-task completion in order.

CALVIN We adapt the CALVIN benchmark [43] to have the target task *Open Drawer - Turn on Lightbulb - Move Slider Left - Turn on LED* and 21D robot and object states. It uses the Franka Panda arm with 7D end-effector pose control. The offline data is from 1,239 trajectories of play data [43]. The agent receives a reward of 1 for every sub-task completion in the correct order.

4.2 Baselines and Ablated Methods

- **Dreamer** [3] and **TD-MPC** [4] learn a flat (single-step) dynamics and train a policy using latent imagination to achieve a high sample efficiency.
- **DADS** [29] discovers skills and learns a dynamics model through unsupervised learning.
- **LSP** [36] plans in the skill space, but using a single-step dynamics model from Dreamer.
- **SPiRL** [12] learns skills and a skill prior, and guides a high-level policy using the learned prior.



(a) Maze (b) Kitchen (c) Mis-aligned Kitchen (d) CALVIN

Figure 4: Learning curves of our method and baselines. All averaged over 5 random seeds.

- **SPiRL + Dreamer** and **SPiRL + TD-MPC** pre-train the skills using SPiRL and learn a policy and model in the skill space (instead of the action space) using Dreamer and TD-MPC, respectively. In contrast to SkiMo, these baselines do not jointly train the model and skills.

4.3 Results

Maze Maze navigation poses a hard exploration problem due to the sparsity of the reward: the agent only receives reward after taking 1,000+ steps to reach the goal. Figure 4a shows that only SkiMo is able to consistently reach the goal, whereas baselines struggle to learn a policy or an accurate model due to the challenges in sparse feedback and long-term planning.

To better understand the result, we qualitatively analyze the behavior of each agent in Appendix, Figure 9. Dreamer and TD-MPC have a small coverage of the maze since it is challenging to coherently explore for 1,000+ steps to reach the goal from taking primitive actions. SPiRL is able to explore a large fraction of the maze, but it does not learn to consistently find the goal due to difficult policy optimization in long-horizon tasks. On the other hand, SPiRL + Dreamer and SPiRL + TD-MPC fail to learn an accurate model and often collide with walls.

Kitchen Figure 4b demonstrates that SkiMo reaches the same performance (above 3 sub-tasks) with 5x less environment interactions than SPiRL. In contrast, Dreamer and TD-MPC rarely succeed on the first sub-task due to the sparse reward. SPiRL + Dreamer and SPiRL + TD-MPC perform better than flat model-based RL by leveraging skills, yet the independently trained model and policy are not accurate enough to consistently achieve more than two sub-tasks.

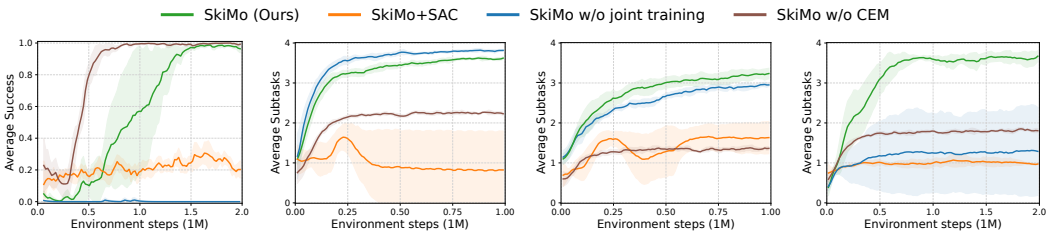
Mis-aligned Kitchen The mis-aligned target task makes the downstream learning harder, because the skill prior, which reflects offline data distribution, offers less meaningful regularization to the policy. However, Figure 4c shows that SkiMo still performs well. This demonstrates that the skill dynamics model is able to adapt to the new distribution of behaviors, which might greatly deviate from the distribution in the offline dataset.

CALVIN One of the major challenges in CALVIN is that the offline data is much more task-agnostic. Any particular sub-task transition has probability lower than 0.1% on average, resulting in a large number of plausible sub-tasks from any state. Figure 4d demonstrates that SkiMo can learn faster than the model-free baseline, SPiRL, which supports the benefit of using our skill dynamics model. Meanwhile, Dreamer performs better in CALVIN than in Kitchen because objects in CALVIN are more compactly located and easier to manipulate; thus, it becomes viable to accomplish initial sub-tasks through random exploration. Yet, it falls short in composing coherent action sequences to achieve a longer task sequence due to the lack of temporally-extended reasoning.

In summary, we show the synergistic benefit of temporal abstraction in both the policy and dynamics model. SkiMo is the only method that consistently solves the long-horizon tasks. Our results also demonstrate the importance of algorithmic design choices (e.g. skill-level planning, joint training of a model and skills) as naive combinations (SPiRL + Dreamer, SPiRL + TD-MPC) fail to learn.

4.4 Ablation Studies

Model-based vs. Model-free In Figure 5, SkiMo achieves better asymptotic performance and higher sample efficiency across all tasks than SkiMo + SAC, which uses model-free RL (SAC [44])



(a) Maze (b) Kitchen (c) Mis-aligned Kitchen (d) CALVIN

Figure 5: Learning curves of our method and ablated models. All averaged over 5 random seeds.

to train the high-level policy to select skills. Since the only difference is in the use of the skill dynamics model for planning, this suggests that the task policy can make more informative decisions by leveraging accurate long-term predictions of the skill dynamics model.

Joint training of skills and skill dynamics model "SkiMo w/o joint training" learns the latent skill space using only the VAE loss in Equation (2). Figure 5 shows that the joint training is crucial for Maze and CALVIN while the difference is marginal in the Kitchen tasks. This suggests that the joint training is essential especially in more challenging scenarios, where the agent needs to generate accurate long-term plans (for Maze) or the skills are very diverse (in CALVIN).

CEM planning As shown in Figure 5, SkiMo learns significantly better and faster in Kitchen, Mis-aligned Kitchen, and CALVIN than SkiMo w/o CEM, indicating that CEM planning can effectively find a better plan. On the other hand, in Maze, SkiMo w/o CEM learns twice as fast. We find that action noise for exploration in CEM leads the agent to get stuck at walls and corners. We believe that with a careful tuning of action noise, SkiMo can solve Maze much more efficiently.

For further ablations and discussion on skill horizon and planning horizon, see Appendix, Section A.

4.5 Long-horizon Prediction with Skill Dynamics Model

To assess the accuracy of long-term prediction of our proposed skill dynamics over flat dynamics, we visualize imagined trajectories in Appendix, Figure 10a, where the ground truth initial state and a sequence of 500 actions (50 skills for SkiMo) are given. Dreamer struggles to make accurate long-horizon predictions due to error accumulation. In contrast, SkiMo is able to reproduce the ground truth trajectory with little prediction error even when traversing through hallways and doorways. This is mainly because SkiMo allows temporal abstraction in the dynamics model, thereby enabling temporally-extended prediction and reducing step-by-step prediction error.

5 Conclusion

We propose *SkiMo*, an intuitive instantiation of saltatory model-based hierarchical RL [2] that combines skill-based and model-based RL approaches. Our experiments demonstrate that (1) a skill dynamics model reduces the long-term prediction error, improving the performance of prior model-based RL and skill-based RL; (2) it leads to temporal abstraction in both the policy and dynamics model, so the downstream RL can do efficient, temporally-extended reasoning without needing to model step-by-step planning; and (3) joint training of the skill dynamics and skill representations further improves the sample efficiency by learning skills useful to predict their consequences. We believe that the ability to learn and utilize a skill-level model holds the key to unlocking the sample efficiency and widespread use of RL agents, and our method takes a step toward this direction.

Limitations and future work While our method extracts fixed-length skills from offline data, the lengths of semantic skills may vary based on the contexts and goals. Future work can learn variable-length semantic skills to improve long-term prediction and planning. Further, although we only experimented on state-based inputs, SkiMo is a general framework that can be extended to RGB, depth, and tactile observations. Thus, we would like to apply this sample-efficient approach to real robots where the sample efficiency is crucial.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant (No.2019-0-00075, Artificial Intelligence Graduate School Program, KAIST) and National Research Foundation of Korea (NRF) grant (NRF-2021H1D3A2A03103683), funded by the Korea government (MSIT). This work was also partly supported by the Annenberg Fellowship from USC. We would like to thank Ayush Jain and Grace Zhang for help on writing, Karl Pertsch for assistance in setting up SPiRL and CALVIN, Kevin Xie for providing code of LSP, and all members of the USC CLVR lab for constructive feedback.

References

[1] S. Legg and M. Hutter. Universal intelligence: A definition of machine intelligence. *Minds and machines*, 17(4):391–444, 2007.

[2] M. Botvinick and A. Weinstein. Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369 (1655), 2014.

[3] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

[4] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

[5] K. Lu, A. Grover, P. Abbeel, and I. Mordatch. Reset-free lifelong learning with skill-space planning. In *International Conference on Learning Representations*, 2021.

[6] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Byey7n05FQ.

[7] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Neural Information Processing Systems*, 2019.

[8] A. Argenson and G. Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OMNB1G5xzd4.

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

[10] Y. Lee, S.-H. Sun, S. Somasundaram, E. S. Hu, and J. J. Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rygrBhC5tQ.

[11] Y. Lee, J. Yang, and J. J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In *International Conference on Learning Representations*, 2020.

[12] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning*, 2020.

[13] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. In *Conference on Robot Learning*, 2021.

[14] M. Dalal, D. Pathak, and R. Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. In *Neural Information Processing Systems*, 2021.

[15] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020.

[16] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim. Demonstration-guided reinforcement learning with learned skills. In *Conference on Robot Learning*, 2021.

[17] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[18] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak. Discovering and achieving goals via world models. In *Neural Information Processing Systems*, 2021.

[19] R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[20] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

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

[22] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation*, pages 763–768, 2009.

[23] O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, and S. Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

[24] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3): 263–279, 2013.

[25] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pages 4654–4663. PMLR, 2018.

[26] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, 2019.

[27] T. Shankar and A. Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, 2020.

[28] Y. Lu, Y. Shen, S. Zhou, A. Courville, J. B. Tenenbaum, and C. Gan. Learning task decomposition with ordered memory policy network. In *International Conference on Learning Representations*, 2021.

[29] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

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

[31] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[32] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning*, 2019.

[33] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei. Gti: Learning to generalize across long-horizon tasks from human demonstrations. In *Robotics: Science and Systems*, 2020.

[34] A. Mandlekar, F. Ramos, B. Boots, L. Fei-Fei, A. Garg, and D. Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *IEEE International Conference on Robotics and Automation*, 2020.

[35] B. Wu, S. Nair, L. Fei-Fei, and C. Finn. Example-driven model-based reinforcement learning for solving long-horizon visuomotor tasks. In *Conference on Robot Learning*, 2021.

[36] K. Xie, H. Bharadhwaj, D. Hafner, A. Garg, and F. Shkurti. Latent skill planning for exploration and transfer. In *International Conference on Learning Representations*, 2020.

[37] D. Shah, A. T. Toshev, S. Levine, and brian ichter. Value function spaces: Skill-centric state abstractions for long-horizon reasoning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vgqS1vkkCbE.

[38] R. S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

[39] T. Shankar, S. Tulsiani, L. Pinto, and A. Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2020.

[40] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[41] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, 2020.

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

[43] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022.

[44] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[46] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. P. Lillicrap, and M. A. Riedmiller. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[47] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

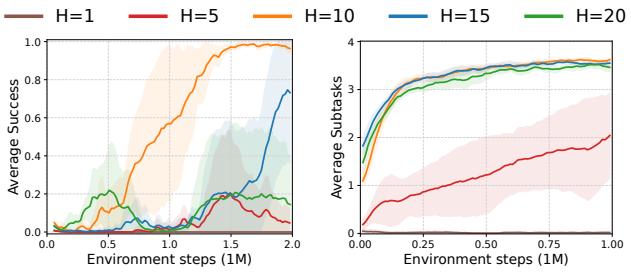
[48] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, 2019.

[49] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893, 2018.

A Further Ablations

We include additional ablations on the Maze and Kitchen tasks to further investigate the influence of skill horizon H and planning horizon N, which is important for skill learning and planning.

A.1 Skill Horizon

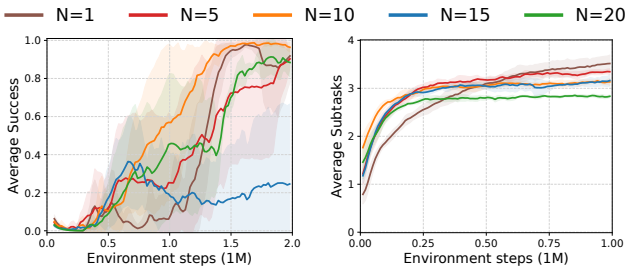


(a) Maze (b) Kitchen

Figure 6: Ablation analysis on skill horizon H.

In both Maze and Kitchen, we find that a too short skill horizon ($H = 1, 5$) unable to yield sufficient temporal abstraction. A longer skill horizon ($H = 15, 20$) has little influence in Kitchen, but it makes the downstream performance much worse in Maze. This is because with too long-horizon skills, a skill dynamics prediction becomes too difficult and stochastic; and composing multiple skills can be not as flexible as short-horizon skills. The inaccurate skill dynamics makes long-term planning harder, which is already a major challenge in maze navigation.

A.2 Planning Horizon

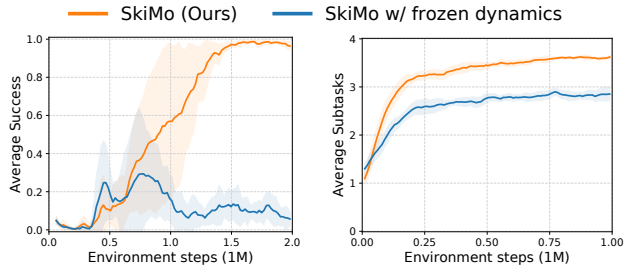


(a) Maze (b) Kitchen

Figure 7: Ablation analysis on planning horizon N.

In Figure 7b, we see that short planning horizon makes learning slower in the beginning, because it does not effectively leverage the skill dynamics model to plan further ahead. Conversely, if the planning horizon is too long, the performance becomes worse due to the difficulty in modeling every step accurately. Indeed, the planning horizon 20 corresponds to 200 low-level steps, while the episode length in Kitchen is 280, demanding the agent to make plan for nearly the entire episode. The performance is not sensitive to intermediate planning horizons. On the other hand, the effect of the planning horizon differs in Maze due to distinct environment characteristics. We find that very long planning horizon (eg. 20) and very short planning horizon (eg. 1) perform similarly in Maze (Figure 7a). This could attribute to the former creates useful long-horizon plans, while the latter avoids error accumulation altogether.

A.3 Fine-Tuning Model



(a) Maze (b) Kitchen

Figure 8: Ablation analysis on fine-tuning the model.

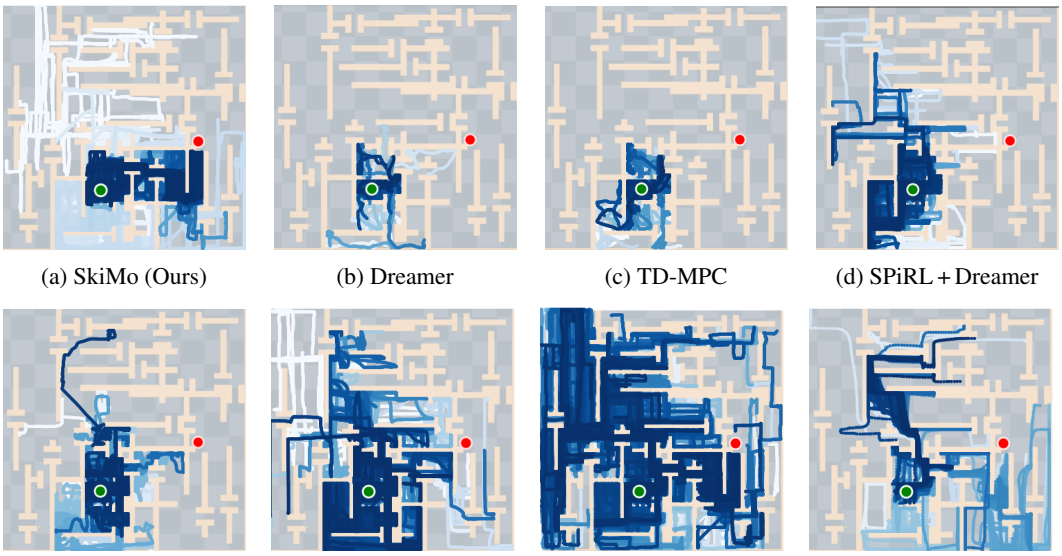
We freeze the skill dynamics model together with the state encoder to gauge the effect of fine-tuning after pre-training. Figure 8 show that without fine-tuning the model, the agent performs worse due to the discrepancy between distributions of the offline data and the downstream task. We hypothesize that fine-tuning is necessary when the agent needs to adapt to a different task and state distribution after pre-training.

B Qualitative Analysis on Maze

B.1 Exploration and Exploitation

To gauge the agent's ability of exploration and exploitation, we visualize the replay buffer for each method in Figure 9. In this visualization, we represent early trajectories in the replay buffer with light blue dots and recent trajectories with dark blue dots. In Figure 9a, the replay buffer of SkiMo (ours) contains early explorations that span to most corners in the maze. After it finds the goal, it exploits this knowledge and commits to paths that are between the start location and the goal (in dark blue). This explains why our method can quickly learn and consistently accomplish the task. Dreamer and TD-MPC only explore a small fraction of the maze, because they are prone to get stuck at corners or walls without guided exploration from skills and skill priors. SPiRL + Dreamer, SPiRL + TD-MPC and SkiMo w/o joint training explore better than Dreamer and TD-MPC, but all fail to find the goal. This is because without the joint training of the model and policy, the skill space is only optimized for action reconstruction, not for planning, which makes long-horizon exploration and exploitation harder.

On the other hand, SkiMo + SAC and SPiRL are able to explore the most portion of the maze, but comparatively the coverage is too wide to enable efficient learning. That is, even after the agent finds the goal through exploration, it continues to explore and does not exploit this experience to accomplish the task consistently (darker blue). This could attribute to the difficult long-horizon credit assignment problem which makes policy learning slow, and the reliance on skill prior which encourages exploration. On the contrary, our skill dynamics model effectively absorbs prior experience to generate goal-achieving imaginary rollouts for the actor and critic to learn from, which makes task learning more efficient. In essence, we find the skill dynamics model useful in guiding the agent explore coherently and exploit efficiently.



(a) SkiMo (Ours) (b) Dreamer (c) TD-MPC (d) SPiRL + Dreamer

(e) SPiRL + TD-MPC (f) SkiMo w/o Joint Training (g) SkiMo + SAC (h) SPiRL

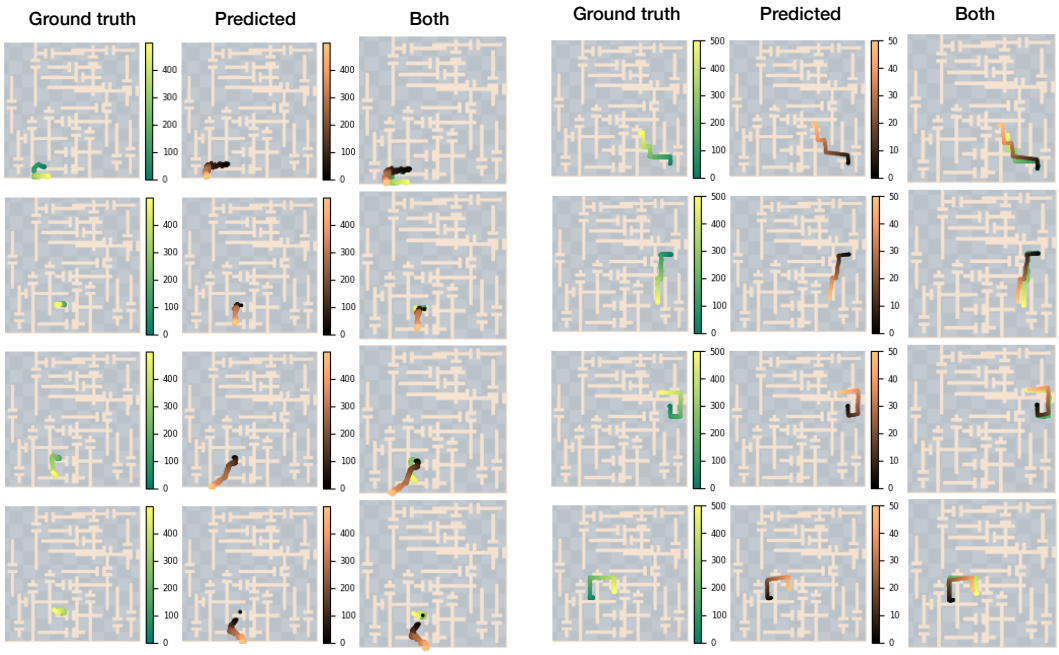
Figure 9: Exploration and exploitation behaviors of our method and baseline approaches. We visualize trajectories in the replay buffer at 1.5M training steps in blue (light blue for early trajectories and dark blue for recent trajectories). Our method shows wide coverage of the maze at the early stage of training, and fast convergence to the solution.

B.2 Long-horizon Prediction

To compare the long-term prediction ability of the skill dynamics and flat dynamics, we visualize imagined trajectories by sampling trajectory clips of 500 timesteps from the agent's replay buffer (the maximum episode length in Maze is 2,000), and predicting the latent state 500 steps ahead (which

will be decoded using the observation decoder) given the initial state and 500 ground-truth actions (50 skills for SkiMo). The similarity between the imagined trajectory and the ground truth trajectory can indicate whether the model can make accurate predictions far into the future, producing useful imaginary rollouts for policy learning and planning.

SkiMo is able to reproduce the ground truth trajectory with little prediction error even when traversing through hallways and doorways while Dreamer struggles to make accurate long-horizon predictions due to error accumulation. This is mainly because SkiMo allows temporal abstraction in the dynamics model, thereby enabling temporally-extended prediction and reducing step-by-step prediction error.



(a) Dreamer (b) SkiMo (Ours)

Figure 10: Prediction results of 500 timesteps using a flat single-step model (a) and skill dynamics model (b), starting from the ground truth initial state and 500 actions (50 skills for SkiMo). The predicted states from the flat model deviate from the ground truth trajectory quickly while the prediction of our skill dynamics model has little error.

C Implementation Details

C.1 Computing Resources

Our approach and all baselines are implemented in PyTorch [45]. All experiments are conducted on a workstation with an Intel Xeon E5-2640 v4 CPU and a NVIDIA Titan Xp GPU. Pre-training of the skill policy and skill dynamics model takes around 10 hours. Downstream RL for 2M timesteps takes around 18 hours. The policy and model update frequency is the same over all algorithms but Dreamer [3]. Since only Dreamer trains on primitive actions, it has 10 times more frequent model and policy updates than skill-based algorithms, which leads to slower training (about 52 hours).

C.2 Algorithm Implementation Details

For the baseline implementations, we use the official code for SPiRL and re-implemented Dreamer and TD-MPC in PyTorch, which are verified on DM control tasks [46]. The table below (Table 1) compares key components of SkiMo with model-based and skill-based baselines and ablated methods.

Dreamer [3] We use the same hyperparameters with the official implementation.

Table 1: Comparison to prior work and ablated methods.

Method	Skill-based	Model-based	Joint training
Dreamer [3] and TD-MPC [4]	✗	✓	✗
SPiRL [12]	✓	✗	✗
SPiRL + Dreamer and SPiRL + TD-MPC	✓	✓	✗
SkiMo w/o joint training	✓	✓	✗
SkiMo + SAC	✓	✗	✓
SkiMo (Ours) and SkiMo w/o CEM	✓	✓	✓

TD-MPC [4] We use the same hyperparameters with the official implementation, except that we do not use the prioritized experience replay [47]. The same implementation is used for the SPiRL + TD-MPC baseline and our method with only minor modification.

SPiRL [12] We use the official implementation of the original paper and use the hyperparameters suggested in the official implementation.

SPiRL + Dreamer [12] We use our implementation of Dreamer and simply replace the action space with the latent skill space of SPiRL. We use the same pre-trained SPiRL skill policy and skill prior networks with the SPiRL baseline. Initializing the high-level downstream task policy with the skill prior, which is critical for downstream learning performance [12], is not possible due to the policy network architecture mismatch between Dreamer and SPiRL. Thus, we only use the prior divergence to regularize the high-level policy instead. Directly pre-train the high-level policy did not lead to better performance, but it might have worked better with more tuning.

SPiRL + TD-MPC [4] Similar to SPiRL + Dreamer, we use our implementation of TD-MPC and replace the action space with the latent skill space of SPiRL. The initialization of the task policy is also not available due to the different architecture used for TD-MPC.

SkiMo (Ours) The skill-based RL part of our method is inspired by Pertsch et al. [12] and the model-based component is inspired by Hansen et al. [4] and Hafner et al. [3]. We elaborate our skill and skill dynamics learning in Algorithm 1, planning algorithm in Algorithm 2, and model-based RL in Algorithm 3. Table 2 lists the all hyperparameters that we used.

Algorithm 1 SkiMo RL *(skill and skill dynamics learning)*

Require: $\mathcal{D}$: offline task-agnostic data
1: Randomly initialize θ, ψ
2: $\psi^- \leftarrow \psi$ ▷ initialize target network
3: **for** each iteration **do**
4: Sample mini-batch $B = (\mathbf{s}, \mathbf{a})_{(0:NH)} \sim \mathcal{D}$
5: $[\theta, \psi] \leftarrow [\theta, \psi] - \lambda_{[\theta,\psi]} \nabla_{[\theta,\psi]} \mathcal{L}(B)$ ▷ $\mathcal{L}$ from Equation (5)
6: $\psi^- \leftarrow (1-\tau)\psi^- + \tau\psi$ ▷ update target network
7: **end for**
8: **return** θ, ψ, ψ^-

Algorithm 2 SkiMo RL *(CEM planning)*

Require: θ, ψ, ϕ : learned parameters, $\mathbf{s}_t$: current state
$\mu^0, \sigma^0 \leftarrow \mathbf{0}, \mathbf{1}$ ▷ initialize sampling distribution
for $i = 1, ..., N_{\text{CEM}}$ **do**
  Sample N_{sample} trajectories of length N from $\mathcal{N}\big(\mu^{i-1}, (\sigma^{i-1})^2\big)$ ▷ sample skill sequences from normal distribution
  Sample N_π trajectories of length N using π_ϕ, D_ψ ▷ sample skill sequences via imaginary rollouts
  Estimate N-step returns of $N_{\text{sample}} + N_\pi$ trajectories using R_ϕ, Q_ϕ
  Compute μ^i, σ^i with top-k return trajectories ▷ update parameters for next iteration
end for
Sample a skill $\mathbf{z} \sim \mathcal{N}\big(\mu^{N_{\text{CEM}}}, (\sigma^{N_{\text{CEM}}})^2\big)$
return $\mathbf{z}$

Algorithm 3 SkiMo RL *(downstream task learning)*

Require: θ, ψ, ψ^- : pre-trained parameters
$\mathcal{B} \leftarrow \emptyset$ ▷ initialize replay buffer
Randomly initialize ϕ
$\phi^- \leftarrow \phi$ ▷ initialize target network
$\pi_\phi \leftarrow p_\theta$ ▷ initialize task policy with skill prior
for not converged **do**
  $t \leftarrow 0, s_0 \sim \rho_0$ ▷ initialize episode
  for episode not done **do**
    $\mathbf{z}_t \sim \text{CEM}(\mathbf{s}_t)$ ▷ MPC with CEM planning in Algorithm 2
    $\mathbf{s}, r_t \leftarrow \mathbf{s}_t, 0$
    for H steps **do**
      $\mathbf{s}, r \leftarrow \text{ENV}(\mathbf{s}, \pi_\theta^L(E_\psi(\mathbf{s}), \mathbf{z}_t))$ ▷ rollout low-level skill policy
      $r_t \leftarrow r_t + r$
    end for
    $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{s}_t, \mathbf{z}_t, r_t)$ ▷ collect H-step environment interaction
    $t \leftarrow t + H$
    $\mathbf{s}_t \leftarrow \mathbf{s}$
    Sample mini-batch $B = (\mathbf{s}, \mathbf{z}, r)_{(0:N)} \sim \mathcal{B}$
    $[\phi, \psi] \leftarrow [\phi, \psi] - \lambda_{[\phi,\psi]} \nabla_{[\phi,\psi]} \mathcal{L}'_{\text{REC}}(B)$ ▷ $\mathcal{L}'_{\text{REC}}$ from Equation (6)
    $\phi_\pi \leftarrow \phi_\pi - \lambda_\phi \nabla_{\phi_\pi} \mathcal{L}_{\text{RL}}(B)$ ▷ $\mathcal{L}_{\text{RL}}$ from Equation (7). Update only policy parameters
    $\psi^- \leftarrow (1-\tau)\psi^- + \tau\psi$ ▷ update target network
    $\phi^- \leftarrow (1-\tau)\phi^- + \tau\phi$ ▷ update target network
  end for
end for
return ψ, ϕ

C.3 Environments and Data

Maze [42, 16] Since our goal is to leverage offline data collected from diverse tasks in *the same environment*, we use a variant of the maze environment [42], suggested in Pertsch et al. [16]. The maze is of size 40×40; an initial state is randomly sampled near a pre-defined region (the green circle in Figure 3a); and the goal position is fixed shown as the red circle in Figure 3a. The observation consists of the agent's 2D position and velocity. The agent moves around the maze by controlling the continuous value of its (x, y) velocity. The maximum episode length is 2,000 but an episode is also terminated when the agent reaches the circle around the goal with radius 2. The reward of 100 is given at task completion. We use the offline data of 3,046 trajectories, collected from randomly sampled start and goal state pairs from Pertsch et al. [16]. Thus, the offline data and downstream task share the same environment, but have different start and goal states (i.e. different tasks). This data can be used to extract short-horizon skills like navigating hallways or passing through narrow doors.

Kitchen [32, 42] The 7-DoF Franka Panda robot arm needs to perform four sequential tasks (open microwave, move kettle, turn on bottom burner, and flip light switch). The agent has a 30D observation space (11D robot proprioceptive state and 19D object states), which removes a constant 30D goal state in the original environment, and 9D action space (7D joint velocity and 2D gripper velocity). The agent receives a reward of 1 for every sub-task completion. The episode length is 280 and an episode also ends once all four sub-tasks are completed. The initial state is set with a small noise in every state dimension. We use 603 trajectories collected by teleoperation from Gupta et al. [32] as the offline task-agnostic data. The data involves interaction with all seven manipulatable objects in the environment, but during downstream learning the agent needs to execute an unseen sequence of four subtasks. That is, the agent can transfer a rich set of manipulation skills, but needs to recombine them in new ways to solve the task.

Mis-aligned Kitchen [16] The environment and task-agnostic data are the same with **Kitchen** but we use the different downstream task (open microwave, flip light switch, slide cabinet door, and open hinge cabinet, as illustrated in Figure 3c). This task ordering is not aligned with the sub-task transition probabilities of the task-agnostic data, which leads to challenging exploration following the prior from data. This is because the transition probabilities in the Kitchen human-teleoperated dataset are not uniformly distributed; instead certain transitions are more likely than others. For example, the first transition in our target task — from opening the microwave to flipping the light switch — is very unlikely to be observed in the training data. This simulates the real-world scenario where the large offline dataset may not be meticulously curated for the target task.

CALVIN [43] We adapt the CALVIN environment [43] for long-horizon learning with the state observation. The CALVIN environment uses a Franka Emika Panda robot arm with 7D end-effector pose control (relative 3D position, 3D orientation, 1D gripper action). The 21D observation space consists of the 15D proprioceptive robot state and 6D object state. We use the teleoperated play data (Task D→Task D) of 1,239 trajectories from Mees et al. [43] as our task-agnostic data. The agent receives a sparse reward of 1 for every sub-task completion in the correct order. The episode length is 360 and an episode also ends once all four sub-tasks are completed. In data, there exist 34 available target sub-tasks, and each sub-task can transition to any other sub-task, which makes any transition probability lower than 0.1% on average. Most of the subtask transitions in our downstream task occupy less than 0.08% of all transitions in the CALVIN task-agnostic dataset.

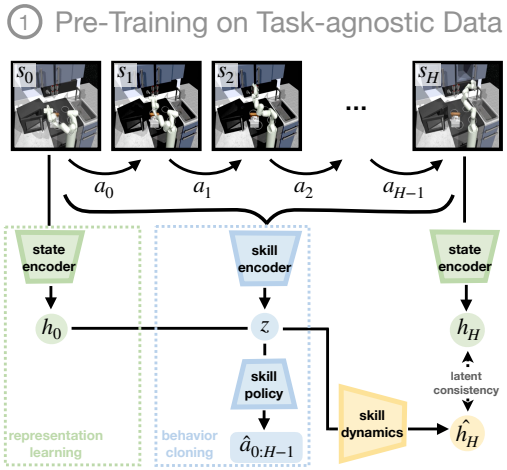
D Application to Real Robot Systems

Our algorithm is designed to be applied on real robot systems by improving sample efficiency of RL using a temporally-abstracted dynamics model. Throughout the extensive experiments in simulated robotic manipulation environments, we show that our approach achieves superior sample efficiency over prior skill-based and model-based RL, which gives us strong evidence for the application to real robot systems. Especially in Kitchen and CALVIN, our approach improves the sample efficiency of learning long-horizon manipulation tasks with a 7-DoF Franka Emika Panda robot arm. Our approach consists of three phases: (1) task-agnostic data collection, (2) skills and skill dynamics model learning, and (3) downstream task learning. In each of these phases, our approach can be applied to physical robots:

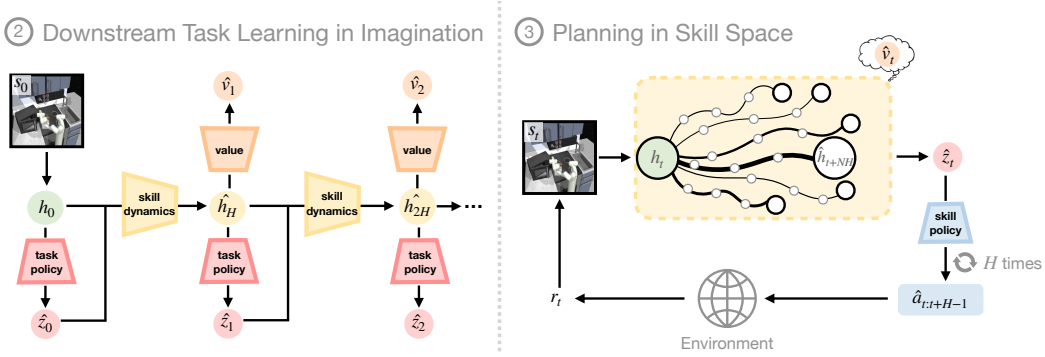
Task-agnostic data collection Our approach is designed to fully leverage task-agnostic data without any reward or task annotation. In addition to extracting skills and skill priors, we further learn a skill dynamics model from this task-agnostic data. Maximizing the utility of task-agnostic data is critical for real robot systems as data collection with physical robots itself is very expensive. Our method does not require any manual labelling of data and simply extracts skills, skill priors, and skill dynamics model from raw states and actions, which makes our method scalable.

Pre-training of skills and skill dynamics model Our approach trains the skill policy, skill dynamics model, and skill prior from the offline task-agnostic dataset, without requiring any additional real-world robot interactions.

Downstream task learning The goal of our work is to leverage skills and skill dynamics model to allow for more efficient downstream learning, i.e., requires less interactions of the agent with



(a) In pretraining, SkiMo leverages offline task-agnostic data to extract skill dynamics and a skill repertoire. Unlike prior works that keep the model and skill policy training separate, we propose to *jointly* train them to extract a skill space that is conducive to plan upon.



(b) In downstream RL, we learn a high-level task policy in the skill space (skill-based RL) and leverage the skill dynamics model to generate imaginary rollouts for policy optimization and planning (model-based RL).

Figure 11: Illustration of our algorithm.

the environment for training the policy. This is especially important on real robot systems where a robot-environment interaction is slow, dangerous, and costly. Our approach directly addresses this concern by learning a policy from imaginary rollouts rather than actual environment interactions. Also, all sort of collected data can help improve the skill dynamics model, which leads to more accurate imagination and policy learning.

In summary, we believe that our approach can be applied to real-world robot systems with only minor modifications.

Table 2: SkiMo hyperparameters.

Hyperparameter	Value		
	Maze	FrankaKitchen	CALVIN
Model architecture			
# Layers of $O_\theta, p_\theta, \pi_\theta^L, E_\psi, D_\psi, \pi_\phi, R_\phi, Q_\phi$		5	
Activation funtion		elu	
Hidden dimension	128	128	256
State embedding dimension	128	256	256
Skill encoder (q_θ)	5-layer MLP	LSTM	LSTM
Skill encoder hidden dimension		128	
Pre-training			
Pre-training batch size		512	
# Training mini-batches per update		5	
Model-Actor joint learning rate ($\lambda_{[\theta,\psi]}$)		0.001	
Encoder KL regularization (β)		0.0001	
Reconstruction loss coefficient (λ_{O})		1	
Consistency loss coefficient (λ_{L})		2	
Low-level actor loss coefficient (λ_{BC})		2	
Planning discount (ρ)		0.5	
Skill prior loss coefficient (λ_{SP})		1	
Downstream RL			
Model learning rate		0.001	
Actor learning rate		0.001	
Skill dimension		10	
Skill horizon (H)		10	
Planning horizon (N)	10	3	1
Batch size	128	256	256
# Training mini-batches per update		10	
State normalization	True	False	False
Prior divergence coefficient (α)	1	0.5	0.1
Alpha learning rate	0.0003	0	0
Target divergence	3	N/A	N/A
# Warm up step	50,000	5,000	5,000
# Environment step per update		500	
Replay buffer size		1,000,000	
Target update frequency		2	
Target update tau (τ)		0.01	
Discount factor (γ)		0.99	
Reward loss coefficient (λ_{R})		0.5	
Value loss coefficient (λ_{Q})		0.1	
CEM			
CEM iteration (N_{CEM})		6	
# Sampled trajectories (N_{sampled})		512	
# Policy trajectories (N_π)		25	
# Elites (k)		64	
CEM momentum		0.1	
CEM temperature		0.5	
Maximum std		0.5	
Minimum std		0.01	
Std decay step	100,000	25,000	25,000
Horizon decay step	100,000	25,000	25,000