Offline Policy Learning via Skill-step Abstraction for Long-horizon Goal-Conditioned Tasks

Donghoon Kim\textsuperscript{1}, Minjong Yoo\textsuperscript{2} and Honguk Woo\textsuperscript{1,2}\textsuperscript{*}
\textsuperscript{1}Department of Artificial Intelligence, Sungkyunkwan University
\textsuperscript{2}Department of Computer Science and Engineering, Sungkyunkwan University
\{qwef523, mjyoo2, hwoo\}@skku.com,

Abstract

Goal-conditioned (GC) policy learning often faces a challenge arising from the sparsity of rewards, when confronting long-horizon goals. To address the challenge, we explore skill-based GC policy learning in offline settings, where skills are acquired from existing data and long-horizon goals are decomposed into sequences of near-term goals that align with these skills. Specifically, we present an ‘offline GC policy learning via skill-step abstraction’ framework (GLvSA) tailored for tackling long-horizon GC tasks affected by goal distribution shifts. In the framework, a GC policy is progressively learned offline in conjunction with the incremental modeling of skill-step abstractions on the data. We also devise a GC policy hierarchy that not only accelerates GC policy learning within the framework but also allows for parameter-efficient fine-tuning of the policy. Through experiments with the maze and Franka kitchen environments, we demonstrate the superiority and efficiency of our GLvSA framework in adapting GC policies to a wide range of long-horizon goals. The framework achieves competitive zero-shot and few-shot adaptation performance, outperforming existing GC policy learning and skill-based methods.

1 Introduction

In goal-oriented environments, task outcomes are directly related to the achievement of specific goals. This poses a challenge for goal-conditioned (GC) policies when trained through reinforcement learning (RL) algorithms, particularly for tasks with long-horizon goals [Ma et al., 2022; Lu et al., 2020; Chebotar et al., 2021]. The challenge is primarily attributed to the reward sparsity associated with achieving long-horizon goals. Meanwhile, skill-based RL [Pertsch et al., 2021; Shi et al., 2023; Jiang et al., 2022] has exhibited great potential in addressing long-horizon and sparse reward tasks by leveraging skills. These skills are temporally abstracted patterns of expert behaviors spanning over multiple timesteps and they can be acquired offline from expert datasets through imitation learning methods.

To address the challenge associated with long-horizon goal-reaching tasks, we investigate skill-based RL in the context of offline GC policy learning. Incorporating the learned skills into the procedure of learning a long-horizon GC policy looks promising, as a long-horizon goal can be decomposed into near-term subgoals that can be achievable by the skills. However, the nature of imitation-based skill acquisition techniques brings certain issues when they are applied to GC policy learning via RL (GCRL), particularly in scenarios with diverse and varied goals. Imitation-based skills, acquired offline from existing trajectory data, tend to specify a constrained range of achievable goals for GCRL. Thus, this can restrict the adaptation capabilities of GC policies with respect to achievable goals, confining them to the limits of the offline data. This limitation often results in diminished performance in GC tasks, especially those with goal distribution shifts; such shifts occur when there is a discrepancy in the distribution of goals between the dataset used for training and the target goals for evaluation.

In this work, we explore skill-based GC policy learning approaches for long-horizon GC tasks, particularly those affected by goal distribution shifts, by presenting a framework for ‘offline GC policy learning via skill-step abstraction’ (GLvSA). The GLvSA framework harnesses data to obtain skill-based GC policies in offline settings, aiming at enhancing their adaptability and efficiency in the environment with long-horizon GC tasks. At its core, the framework employs an iterative process for joint learning of a skill-step model and a GC policy. The model encapsulates expert behaviors in the embeddings of temporally abstracted skills and represents environment dynamics in a skill-aligned latent space. This concept of skill-step abstraction in GLvSA involves distilling the complexity of long-horizon GC tasks into higher-level, distinct segments (i.e., a set of consecutive action sequences). Each segment or skill-step condenses a series of actions into a more manageable form, facilitating a clearer understanding of the GC task and environment dynamics. In each iteration, the model is employed to generate skill-step rollouts, weaving distinct trajectories from the data to derive new imaginary ones. These derived trajectories are then used to acquire additional plausible skills, which in turn, broaden the range of achievable goals by GC policies grounded in these skills. This iterative process allows for continual model improvement that

\textsuperscript{*}Corresponding author.
addresses the limitation inherently associated with conventional imitation-based skill learning approaches, thereby enhancing the adaptability of skill-based GC policies.

To accelerate both offline learning and online fine-tuning of GC policies, the framework incorporates a modular structure within the policy hierarchy. This design is centered on generating near-term goals, which are closely aligned with the acquired skills (i.e., skill-step goals). The modular structure also enables parameter-efficient policy updates for GC tasks with goal distribution shifts, through separate module optimization for skill-step goal generation. This focused optimization streamlines the adaptation process, making it more effective and efficient (i.e., through few-shot fine-tuning), particularly in environments where goals are constantly evolving.

Through various comparison and ablation experiments with the maze and Franka kitchen environments, we demonstrate that the GC policies learned via our GLvSA framework achieve competitive performance for both the zero-shot evaluation and few-shot adaptation scenarios, compared to several state-of-the-art GC policy learning methods (i.e., [Ghosh et al., 2021; Yang et al., 2021]) and skill-based RL methods (i.e., [Pertsch et al., 2021; Shi et al., 2023]). The contributions of our work are summarized as follows.

- We present GLvSA, a novel offline policy learning framework using a skill-step model, to improve the adaptability of GC policies upon goal distribution shifts.
- We devise an offline GC policy training scheme, where a skill-step model and a GC policy are jointly learned from data and progressively refined via an iterative process.
- We develop a modular GC policy structure to accelerate both policy learning and fine-tuning.
- We validate the effectiveness of GLvSA with zero-shot and few-shot adaptation scenarios, by applying it to navigation and robot manipulation tasks, each presenting distinct challenges in terms of goal distribution shifts.
- Our GLvSA stands as the first to incorporate offline skill and model learning in the realm of GCRL.

## 2 Preliminaries

**GCRL** addresses the problem of goal-augmented Markov decision processes (GA-MDPs) $\mathcal{M}_G$, which are represented by an MDP $\mathcal{M} = (S, \mathcal{A}, \mathcal{P}, R, \gamma)$ and an additional tuple of a goal space, a goal distribution, and a state-goal mapping function $(G, p_g, \Phi)$ [Liu et al., 2022]. In a GA-MDP $\mathcal{M}_G$, $S$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{P} : S \times \mathcal{A} \times S \rightarrow [0, 1]$ is a transition probability, and $\gamma$ is a discount factor. A reward function $R : S \times \mathcal{A} \times G \rightarrow \mathbb{R}$ is determined by a given goal and is formed as $R : S \times \mathcal{A} \times G \rightarrow \mathbb{R}$ that represents sparse rewards; i.e.,

$$R(s, a, g) = 1(\Phi(s' \sim \mathcal{P}(s|s, a)) = g)$$  \hspace{1cm} (1)

where $\Phi : S \rightarrow \mathcal{G}$ maps states $s$ to a goal $g \in \mathcal{G}$. This reward structure can be used for RL-based GC policy learning. A GC policy such as $\pi : S \times \mathcal{G} \times \mathcal{A} \rightarrow [0, 1]$ is learned by

$$\mathbb{E}_{g \sim p_g} \left[ \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t \sim \pi(\cdot|s_t, g), g) \right]$$  \hspace{1cm} (2)

In principle, GCRL facilitates multi-task learning by treating distinct tasks as goals, and it also addresses multi-stage complex tasks by incorporating the composite goal of multiple subtasks into its goal representations.

**Skill-based RL** is a hierarchical learning paradigm [Sutton et al., 1999] that leverages the notion of skills, which refers to reusable units of prior knowledge and useful behavior patterns extracted from the data. This hierarchical RL aims to accelerate the process of policy learning through supervising low-level policies on skills, particularly when tackling long-horizon, sparse reward tasks. In [Pertsch et al., 2021], a skill corresponds to an abstracted $H$-step consecutive action sequence and is represented in a variational auto-encoder (VAE)-based embedding space. Accordingly, skill embeddings $z$ are learned on $H$-step sub-trajectories $\tau^{sub}$ such that $\tau_{i:i+H} = (s_i, \ldots, s_{i+H}, a_i, \ldots, a_{i+H-1}) = (s_{0:H}, a_{0:H-1})$ (3) that are sampled from trajectories $\tau$ in the data. These embeddings are obtained offline by the skill loss $\mathcal{L}_{skill}$ such as

$$\mathbb{E}_{\tau^{sub} \sim \tau} \left[ \sum_{t=0}^{H-1} (\pi^L_g(s_t, z) - a_t)^2 + \beta \cdot K\ell(Q_{\theta}(z|\tau^{sub})||P(z)) \right]$$  \hspace{1cm} (4)

where $K\ell$ denotes the KL divergence and $P(z) \sim N(0, I)$ is a prior distribution of skills. Here, a skill encoder $q_{\theta}$ converts action sequences to skill embeddings $z$ and a low-level policy $\pi^L_g$ serves as a skill decoder, producing an action for a state-skill pair. By this encoding-decoding structure, skills are captured from diverse experiences and establish building blocks that enable an agent to tackle complex tasks effectively.

Unlike the conventional RL approach that directly selects actions, in skill-based RL, the agent’s GC policy $\pi(a|s, g)$ is composed of two modules, a low-level policy $\pi^L_g(a|s, z)$ and a skill-based high-level policy $\pi^Z_g(z|s, g)$, i.e.,

$$\pi(a|s, g) = \pi^L_g(a|s, z) + \pi^Z_g(z|s, g).$$  \hspace{1cm} (5)

In the hierarchy, the behavior of $\pi^Z_g$ adheres to the constraints of a prior $p_\theta$, which regularizes the policy search space, following the entropy maximization RL scheme in [Haarnoja et al., 2018]. Thus, the learning objective of a GC policy in Eq. (2) can be rewritten using the two policy modules $\pi^L_g, \pi^Z_g$ and the prior $p_\theta$ optimized by

$$\mathbb{E}_{g \sim p_g} \left[ \sum_{t=0}^{T-1} \sum_{k=1}^{(t+1)H-1} \left[ R(s_k, a_k \sim \pi^L_g(\cdot|s_k, z_{tH}), g) - K\ell(\pi^Z_g(z_{tH}|s_{iH}, s_{tH})) \right] \right]$$  \hspace{1cm} (6)

where $z_{tH} \sim \pi^Z_g(\cdot|s_{tH}, g)$ and $H$ is the skill-step horizon.

1For notation consistency as in Table 1, we use the model parameters in subscripts for learnable modules from this point forward. Different subscripts are uniquely meaningful in our approach but do not apply broadly in other related works.
3 Our Framework

To address the challenge of tackling long-horizon goal-reaching tasks, we incorporate the skill-based RL strategy into the offline learning process for GC policies. We achieve this integration in our GLvSA framework, in which a skill-step model encapsulating skills and environment dynamics at the skill abstraction level, and a GC policy based on the model are jointly and iteratively learned during the offline training phase. In the online adaptation phase, the GC policy learned offline can be evaluated in a zero-shot manner without any policy updates or it can be parameter-efficiently fine-tuned through few-shot learning. Adaptation is required for the cases when confronting downstream tasks with significant goal distribution shifts. Figure 1 depicts these two phases of GLvSA: the offline training phase and the adaptation phase. The offline training phase consists of two tasks using the current dataset: joint learning of the skill-step model and GC policy, and model-guided rollouts. During joint learning, modules for the skill-step model and GC policy are updated using trajectories from the dataset. Subsequently, the model-guided rollouts produce trajectories, which are added to the dataset for use in future iterations. For downstream GC tasks, the GC policy can be directly deployed and evaluated in a “zero-shot” manner. It can be also fine-tuned via “few-shot” updates in a parameter-efficient way to better adapt to a specific task, particularly for the cases subject to goal distribution shifts. Our GC policy hierarchy employs a modular scheme for the policy network. This involves dividing the network into three distinct modules, each serving a specific inference function. The modular scheme allows for parameter-efficient policy adaptation by confining policy updates within one for skill-step goal generation, thereby maintaining the stability of the other modules.

Table 1 lists the notations of modules in GLvSA. They include the skill-based GC policy with low-level policy (skill decoder), high-level policy (skill policy), and inverse skill-step dynamics modules, as well as the skill-step model with dual dynamics, skill encoder, and skill prior modules. These also include a state encoder-decoder, by which the latent state space can align with the skills. Each module subscript denotes the role in policy adaptation. ψ, θ, and φ indicate fine-tuned, frozen, and unused modules, respectively, in policy adaptation; e.g., q_φ and D_φ are used only for the offline training.

3.1 Skill-step Model Learning

To broaden the range of skills and goals beyond the scope of merely imitating individual trajectories in the offline dataset, we employ the iteratively refined skill-step model, involving skill acquisition, model refinement, policy updates, and trajectory generation. From the iterations onwards, we use the augmented dataset comprising both the original trajectories and those derived from prior iterations’ rollouts. In each iteration, all the modules in Table 1 are jointly optimized, using the losses in Eq. (11), as depicted in Figure 2. This approach bridges distinct trajectories incrementally in the latent state space, enhancing skill and goal coverage.

**Skill Acquisition**

We use conditional-β-VAE networks [Sohn et al., 2015; Higgins et al., 2017] to obtain skill embeddings z, following a similar approach in [Pertsch et al., 2021]. A skill encoder q_φ transforms a sub-trajectory τ^{sub} in Eq. (3) into an embedding z using Eq. (4), while a skill decoder π^θ̂ reconstructs the sub-trajectory from z. The skill encoder is also updated further through Eq. (8). Furthermore, for minibatches B = \{τ^{sub}_i\}_{i=1}^N sampled for the dataset D, we obtain a skill prior p_θ that infers the distribution of skills for a given latent state by optimizing the skill prior loss \( L_{\text{prior}} \) defined as

\[
\mathbb{E}_{B \sim \mathcal{D}} \left[ KL(p_\theta(z|h_i) \| q_\phi(z|\tau^{sub}) \right]
\]

where sg is a stop-gradient function and h_i = sg(E_θ(s_t)) is a latent state for the first state s_t in sub-trajectory \( \tau^{sub} \). The skill prior p_θ facilitates rollouts in the latent state space.

**Model Refinement**

To enable rollouts in the latent state space, we jointly optimize state embeddings and two dynamics modules such as

![Figure 1: GLvSA framework](image1)

![Figure 2: Learning process in GLvSA framework](image2)
skill-step dynamics $P_\theta$ and flat dynamics $P_\phi$, similar to [Shi et al., 2023]. A state encoder $E_\theta$ represents states in a state embedding space $\mathbb{H}$ that can be reconstructed by a state decoder $D_\phi$. Given a state embedding and a skill, $P_\phi$ predicts the next state embedding by an execution of the single timestep skill, while $P_\theta$ predicts the state by an execution of the $H$-timestep of skill. We also obtain an inverse skill-step dynamics module $P_\theta^{-1}$, which infers a skill for a pair of starting latent state $h_0$ and skill-step latent state $h_{t+H}$. These modules (listed in the ‘Model’ row of Table 1) are jointly learned by the model loss $L_{\text{model}}$ such as

$$
E_{B \sim \mathcal{D}} \left[ \sum_{k=0}^{H-1} \left( (D_\phi(h_{t+k}) - s_{t+k})^2 + (P_\phi(h_{t+k}, z) - \hat{h}_{t+k+1})^2 \right) \right]
$$

$$
+ \sum_{k=0}^{H-1} \left( (P_\phi(h_{t}, z) - \hat{h}_{t+1})^2 + KL(q(z || P_\theta^{-1}(z|h_t, h_{t+H}))) \right)
$$

where $h_t = E_\phi(s_t)$, $\hat{h}_t = E_\phi(s_t)$, $\theta$ is slowly copied from $\theta$, and $z = q_\psi(\tau_{\text{subj}})$ is the skill embedding. Note that the skill encoder $q_\psi$ is updated by this joint optimization in addition to Eq. (5). By this, the latent state space ($h \in \mathbb{H}$) becomes closely aligned with the skills, thus making it easier to stitch sub-trajectories of different trajectories.

Trajectory Generation

Figure 3 illustrates model-guided rollouts. To derive new trajectories in each iteration, we first randomly select branching states from an existing trajectory (a sequence of state-action pairs). Branching states are used as initial states for the model-guided rollouts. For each branching state, we sample a skill using the skill prior $p_\rho$ and then execute the skill to obtain a rollout in the latent space using the flat dynamics $P_\phi$. Subsequently, we transform this rollout outcome, composed of latent variables $h$ and $z$, into an imaginary trajectory of raw state-action pairs $(s, a)$ using the state decoder $D_\phi$ and the low-level policy (skill decoder) $\pi_\phi$. These newly derived trajectories are added to the dataset for next iterations and further learning.

3.2 Skill-step GC Policy Hierarchy

Following skill-based RL approaches, we structure our GC policy as Eq. (5) in a hierarchy with the (high-level) skill policy $\pi_\psi$ and the (low-level) skill decoder $\pi_\phi$. To accelerate policy learning and adaptation, we also devise a modular GC policy structure that encompasses a skill-step goal generator $f_\psi$ and an inverse skill-step dynamics module $P_\theta^{-1}$. Then, we decompose the skill policy into

$$
\pi_\psi^Z(s_t, g) = P_\theta^{-1}(z|h_t, \hat{h}_{t+H}) \circ f_\psi(\hat{h}_{t+H} | h_s, g) \circ E_\theta(h_t|s_t).
$$

By this decomposition, for a pair of current state $s_t$ and long-horizon goal $g$, the GC policy first infers a skill-step goal $\hat{h}_{t+H}$ and then obtains its corresponding skill $z$ via the inverse skill-step dynamics. Note that the skill decoder $\pi_\phi^Z$ is learned via reconstruction in Eq. (4) and it is responsible for translating $z$ into an action sequence.

In this hierarchy, the inverse skill-step dynamics $P_\theta^{-1}$ is trained as part of the skill-step model in Eq. (8). The adaptation capabilities of the inverse skill-step dynamics rest on such conditions where the environment dynamics remain consistent between the training datasets and downstream tasks; i.e., they share the same underlying world model. In this respect, for downstream tasks with goal distribution shifts in the same environment, we leverage this hierarchy in a way that only the skill-step goal generator needs to be updated. This facilitates parameter-efficient policy updates in the adaptation phase. In this sense, we optimize the skill-step goal generator $f_\psi$ by the skill-step goal loss $L_{sg}$ such as

$$
E_{B \sim \mathcal{D}} \left[ \left( \hat{h}_{t+H} - f_\psi(h_t, g) \right)^2 \right] + \left( f_\psi(h_t, g) - P_\theta(h_t, z) \right)^2
$$

where $\hat{h}_{t+H} = E_\phi(s_{t+H})$ and $z \sim P_\theta^{-1}(z|h_t, f_\psi(h_t, g))$. In Eq. (10), the first term represents behavior cloning error, and the second term serves as a sanity check for cyclic consistency, ensuring that the generated skill-step goal is consistent with the outcome produced by the execution of the skill, which is determined by the skill-step goal.

3.3 Offline Training

In each iteration during the training phase, all the learnable modules (9 modules in Table 1) in GLySA are jointly optimized by the losses based on skills, a skill prior, environment dynamics, and skill-step goals, incorporating Eq. (4)-(10).

$$
L = L_{\text{skill}} + L_{\text{prior}} + L_{\text{model}} + L_{sg}
$$

See Algorithm 1 in Appendix listing the iterative learning procedure for the offline training phase.

3.4 Online Adaptation

After the offline training phase, we can immediately construct the GC policy (structured as Eq. (5) and (9)) using the learned modules and evaluate it in a zero-shot manner. For downstream tasks with different goal distributions, we can also adapt the policy efficiently through few-shot online updates. In this case, we tune only the skill-step goal generator $f_\psi$ via RL, while freezing the other modules. The skill-step goal generator $f_\psi$ is updated through value prediction-based reward maximization, in addition to prior regularization and.

\[4285\]
state consistency regularization, i.e.,
\[
\mathbb{E}_{B'} \left[ -Q(h_t, \pi'_\theta(h_t, g)) + \alpha \cdot KL(\pi'_\theta(z|ht, g)||\pi_\phi(z|h_t)) + (\mathcal{P}_\theta(h_{t+H}|h_t, \pi'_\theta(z|ht, g)) - f_\psi(h_{t+H}|h_t, g))^2 \right]
\]
(12)
where \( h_t = \text{sg}(E_\theta(s_t)) \) and \( B' \) has skill-step transitions \((s_t, z, s_{t+H})\) which are collected online from the environment. The state consistency regularization ensures the skill-step reachability of the skill-step goal \( f_\psi(h_{t+H}|h_t, g) \). This regularization is similar to the second term of Eq. (10). For this policy adaptation via RL, the modules parameterized by \( \theta \) and \( \phi \) except for \( f_\psi \) remain frozen.

4 Experiments
4.1 Experiment Settings
The experiment involves two environments from D4RL [Fu et al., 2020]: maze navigation (maze) and Franka kitchen simulation (kitchen). In the maze, the agent’s objective is to reach a target location from starting location, with states including the agent’s coordinates, velocity, and the goal which is defined by pairs of starting and target locations, and rewards are given only upon reaching this target location. In the Franka kitchen environment, a robot arm manipulates kitchen objects, receiving rewards for achieving the desired state of each target object such as Microwave, Kettle, Bottom burner, Top burner, Light switch, Slide cabinet, and Hinge cabinet. The robot’s state includes its joint and gripper positions, as well as the status of kitchen objects, with goals represented by the states of four target objects. Both environments use offline datasets for training; i.e., 3,046 trajectories in the maze and 603 in the kitchen from [Pertsch et al., 2022] and [Fu et al., 2020], respectively. In the following experiments, the GC policies’ adaptation capabilities are tested under different goal conditions (None, Small, Medium, and Large), which represent the degree of difference between the goal distribution in the offline dataset and that for the adaptation phase.

For evaluation metrics, we adopt a normalized score, ranging from 0 to 100, which is calculated by
\[
\begin{cases} 
I \left[ (\Phi(s) - g)^2 \leq 1 \right] \times 100 & \text{for maze}, \\
(\text{achieved_obj}/\text{target_obj}) \times 100 & \text{for kitchen}
\end{cases}
\]
(13)
where \( \Phi \) maps states \( s \) to a specific goal \( g \), and \( \text{achieved_obj} \) and \( \text{target_obj} \) represent the number of target objects successfully manipulated and the number of target objects in the goal, respectively. The detailed settings for goal specifications and goal distribution shifts are in Appendix A.

For comparison, we use several baselines, encompassing state-of-the-art GC policy learning and skill-based RL methods. Our GC problem settings are characterized by two main constraints: (1) offline GC policy learning, wherein data collection by the policy is prohibited during the training phase, and (2) online GC policy adaptation, which permits either zero-shot evaluation or few-shot updates. The policy adaptation aims at tackling a range of tasks with different degrees of goal distribution shifts. Given the aforementioned constraints, we adapt several baselines, which typically rely on online environment interactions and data collection, to function with offline pre-trained GC policies.

GCRL [Ghosh et al., 2021] is used as a baseline for GCRL experiments to evaluate offline GC policy learning without skill-based strategies. It iteratively collects data through online environment interactions and performs hindsight relabeling and imitation learning to optimize the GC policy. In the experiments, online data collection is replaced with an expert dataset for offline settings, similar to [Yang et al., 2021; Ma et al., 2022]. WGCSL [Yang et al., 2021] is a GCRL variant, employing the advantage-based behavior cloning weights. This method is used to evaluate the state-of-the-art performance of offline GCRL. SPiRL+GCSL is a GCRL variant of the skill-based RL method, SPiRL [Pertsch et al., 2021] to evaluate the GC performance of skill-based RL in offline settings. In our experiments, we train a GC policy using the skills in a supervised manner. We also incorporate the GCRL’s objective into the skill-based RL objective structure. SkiMo+GCSL is a GC variant of the skill-based model-based RL, SkiMo [Shi et al., 2023] which jointly optimizes a model and skills. We adapt SkiMo with GC policy training, and combine the objectives of GCRL and skill-based RL, in the same way of SPiRL+GCSL. This method relies solely on the offline data for model and skill learning, unlike our approach which employs iterative learning through the model-guided skill rollouts. It shows the state-of-the-art GC performance of skill-based RL in offline settings.

4.2 Zero-shot Evaluation Performance
In zero-shot scenarios, we evaluate the GC policy of each method without any policy updates. Table 2 reveals that our GLvSA achieves superior zero-shot performance in both maze and kitchen environments, where the “Dist. shift” column specifies different degrees of goal distribution shifts between the offline dataset and the evaluation tasks. Compared with the most competitive baseline SkiMo+GCSL, GLvSA demonstrates a gain of 14.6, 20.2, and 34.2 in normalized scores for Small, Medium, and Large distribution shifts, respectively, in the maze environment, as well as a gain of 33.8, 39.1, 30.5, respectively, in the kitchen. As the degree of distribution shifts increases, we observe a greater performance drop, particularly for the action-step baselines (GCRL, WGCSL) which do not incorporate skill-based strategies. In contrast, GLvSA displays robustness against these distribution shifts, demonstrating the ability of our GC policy to generalize across a wider range of goals. Importantly, our approach employs the skill-step model to guide trajectory generation, using iterative joint learning of skills, goals, and the model. This renders the skills and goals obtained beyond the scope of the initial offline dataset, marking a distinction from the other skill-based baselines such as SPiRL+GCSL and SkiMo+GCSL, which more rely on the dataset. Figure 4 illustrates the expanding goal coverage in the latent state space of the kitchen environment, achieved by GLvSA over iterations. This growth indicates that a variety of goals are being learned through trajectory generation, forming the basis for strong zero-shot performance.
While the baseline’s 1-shot and 10-shots performance degrades, compared to its zero-shot performance, GLvSA exhibits comparable performance by 1-shot and an improvement by 10-shots, compared to its respective zero-shot; e.g., in the kitchen environment with Large shift, our performance increases from a zero-shot score of 72.2 to 78.4 at 10-shots. As the shot counts increase to 25 and 50, the performance gaps between our GLvSA and the baseline narrow to 32.9 and 15.3 in the maze, and 48.7 and 38.3 in the kitchen, respectively. Nevertheless, these gaps remain substantial. Such results highlight the efficiency of our GC policy hierarchy, which facilitates parameter-efficient updates specifically targeting the skill-step goal generation, thereby enabling few-shot policy adaptation.

### 4.4 Ablations

We conduct ablation studies in the kitchen environment.

**Skill-step model structure.** Table 4 contrasts the zero-shot performance by our model structure employing the skill-step dynamics $\mathcal{P}_\theta$ for model-guided rollouts against a conventional model structure that solely relies on the flat dynamics $\mathcal{P}_\theta$. We also include the “No rollout” case. This comparison sheds light on the advantages of using the skill-step dynamics. Our approach (w/ $\mathcal{P}_\theta$) consistently yields gains over its counterpart (w/o $\mathcal{P}_\theta$). For the cases of larger distribution shifts, we observe a significant performance drop of (w/o $\mathcal{P}_\theta$), which is attributed to the compounding error, a known drawback of model-based RL approaches [Janner et al., 2019; Lu et al., 2020]. The integration of $\mathcal{P}_\theta$ into the model regulates the latent state space using the skills, enabling more dependable rollouts. This, in turn, facilitates enhancements in skill acquisition, model refinement, and policy optimization, even upon substantial goal distribution shifts. We observe that (w/o $\mathcal{P}_\theta$) performs worse than the No rollout case, which is also associated with the compounding error effect. The impact of the model errors can be more pronounced than the benefits of enhanced diversity from trajectory generation.

**Skill horizon.** Table 5 specifies the impact of skill horizon $H$ (defined in Eq. (3)) on the zero-shot performance, where $H$ sets to 1, 5, 10, and 40 in timesteps. We observe that the performance peaks at $H = 10$. As the skill horizon $H$ decreases, the temporal abstraction capability of skills diminishes, leading to a performance drop. This is because the range regularized by the skill-step dynamics $\mathcal{P}_\theta$ in the latent state space contracts, which in turn impacts the state and skill
alignment. On the contrary, the larger horizon $H = 40$ incurs the difficulty for learning the skill-step dynamics $P_a$ and goal generator $f_{Z}$, thus resulting in a performance decline.

<table>
<thead>
<tr>
<th>Dist. shift</th>
<th>$H = 1$</th>
<th>$H = 5$</th>
<th>$H = 10$ (Ours)</th>
<th>$H = 40$</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>44.8 ± 1.4</td>
<td>92.8 ± 1.9</td>
<td>98.7 ± 1.3</td>
<td>90.1 ± 3.0</td>
</tr>
<tr>
<td>Small</td>
<td>41.4 ± 1.5</td>
<td>87.4 ± 1.0</td>
<td>88.6 ± 1.3</td>
<td>76.9 ± 1.1</td>
</tr>
<tr>
<td>Medium</td>
<td>32.1 ± 3.1</td>
<td>78.8 ± 1.1</td>
<td>86.8 ± 1.7</td>
<td>65.8 ± 2.0</td>
</tr>
<tr>
<td>Large</td>
<td>29.5 ± 1.6</td>
<td>66.7 ± 0.5</td>
<td>72.2 ± 3.4</td>
<td>60.2 ± 2.5</td>
</tr>
</tbody>
</table>

Table 5: Effect of skill horizon $H$

**Skill-step goal generation.** To assess the effectiveness of the policy hierarchy incorporating the skill-step goal generation, we compare the adaptation performance of the proposed policy hierarchy in Section 3.2 to its counterpart implemented without the skill-step goal generation (w/o $f_{Z}$). In addition, we introduce another variant, denoted as (w/ 2-skill-steps), which specifically employs a goal state two skill-steps away. This contrasts with our approach that not only uses a goal state a single skill-step away (a skill-step goal) but also harnesses the alignment of skills and goals in terms of temporal abstraction. Table 6 shows that our policy hierarchy outperforms both the counterpart and the variant. Our policy hierarchy effectively mitigates the exploration inefficiency that emerges due to multiple skill combinations to achieve a long-horizon goal, exploiting the skill-step goal generation.

<table>
<thead>
<tr>
<th>Shot</th>
<th>(w/o $f_{Z}$)</th>
<th>(w/ 2-skill-steps)</th>
<th>GLvSA (w/ $f_{Z}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52.3 ± 8.3</td>
<td>57.3 ± 7.6</td>
<td>68.0 ± 5.9</td>
</tr>
<tr>
<td>10</td>
<td>56.0 ± 10.9</td>
<td>62.1 ± 8.9</td>
<td>78.4 ± 4.2</td>
</tr>
<tr>
<td>25</td>
<td>58.9 ± 5.0</td>
<td>60.2 ± 5.9</td>
<td>83.5 ± 8.8</td>
</tr>
<tr>
<td>50</td>
<td>63.9 ± 17.1</td>
<td>70.8 ± 4.6</td>
<td>92.5 ± 2.3</td>
</tr>
</tbody>
</table>

Table 6: Effect of skill-step goal generation

**Sanity check by cyclic consistency.** Table 7 shows the effect of the sanity check in Eq. (10) which ensures the cyclic consistency between the deduced skill-step goals and the outcomes of skill executions. We compare an implementation using only the behavior cloning term without the sanity check (denoted as BC) to our approach (w/ sanity check). We also implement another method (denoted as BC+SR) in which the high-level policy $\pi_{Z}$ is regularized by skill embeddings $z$ through KL-Divergence with the inverse skill-step dynamics, similar to SAC [Haarnoja et al., 2018] and SPIRL [Pertsch et al., 2021]. Our approach outperforms both BC and BC+SR across all degrees of distribution shifts. The sanity check regularizes $\pi_{Z}$ robustly, assisting the skill-step goal generator $f_{Z}$ to predict achievable near-term goals by the skills.

<table>
<thead>
<tr>
<th>Dist. shift</th>
<th>BC</th>
<th>BC+SR</th>
<th>GLvSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>95.6 ± 0.8</td>
<td>85.5 ± 2.4</td>
<td>98.7 ± 1.3</td>
</tr>
<tr>
<td>Small</td>
<td>87.7 ± 0.5</td>
<td>75.4 ± 2.3</td>
<td>88.6 ± 1.3</td>
</tr>
<tr>
<td>Medium</td>
<td>75.7 ± 3.2</td>
<td>58.2 ± 4.6</td>
<td>86.8 ± 1.7</td>
</tr>
<tr>
<td>Large</td>
<td>67.3 ± 4.0</td>
<td>50.0 ± 0.5</td>
<td>72.2 ± 3.4</td>
</tr>
</tbody>
</table>

Table 7: Effect of sanity check

6 Conclusion

In this work, we presented a novel offline GC policy learning framework designed to address long-horizon GC tasks subject to goal distribution shifts. By employing the skill-step model-guided rollouts, our framework extends the range of achievable goals and enhances the adaptation capabilities of the GC policy within offline scenarios. We also proposed a modular policy hierarchy that features the ability to generate near-term goals, attainable by the skills. This hierarchy bolsters both the robustness of the GC policy and its adaptation efficiency against goal distribution shifts. In our future work, we aim to extend the skill-step model for more complex circumstances where distribution shifts in both environment dynamics and goals occur concurrently. Our focus will be on generalizing the skills to better capture the intricacies of environment dynamics patterns during the training phase.
Acknowledgements
We would like to thank anonymous reviewers for their valuable comments and suggestions. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-01045, 2022-0-00043, 2020-0-01821, 2019-0-00421) and by the National Research Foundation of Korea (NRF) grant funded by the MSIT (No. RS-2023-00213118, NRF-2020M3C1C2A01080819) and by Samsung electronics.

Contribution Statement
Donghoon Kim and Minjong Yoo made equal contributions. All the authors participated in designing research, performing research, analyzing data, and writing the paper.

References


