Learn Goal-Conditioned Policy with Intrinsic Motivation for Deep Reinforcement Learning

Jinxin Liu¹²³  Donglin Wang²³*  Qiangxing Tian¹²³  Zhengyu Chen¹²³
¹Zhejiang University, ²Westlake University, ³Westlake Institute for Advanced Study.
{liujinxin, wangdonglin, tianqiangxing, chenzhengyu}@westlake.edu.cn

Abstract
It is of significance for an agent to autonomously explore the environment and learn a widely applicable and general-purpose goal-conditioned policy that can achieve diverse goals including images and text descriptions. Considering such perceptually-specific goals, one natural approach is to reward the agent with a prior non-parametric distance over the embedding spaces of states and goals. However, this may be infeasible in some situations, either because it is unclear how to choose suitable measurement, or because embedding (heterogeneous) goals and states is non-trivial. The key insight of this work is that we introduce a latent-conditioned policy to provide goals and intrinsic rewards for learning the goal-conditioned policy. As opposed to directly scoring current states with regards to goals, we obtain rewards by scoring current states with associated latent variables. We theoretically characterize the connection between our unsupervised objective and the multi-goal setting, and empirically demonstrate the effectiveness of our proposed method which substantially outperforms prior techniques in a variety of tasks.

1 Introduction
Deep reinforcement learning (RL) makes it possible to drive agents to achieve sophisticated goals in complex and uncertain environments, from computer games (Badia et al. 2020) to real robot control (Lee et al. 2018) (Berner et al. 2019) to perceptual tasks (Higgins et al. 2017; Nair et al. 2018; Sermanet et al. 2018; Warde-Farley et al. 2019). However, these approaches taking the prior non-parametric measure function may limit the repertoires of behaviors and impose manual engineering burdens.

On the contrary, latent variable based methods assume that goals (latent variables) and states come from different spaces and the distribution of goals (latent variables) is known a priori. In parallel, such methods autonomously learn a reward function and a latent-conditioned policy through the lens of empowerment (Salge, Giaccio, and Polani 2014; Eysenbach et al. 2018; Sharma et al. 2020). However, such policy is conditioned on latent variables rather than perceptually-specific goals. Applying this procedure to goal-reaching tasks, similar to the parameter initialization or hierarchical RL, needs an external reward function for new tasks; otherwise the learned latent-conditioned policy cannot be applied directly to perceptually-specific goals.

In this paper, we incorporate a latent variable based objective into the perceptual goal-reaching tasks. Specifically, we decouple the task generation (including perceptual goals and associated reward functions) and goal-conditioned policy optimization, which are often intertwined in prior approaches. For the task generation, we employ a latent variable based objective (Eysenbach et al. 2018) to learn a latent-conditioned policy, run to generate goals, and a discriminator, serve as the reward function. Then our goal-conditioned policy is rewarded by the discriminator to imitate the trajectories, relabeled as goals, induced by the latent-conditioned policy. This procedure enables the acquired discriminator as a proxy to reward the goal-conditioned policy for various relabeled goals. In essence, the latent-conditioned policy can reproducibly influence the environment, and the goal-conditioned policy perceptibly imitates these influences.

The main contribution of our work is an unsupervised RL method that can learn a perceptual goal-conditioned policy via intrinsic motivation (GPM). Our training procedure decouples the task (goals and rewards) generation and policy optimization, which makes the obtained reward function universal and effective for various relabeled goals, including images and texts. We formally analyze the effectiveness of our relabeling procedure, and empirically find that our intrinsic reward is well shaped by the environment’s dynamics and as a result benefits the training efficiency on extensive tasks.
2 Preliminaries

The goal in a reinforcement learning problem is to maximize the expected return in a Markov decision process (MDP) \( M \), defined by the tuple \((S, A, \rho, r, \gamma)\), where \( S \) and \( A \) are state and action spaces, \( \rho(s_{t+1}|s_t, a_t) \) gives the next-state distribution upon taking action \( a_t \) in state \( s_t \), \( r(s_t, a_t, s_{t+1}) \) is the reward received at transition \( s_t \xrightarrow{a} s_{t+1} \), and \( \gamma \) is a discount factor. The objective is to learn the policy \( \pi_\theta(a_t|s_t) \) by maximizing \( \mathbb{E}[R(T)|\tau] = \mathbb{E}[\rho(T)|\theta] \left[ \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t, s_{t+1}) \right] \), where \( \tau(\tau; \theta) \) denotes the induced trajectories by policy \( \pi_\theta \) in the environment:

\[
\pi(\tau; \theta) = \pi(s_0) \cdot \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t). 
\]

Multi-goal RL augments the above optimization with a goal \( g \) by learning a goal-conditioned policy \( \pi_\theta(a_t|s_t, g) \) and optimizing \( \mathbb{E}[\rho(g)] \mathbb{E}[\rho(\tau|g) \left[ R(\tau) \right]] \) with reward \( r(s_t, a_t, s_{t+1}, g) \). Such optimization can also be interpreted as a form of mutual information between the goal \( g \) and agent’s trajectory \( \tau \) (Warde-Farley et al. 2019).

\[
\max I(\tau; g) = \mathbb{E}[\rho(g)] \mathbb{E}[\rho(\tau|g)] \left[ \log \mathbb{P}(g) \right] - \log \mathbb{P}(g). 
\]

If \( \log \mathbb{P}(g) \) is unknown and the goal \( g \) is a latent variable, latent variable-based models normally maximize the mutual information between the latent variable \( \omega \) and agent’s behavior \( b \), and lower-bound this mutual information by approximating the posterior \( \mathbb{P}(\omega|b) \) with a learned \( q_\phi(\omega|b) \):

\[
I(\omega; \mu) \geq \mathbb{E}[\rho(\omega)] \left[ \log \mathbb{E}[\rho(\omega)] - \log \mathbb{P}(\omega) \right], 
\]

where the specific manifestation of agent’s behavior \( b \) can be an entire trajectory \( \tau \), an individual state \( s \) or a final state \( s_f \). It is thus applicable to train \( \pi_\theta(a_t|s_t, \omega) \) with learned \( q_\phi \) (as reward).

Several prior works have sought to incorporate the latent-conditioned \( \pi_\theta(a_t|s_t, \omega) \) (as low-level skills) into hierarchical RL (Zhang, Yu, and Xu 2021) or reuse the learned \( q_\phi \) (as predefined tasks) in meta-RL (Gupta et al. 2018), while we claim to reuse both \( \pi_\theta \) and \( q_\phi \) with our relabeling procedure.

3 The Method

In this section, we first formalize the problem and introduce the framework. Second, we illustrate our GIPIM objective and elaborate on the process of how to jointly learn the latent-conditioned policy and a goal-conditioned policy. Third, we formally verify our (unsupervised) objective and understand how GIPIM relates to the standard multi-goal RL.

3.1 Overview

As shown in Figure 1 (right), our objective is to learn a goal-conditioned policy \( \pi_\theta(a_t|s_t, \omega) \) that inputs state \( s_t \) and perceptually-specific goal \( g \) and outputs action \( a_t \). To efficiently generate tasks for training the goal-conditioned policy \( \pi_\theta \), we introduce another latent-conditioned policy \( \pi_\mu(a_t|s_t, \omega) \), which takes as input a state \( s_t \) and a latent variable \( \omega \) and outputs action \( a_t \) to generate goals, and the associated discriminator \( q_\phi \) (i.e., generating tasks). Additionally, we assume that we have access to a procedural relabeling function \( f_\kappa \) (we will discuss this assumption latter), which can relabel states \( s_t \) as goals \( g \) for training \( \pi_\theta \). On this basis, \( \pi_\theta(a_t|s_t, g) \) conditioned on the relabeled goal \( g \) interacts with the reset environment under the instruction of the associated

Figure 1: Framework of GIPIM. We jointly train the latent-conditioned policy \( \pi_\mu \) and the discriminator \( q_\phi \) to understand skills which specify task objectives (e.g., trajectories, the final goal state), and use such understanding to reward the goal-conditioned policy \( \pi_\theta \) for completing such tasks (relabeled states). In this diagram, state \( s_{t+1} \) (e.g., joints) induced by \( \pi_\mu \) is converted into the perceptually-specific goal \( g_t \) (e.g., images or text descriptions) for \( \pi_\theta \). Note that the two environments above are same, and the initial states \( s_0 \) of \( \pi_\mu \) and \( s_0 \) of \( \pi_\theta \) are sampled from the same (fixed) distribution.

\[
q_\phi. \text{ We use the non-tilde } s \text{ and the tilde } \tilde{s} \text{ to distinguish between the states of two policies respectively. Actually, } \tilde{s} \text{ and } s \text{ come from the same state space. In the following, if not specified, goal } g \text{ refers to the perceptually-specific goal, and no longer includes the case that goal is a latent variable.}
\]

To ensure the generated tasks (by \( \pi_\mu \)) are reachable for \( \pi_\theta \), we explicitly make the following assumption:

**Assumption 1** The initial state of the environment is fixed.

3.2 Proposed GIPIM Method

In order to jointly learn the latent-conditioned \( \pi_\mu(a_t|s_t, \omega) \) and goal-conditioned \( \pi_\theta(a_t|\tilde{s}, g) \), we maximize the mutual information between the state \( s_t \) and latent variable \( \omega \) for \( \pi_\mu \), and simultaneously maximize the mutual information between the state \( s_t \) and goal \( g \) for \( \pi_\theta \). Consequently, the overall objective to be maximized can be expressed as follows:

\[
F(\mu, \theta) = I(s_t; \omega) + I(\tilde{s}_t; g). 
\]

For clarification, Figure 2 depicts the graphical model for the latent variable \( \omega \), state \( s_t \) induced by \( \pi_\mu \), goal \( g \) relabeled

\[\text{1}^\text{In appendix, we empirically find the assumption can be lifted.}\]

\[\text{2}^\text{To further clarify the motivation, we conduct the ablation study to compare our method (maximizing } I(s_t; \omega) \text{ plus } I(\tilde{s}_t; g) \text{ with that maximizing } I(s_t; \omega) \text{ and } I(\tilde{s}_t; g) \text{ in appendix.}\]
from $s$, and state $\hat{s}$ induced by $\pi_\theta$. As seen, the latent variable $\omega \sim p(\omega)$ is firstly used to generate state $s$ via the policy $\pi_\mu$, interacting with the environment. Then, we relabel the generated state $s$ to goal $g$. After that, $\pi_\theta$ conditioned on $g$ interacts with the environment to obtain the state $\hat{s}$ at another episode. In particular, $\pi_\theta$ is expected to generate diverse behavior modes by maximizing $\mathcal{I}(s; \omega)$, while $\pi_\theta$ behaving like $\hat{s}$ is “imitate” (see next $q_\phi$) these behaviors by taking as input the relabeled goal (indicated by Figure 3).

Based on the context, the correlation between $\hat{s}$ and $\omega$ is no less than that between $s$ and $\omega$: $\mathcal{I}(\hat{s}; g) \geq \mathcal{I}(s; \omega)$, (Beaudry and Renner 2011). Thus, we can obtain the lower bound:

$$F(\mu, \theta) \geq \mathcal{I}(s; \omega) + \mathcal{I}(\hat{s}; g)$$

where $p_m(\cdot) \triangleq p(\omega, \theta, \hat{s}; s, \mu, \kappa, \theta)$ denotes the joint distribution of $\omega, s, g, \hat{s}$ and $\theta$ specified by the graphic model in Figure 2. Since it is difficult to exactly compute the posterior distributions $p(\omega|s)$ and $p(\omega|\hat{s})$. Jensen’s Inequality (Barber and Agakov 2003) is further applied for approximation by using a learned discriminator network $q_\phi(\omega|\cdot)$. Thus, we have $\mathcal{F}(\mu, \theta) \geq \mathcal{J}(\mu, \phi, \theta)$, where

$$\mathcal{J}(\mu, \phi, \theta) \triangleq 2\mathcal{H}(\omega) + \mathbb{E}_{p_m(\cdot)}[\log p(\omega|s) + \log p(\omega|\hat{s})].$$

It is worth noting that the identical discriminator $q_\phi$ is used for the variational approximation of $p(\omega|s)$ and $p(\omega|\hat{s})$. For the state $s$ induced by skill $\pi_\mu(\cdot|\cdot; \omega)$ and $\hat{s}$ originating from $\pi_\mu(\cdot|\cdot; g)$, the shared discriminator $q_\phi$ assigns a similarly high probability on $\omega$ for both states $s$ and $\hat{s}$ associated with the same $\omega$. Therefore, $q_\phi$ can be regarded as a reward network shared by the latent-conditioned $\pi_\mu$ and goal-conditioned $\pi_\theta$. Intuitively, we factorize acquiring the goal-conditioned policy $\pi_\theta$ and learn it purely in the space of the agent’s embodiment (i.e., the latent $\omega$) — separate from the perceptually-specific goal $g$ (e.g., states, images and texts), where the latent $\omega$ and the perceptual $g$ have different characteristics due to the underlying manifold spaces.

According to the surrogate objective $\mathcal{J}(\mu, \phi, \theta)$, we propose an alternating optimization between $\pi_\mu$, $q_\phi$ and $\pi_\theta$.

**Step I:** Fix $\pi_\mu$ and update $\pi_\theta$ and $q_\phi$. In this case, $\theta$ is not a variable to update and thus $\mathcal{J}(\mu, \phi, \theta)$ becomes

$$\mathcal{J}(\mu, \phi, \theta) = \mathbb{E}_{p(\omega, s|\mu)}[\log q_\phi(\omega|s)] + \mathbb{E}_{p_m(\cdot)}[\log q_\phi(\omega|\hat{s}) - 2 \log p(\omega)].$$

According to Equation 4, $\pi_\mu$ can be thus optimized by setting the intrinsic reward at time step $t$ as

$$r_t = \log q_\phi(\omega|s_{t+1}) - \log p(\omega),$$

where the term $- \log p(\omega)$ is added for agents to avoid artificial termination and reward-hacking issues (Amodei et al. 2016; Eysenbach et al. 2018). We implement this optimization with SAC. In parallel, the reward network (discriminator) $q_\phi$ can be updated with SGD by maximizing

$$\mathbb{E}_{p(\omega)p(s|\omega; \mu)}[\log q_\phi(\omega|s)].$$

Algorithm 1: Learning process of our proposed GPIM

```plaintext
1: while not converged do
2: # Step I: generate goals and reward functions.
3: Sample the latent variable: $\omega \sim p(\omega)$.
4: Reset Env. & sample initial state: $s_0 \sim p_0(s)$.
5: for $t = 0, 1, ..., T - 1$ steps do
6: Sample action: $a_t \sim \pi_\mu(a_t|s_t, \omega)$.
7: Step environment: $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$.
8: Relabel: $g_t = f_\kappa(s_{t+1})$. ▷ Record.
9: Compute reward $r_t$ for policy $\pi_\mu$, using 5.
10: Update policy $\pi_\mu$ to maximize $r_t$ with SAC.
11: Update discriminator ($q_\phi$) to maximize 6 with SGD.
12: end for
13: # Step II: $\pi_\theta$ imitates $\pi_\mu$ with the relabeled goals and the associated rewards (for the same $\omega$).
14: Reset Env. & sample initial state: $s_0 \sim p_0(s)$.
15: for $t = 0, 1, ..., T - 1$ steps do
16: Recap dynamic (time-varying) goal $g_t$ from the recorded goals in line 8. # Note: $g_t = g_t$ for static (fixed) goals.
17: Sample action: $a_t \sim \pi_\mu(a_t|s_t, \omega)$.
18: Step environment: $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$.
19: Compute reward $\hat{r}_t$ for policy $\pi_\theta$ using 5.
20: Update policy $\pi_\theta$ to maximize $\hat{r}_t$ with SAC.
21: end for
22: end while
```

**Step II:** Fix $\pi_\mu$ and $q_\phi$ to update $\pi_\theta$. In this case, $\mu$ and $\phi$ are not variables to update, and $\mathcal{J}(\mu, \phi, \theta)$ can be simplified as

$$\mathcal{J}(\theta) = \mathbb{E}_{p_m(\cdot)}[\log q_\phi(\omega|s)] + \mathbb{E}_{p(\omega, s|\mu)}[\log q_\phi(\omega|s) - 2 \log p(\omega)].$$

According to Equation 7, $\pi_\theta$ can thus be optimized by setting the intrinsic reward at time step $t$ as

$$\tilde{r}_t = \log q_\phi(\omega|\hat{s}_{t+1}) - \log p(\omega),$$

where the term $- \log p(\omega)$ is added for the same reason as above and we also implement this optimization with SAC. Note that we do not update $q_\phi$ with the data induced by $\pi_\theta$.

These two steps are performed alternately until convergence (see Algorithm 1). In summary, we train the goal-conditioned $\pi_\theta$ along with an extra latent-conditioned $\pi_\mu$ as a procedural relabel function $f_\kappa$, which explicitly decouples the procedure of unsupervised RL into task generation (including goals and reward functions) and policy optimization.

For clarity, we state three different settings for $f_\kappa$: (1) if $f_\kappa = q_\phi$, our objective is identical to the latent variable-based models, maximizing $\mathcal{I}(\omega; s)$ to obtain the latent-conditioned policy (Eysenbach et al. 2018); (2) if $f_\kappa(s) = s$, this procedure is consistent with the hindsight relabeling (Andrychowicz et al. 2017); (3) if $f_\kappa(s)$ and $s$ have different spaces (not latent spaces), this relabeling is also a reasonable belief under the semi-supervised setting, e.g., the social partner in Colas et al. (2020). In our experiment, we will consider (2) and (3) to learn $\pi_\theta(\cdot|g)$ that is conditioned on perceptually-specific goals. For (3), it is easy to procedurally generate the image-based goals from (joint-based) states with the MuJoCo Physics Engine’s (Todorov, Erez, and Tassa 2012) built-in renderer. Facing high-dimensional
goals, we also incorporate a self-supervised loss over the perception-level (Hafner et al. 2020; Lu et al. 2020) for \( \pi_\theta \).

### 3.3 Theoretical Analysis

Normally, multi-goal RL seeks the goal-conditioned policy \( \pi_\theta(a|s, g) \) that maximizes \( \mathcal{I}(s; g) \) with prior goal-distribution \( p'(g) \) and the associated reward \( p'(g|s) \). While GPIM learns \( \pi_\theta(a|s, g) \) by maximizing \( \mathcal{I}(s; \omega) + \mathcal{I}(\hat{s}; \omega) \) without any prior goals and rewards. Here, we characterize the theoretical connection of returns between the two objectives under deterministic \( \pi_\mu \) and Assumption 2.

**Assumption 2** The relabeling function \( \eta_\mu \) is bijective and the environment is deterministic.

Let \( \eta_\mu(\pi_\theta) \equiv \mathcal{I}(s; g) = \mathbb{E}_{p'(s, \hat{s}, g)} [\log p'(g|s) - \log p'(g)] \), where the expectation is taken over the rollout \( p'(s, \hat{s}, g; \theta) = p'(g)p'(s|g; \theta) \), and \( \hat{\eta}_\mu(\pi_\theta) \equiv \mathbb{E}_{p_{\pi_\mu}(\cdot)} [\log p(\omega|s) - \log p(\omega)] \), where the joint distribution \( p_{\pi_\mu}(\cdot) \equiv p(\omega, s, g, \hat{s}, \mu; \theta) = p(\omega)p(s|\omega; \mu^*)p(g|s; \mu)p(\hat{s}|g; \theta); \mu^* = \arg \max_\mu \mathbb{I}(s; \omega) \) and \( \mathbb{E}_{p(\omega, s, \mu^*)} [\log p(s|\omega; \mu^*)] = 0 \). According to our training procedure (\( \pi_\theta \) is not affected by \( \pi_\theta \)) in Algorithm 1 it is trivial to show that \( \hat{\eta}_\mu(\pi_\theta) \) is a surrogate for our \( \mathcal{I}(s; \omega) + \mathcal{I}(\hat{s}; \omega) \) in Equation 3. We start by deriving that the standard multi-goal RL objective \( \hat{\eta}_\mu(\pi_\theta) \) and our (unsupervised) objective \( \hat{\eta}_\mu(\pi_\theta) \) are equal under some mild assumptions and then generalize this connection to a general case.

**Special case:** We first assume the prior goal distribution \( p'(g) \) for optimizing \( \eta_\mu(\pi_\theta) \) matches the goal distribution \( \mathbb{E}_{p_\omega}(p(g|\omega; \mu^*, \kappa)) \) induced by \( \pi_\mu \) and \( f_\kappa \) for optimizing \( \hat{\eta}_\mu(\pi_\theta) \). Then, we obtain:

\[
\hat{\eta}_\mu(\pi_\theta) - \eta_\mu(\pi_\theta) = \mathbb{E}_{p_{\pi_\mu}(\cdot)} [\log p(\omega|s) - \log p(\omega) - \log p'(g|s) + \log p'(g)] = \mathbb{E}_{p_{\pi_\mu}(\cdot)} [\log p(\hat{s}|\omega; \mu^*, \kappa, \theta) - \log p(\hat{s}|g; \theta)] = 0. \tag{9}
\]

Equation 9 comes from our relabeling procedure (with deterministic \( \pi_\mu \) and Assumption 3), specifying that \( p'(s|\omega; \mu^*, \kappa, \theta) = \mathbb{E}_{p_\omega}(p(s|\omega; \mu^*)|p(\hat{s}|g; \theta)) \), \( \mathbb{E}_{p_{\pi_\mu}(\cdot)} [\log p(s|\omega; \mu^*)] = 0 \) and \( \mathbb{E}_{p_{\pi_\mu}(\cdot)} [\log p(g|s; \mu^*)] = 0 \). Essentially, this special case shows that without inductive bias on the self-generated goal distribution, our learning procedure leads to the desired goal-conditioned policy \( \pi_\theta \).

**General case:** Suppose that we do not have any prior connection between the goal distributions \( p'(g) \) w.r.t optimizing \( \eta_\mu(\pi_\theta) \) and the self-generated \( \mathbb{E}_{p_\omega}(p(g|\omega; \mu^*, \kappa)) \) w.r.t optimizing \( \hat{\eta}_\mu(\pi_\theta) \). The following theorem provides such a performance guarantee (Please see appendix for a full derivation):

**Theorem 1** Let \( \eta_\mu(\pi_\theta) \) and \( \hat{\eta}_\mu(\pi_\theta) \) be as defined above, and assume relabeling \( f_\kappa \) is bijective, then:

\[
\hat{\eta}_\mu(\pi_\theta) - \eta_\mu(\pi_\theta) \leq 2R_{\max} \sqrt{\epsilon / 2},
\]

where \( R_{\max} = \max_{p_{\pi_\mu}(\cdot)} \mathbb{E}_{p(\omega)} [D_{KL}(p'(g|s)\log p'(g) - \log p'(g))] \) and \( \epsilon = \mathbb{E}_{p(\omega)} [D_{KL}(p(s|\omega; \mu^*)|p'(s))] \).

This theorem implies that as long as we improve the return w.r.t \( \hat{\eta}_\mu(\pi_\theta) \) by more than \( 2R_{\max} \sqrt{\epsilon / 2} \), we can guarantee improvement w.r.t the return \( \eta_\mu(\pi_\theta) \) of standard multi-goal RL.

Note that the analysis presented above makes an implicit requirement that the goal distribution is valid for training \( \pi_\theta \) (i.e., there is the corresponding target in the environment for the agent to pursue). This requirement is well satisfied for the multi-goal RL. However, ambiguity appears when the goal distribution \( \mathbb{E}_{\pi_\theta}(p'(g|\omega; \mu^*, \kappa)) \), induced by \( \eta_\mu(\pi_\theta) \) and \( f_\kappa \), and the existed target \( p'(g) \) in the reset environment for training \( \pi_\theta \) have different supports. For example, the generated goal is “reaching red square”, while such target does not exist in the reset environment for training \( \pi_\theta \) (Algorithm 1 line 14). Thus, we introduce relabeling over the environment (see appendix) for granting valid training.

### 4 Related Work

**Investigating the goal distribution:** For goal-reaching tasks, many prior methods (Schaul et al. 2015; Andrychowicz et al. 2017; Levy et al. 2017; Pong et al. 2018; Hartikainen et al. 2019) assume an available distribution of goals during the exploration. In the unsupervised RL setting, the agent needs to automatically explore the environment and discover potential goals for learning the goal-conditioned policy. Several works (Colas, Sigaud, and Oudeyer 2018; Père et al. 2018; Warde-Farley et al. 2019; Pong et al. 2019; Kovác, Laversanne-Finot, and Oudeyer 2020) also adopt heuristics to acquire the goal distribution based on previously visited states, which is orthogonal to our relabeling procedure.

**Learning the goal-achievement reward function:** Building on prior works in standard RL algorithms (Schaul et al. 2015; Schulman et al. 2017; Haarnoja et al. 2018) that learn policies with prior goals and rewards, unsupervised RL faces another challenge — automatically learning the goal-achievement reward function. Two common approaches to obtain rewards are (1) applying the pre-defined function on the learned goal representations, and (2) directly learning a reward function. Estimating the reward with the pre-defined function typically assumes the goal space is the same as the state space, and learns the embeddings of states and goals with various auxiliary tasks (self-supervised loss in perception-level (Hafner et al. 2020); Sermanet et al. 2018; Warde-Farley et al. 2019; Liu et al. 2021) employ the contrastive loss to acquire embeddings for high-dimensional inputs, and Nair et al. (2018);Florensa et al. (2019); Nair et al. (2019); Pong et al. (2019) elicit the features with the generative models. Over the learned representations, these approaches apply a prior non-parametric measure function (e.g., the cosine similarity) to provide rewards. This contrasts with our decoupled training procedure, where we acquire rewards by scoring current states with their associated latent variables, instead of the perceptual goals. Such procedure provides more flexibility in training the goal-conditioned policy than using pre-defined measurements, especially for the heterogeneous states and goals.

Another approach, directly learning a reward function, aims to pursue skills (the latent-conditioned policy) by maximizing the empowerment (Salge, Glackin, and Polani 2014), which draws a connection between option discovery and information theory. This procedure (Achiam et al. 2018; Eysenbach et al. 2018; Gregor, Rezende, and Wierstra 2017; Campos et al. 2020; Sharma et al. 2020; Tian, Liu, and Wang 2020) typically maximizes the mutual information between a latent variable and the induced behav-
iors (states or trajectories), which is optimized by introducing a latent-conditioned reward function. We explicitly relabels the states induced by the latent-conditioned policy and reuses the learned discriminator for instructing “imitation”.

**Hindsight, self-play and knowledge distillation:** Our method is similar in spirit to goal relabeling methods like hindsight experience replay (HER) (Andrychowicz et al. 2017) which replays each episode with a different goal in addition to the one the agent was trying to achieve. By contrast, our GPIM relabels the task for another policy while keeping behavior invariant. The self-play (Sukhbaatar et al. 2017) and knowledge distillation (Xu et al. 2020) are also related to our relabeling scheme, aiming to refine the training of one task with another associated task.

## 5 Experiments

Extensive experiments are conducted to evaluate our proposed GPIM method, where the following four questions will be considered in the main paper: (1) By using the “archery” task, we clarify whether \( q_{f_\theta} \) can provide an effective reward function on learning the goal-conditioned policy \( \pi_{\theta} \). Furthermore, more complex tasks including navigation, object manipulation, atari games, and mujoco tasks are introduced to answer: (2) Does our model learn effective behaviors conditioned on a variety of goals (with different procedural relabeling \( f_\kappa \), including high-dimensional images and text descriptions that are heterogeneous to states? (3) Does the proposed GPIM on learning the goal-conditioned policy outperform baselines? (4) Does the learned reward function produce better expressiveness of tasks, compared to the prior non-parametric function in the embedding space? For more experimental questions, analysis and results, see Appendix 7 and https://sites.google.com/view/gpim (video).

![Figure 3: "Archery" tasks (left) and the learned rewards (right) on both deterministic and stochastic environments.](image)

**Visualizing the learned reward function.** We start with simple "archery" task to visualize how the learned reward function (discriminator \( q_{f_\theta} \)) accounts for goal-conditioned behaviors in environment. The task shown in Figure 3 requires choosing an angle at which we shoot an arrow to the target. The left upper subfigure shows that in a deterministic environment, given three different but fixed targets (with different colors), the arrow reaches the corresponding target successfully under the learned reward function \( q_{f_\theta} \). The reward as a function of the final location of arrows in three tasks is shown on the right. We can find that the learned reward functions resemble convex in terms of the distance between final states to targets. Specifically, the maximum value of the learned reward function is achieved when the final state is close to the given target. The farther away the agent’s final state is from the target, the smaller this reward value is. Similarly, the same conclusion can be drawn from the stochastic environment in the left lower subfigure, where the angle of the arrow has a 50% chance to become a mirror symmetric angle. We see that the learned reward function substantially describes the environment’s dynamics and the corresponding tasks, both in deterministic and stochastic environments. This answers our first question.

**Scaling to more complex tasks.** To answer our second question, we now consider more complex tasks as shown in Figure 4 (1) In 2D navigation tasks, an agent can move in each of the four cardinal directions. We consider the following two tasks: moving the agent to a specific coordinate named \( x-y \) goal (see appendix for details) and moving the agent to a specific object with certain color and shape named \( \text{color-shape} \) goal. (2) Object manipulation considers a moving agent in 2D environment with one block for manipulation, and the other block as a distractor. The agent first needs to reach the block and then move the block to the target location, where the block is described using color and shape. In other words, the description of the goal contains the \( \text{color-shape} \) goal of the true block and the \( x-y \) goal of the target coordinate. (3) Three \textit{atari} games including seastquest, berzerk and montezuma require an agent to reach the given final states. (4) We use three \textit{mujoco} tasks (swimmer, half cheetah, and fetch) taken from OpenAI GYM (Brockman et al. 2016) to fast imitate given expert trajectories. Specifically, the static goals for \( \pi_{\theta} \) in 2D navigation, object manipulation and atari games are the relabeled final state \( s_T \) induced by the latent-conditioned policy \( \pi_{\mu} \): \( g_t = f_\kappa(s_T) \), and the dynamic goals for \( \pi_{\theta} \) in mujoco tasks are the relabeled states induced by \( \pi_{\mu} \) at each time step: \( g_t = f_\kappa(s_{t+1}) \) for \( 0 \leq t \leq T – 1 \).

The left subfigure of Figure 4(a) shows the learned behavior of navigation in continuous action space given the \( x-y \) goal which is denoted as the small circle, and the right subfigure shows the trajectory of behavior with the given color-shape goal. As observed, the agent manages to learn navigation tasks by using GPIM. Further, 2D navigation with color-shape goal (Figure 4(a) right) and object manipulation tasks (Figure 4(b)) show the effectiveness of our model facing heterogeneous goals and states. Specifically, Figure 4(b) shows the behaviors of the agent on object manipulation, where the agent is asked to first arrive at a block (i.e., blue circle and green square respectively) and then push it to the target location inside a dark circle (i.e., [6.7, 8.0] and [4.8, 7.9] respectively), where the red object exists as a distractor. Figure 4(c) shows the behaviors of agents that reach the final states in a higher dimensional (action, state and goal) space on \textit{seastquest} and \textit{montezuma revenge} respectively. Figure 4(d-e) shows how the agent imitates expert trajectories (dynamic goals) of \textit{swimmer} and \textit{half cheetah}. We refer the reader to appendix for more results (task \textit{berzerk} and \textit{fetch}).

By learning to reach diverse goals generated by the latent-conditioned policy and employing the self-supervised loss over the perception-level to represent goals, the agent learns...
the ability to infer new goals later encountered by the agent. For example, as in Figure 4(a) (right), learning three behaviors with the goal of red-square, red-circle or blue-square (gray background) makes the agent accomplish the new goal of blue-circle (white background). In appendix, we also conduct the ablation study to show how the self-supervised loss (in the perception-level) affects behaviors, and provide more experiments to show the generalization to unseen goals.

Compared to the usual relabeling procedure (e.g., HER with static goals) or latent variable base methods (e.g., DIAYN, equivalent to \( f_x = q_\phi \) in our GPIM), our approach scales to dynamic goals. Considering the setting of dynamic relabeling in fetch task, we further demonstrate the ability of GPIM on temporally-extended tasks, where the 3D-coordinates of the gripper of the robotic arm is relabeled as goals for “imitation” in the training phase. During test, we employ a parameterized complex curve, \((x,y,z) = (t/5, \cos(t)/5 - 1/5, \sin(t)/5)\), for the gripper to follow and show their performance in Figure 5 (left). It is worth noting that during training the agent is required to imitate a large number of simple behaviors and has never seen such complex goals before testing. We also validate GPIM on a Franka Panda robot (Figure 5 (right)), purposing tracking of hand movement, with MediaPipe (Lugaresi et al. 2019) to capturing features of images. It is observed from Figure 5 that the imitation curves are almost overlapping with the given desired trajectories, indicating that the agent using GPIM framework has the potential to learn such compositional structure of goals during training and generalize to new composite goals during test.

**Comparison with baselines.** For the third question, we mainly compare our method to three baselines: RIG (Nair et al. 2018), DISCERN (Warde-Farley et al. 2019), and L2 Distance. L2 Distance measures the distance between states and goals, where the \( L^2 \) distance \(-||s_{t+1} - q_t||^2/\sigma_{\text{pixel}}\) is considered with a hyperparameter \( \sigma_{\text{pixel}} \). Note that \( 2D \) navigation with the color-shape goal and object manipulation using text description makes the dimensions of states and goals different, so \( L^2 \) cannot be used in these two tasks. In RIG, we obtain rewards by using the distances in two embedding spaces and learning two independent VAEs, where one VAE is to encode states and the other is to encode goals. For this heterogeneous setting, we also conduct baseline RIG+ by training one VAE only on goals and then reward agent with the distance between the embeddings of goals and relabeled states (i.e., \( g_k vs. \ f_X(\tilde{s}_{t+1}) \)). We use the normalized distance to goals as the evaluation metric, where we generate 50 goals (tasks) as validation.

We show the results in Figure 6 by plotting the normalized distance to goals as a function of the number of actor’s steps, where each curve considers 95% confidence interval in terms of the mean value across three seeds. As observed, our GPIM consistently outperforms baselines in almost all tasks except for the RIG in 2D navigation (x-y goal) due to the simplicity of this task. Particularly, as the task complexity increases from 2D navigation (x-y goal) to 2D navigation (color-shape goal) and eventually object manipulation (mixed x-y goal and color-shape goal), GPIM converges faster than baselines and the performance gap between our GPIM and baselines becomes larger. Moreover, although RIG learns fast on navigation with x-y goal, it fails to accomplish complex navigation with color-shape goal because the embedding distance between two independent VAEs has difficulty in capturing the correlation of heterogeneous states and goals. Even with a stable VAE, RIG+ can be poorly suited for training the goal-reaching policy. Especially in high-dimensional action space and on more exploratory tasks (atari and mujoco tasks), our method substantially outperforms the baselines.

To gain more intuition for our method, we record the distance \( (\Delta d) \) between the goal induced by \( \pi_\mu \) and the final state induced by \( \pi_\theta \) throughout the training process of the 2D navigation (x-y goal). In this specific experiment, we update \( \pi_\mu \) and \( q_\phi \) but ignore the update of \( \pi_\theta \) before 200 k steps to show the exploration of \( \pi_\mu \) at the task generation phase. As shown in Figure 7, \( \Delta d \) steadily increases during the first 200 k steps, indicating that the latent-conditioned policy \( \pi_\mu \) explores the environment (i.e., goal space) to distinguish skills more easily (with \( q_\phi \)), and as a result, gen-
In this paper, we propose GPIM to learn a goal-conditioned policy in an unsupervised manner. The core idea of GPIM lies in that we introduce a latent-conditioned policy with a procedural relabeling procedure to generate tasks (goals and the associated reward functions) for training the goal-conditioned policy. For goal-reaching tasks, we theoretically describe the performance guarantee of our (unsupervised) objective compared with the standard multi-goal RL. We also conduct extensive experiments on a variety of tasks to demonstrate the effectiveness and efficiency of our method.

There are several potential directions for future in our unsupervised relabeling framework. One promising direction would be developing a domain adaptation mechanism when the interaction environments (action/state spaces, dynamics, or initial states) wrt learning $\pi_{\mu}$ and $\pi_{\theta}$ are different. Additionally, GPIM can get benefits from more extensive exploration strategies to control the exploration-exploitation trade-off. Finally, latent-conditioned $\pi_{\mu}$ (generating goals and reward functions) is not affected by the goal-conditioned $\pi_{\theta}$ in GPIM. One can develop self-paced (curriculum) learning over the two policies under the unsupervised RL setting.
Acknowledgments

The authors would like to thank Yachen Kang for helpful discussions and the anonymous reviewers for the valuable comments. This work is supported by NSFC General Program (62176215).

References

Achiam, J.; Edwards, H.; Amodei, D.; and Abbeel, P. 2018. Variational Option Discovery Algorithms. CoRR, abs/1807.10299.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P. F.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. CoRR, abs/1606.06565.

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, O. P.; and Zaremba, W. 2017. Hindsight experience replay. In Advances in neural information processing systems, 5048–5058.

Badia, A. P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, D.; and Blundell, C. 2020. Agent57: Outperforming the atari human benchmark. arXiv preprint arXiv:2003.13350.

Barber, D.; and Agakov, F. V. 2003. The IM algorithm: a variational approach to information maximization. In Advances in neural information processing systems, None.

Beaudry, N. J.; and Renner, R. 2011. An intuitive proof of the data processing inequality. arXiv preprint arXiv:1107.0740.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym.

Campos, V. A.; Trott, A.; Xiong, C.; Socher, R.; i Nieto, X. G.; and Torres, J. 2020. Explore, Discover and Learn: Unsupervised Discovery of State-Covering Skills. ArXiv, abs/2002.03647.

Colas, C.; Karch, T.; Lair, N.; Dussoux, J.; Moulin-Frier, C.; Dominey, P. F.; and Oudeyer, P. 2020. Language as a Cognitive Tool to Imagine Goals in Curiosity Driven Exploration. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Linard N., eds., Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Colas, C.; Sigaud, O.; and Oudeyer, P.-Y. 2018. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. arXiv preprint arXiv:1802.05054.

Eysenbach, B.; Gupta, A.; Ibarz, J.; and Levine, S. 2018. Diversity is all you need: Learning skills without a reward function. arXiv preprint arXiv:1802.06070.

Florens, C.; Degrave, J.; Heess, N.; Springenberg, J. T.; and Riedmiller, M. 2019. Self-supervised learning of image embedding for continuous control. arXiv preprint arXiv:1901.00943.

Gregor, K.; Rezende, D. J.; and Wierstra, D. 2017. Variational Intrinsic Control.

Gupta, A.; Eysenbach, B.; Finn, C.; and Levine, S. 2018. Unsupervised Meta-Learning for Reinforcement Learning. CoRR, abs/1806.04640.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290.

Hafner, D.; Ortega, P. A.; Ba, J.; Parr, T.; Friston, K. J.; and Heess, N. 2020. Action and Perception as Divergence Minimization. CoRR, abs/2009.01791.

Hartikainen, K.; Geng, X.; Haarnoja, T.; and Levine, S. 2019. Dynamical Distance Learning for Semi-Supervised and Unsupervised Skill Discovery. In International Conference on Learning Representations.

Higgins, I.; Pal, A.; Rusu, A.; Matthey, L.; Burgess, C.; Pritzel, A.; Botvinick, M.; Blundell, C.; and Lerchner, A. 2017. Darla: Improving zero-shot transfer in reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, 1480–1490. JMLR. org.

Jabri, A.; Hsu, K.; Gupta, A.; Eysenbach, B.; Levine, S.; and Finn, C. 2019. Unsupervised curricula for visual meta-reinforcement learning. In Advances in Neural Information Processing Systems, 10519–10531.

Khirodkar, R.; Yoo, D.; and Kitani, K. M. 2018. VADRA: Visual adversarial domain randomization and augmentation. arXiv preprint arXiv:1812.00491.

Kovač, G.; Lavasanne-Finot, A.; and Oudeyer, P.-Y. 2020. Grimgep: learning progress for robust goal sampling in visual deep reinforcement learning. arXiv preprint arXiv:2008.04388.

Laskin, M.; Srivivas, A.; and Abbeel, P. 2020. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. 119: 5639–5650.

Lee, L.; Eysenbach, B.; Parisotto, E.; Xing, E. P.; Levine, S.; and Salakhutdinov, R. 2019. Efficient Exploration via State Marginal Matching. CoRR, abs/1906.05274.

Lee, Y.; Sun, S.-H.; Somasundaram, S.; Hu, E. S.; and Lim, J. J. 2018. Composing complex skills by learning transition policies. In International Conference on Learning Representations.

Levy, A.; Konidaris, G.; Platt, R.; and Saenko, K. 2017. Learning multi-level hierarchies with hindsight. arXiv preprint arXiv:1712.00948.

Liu, G.; Zhang, C.; Zhao, L.; Qin, T.; Zhu, J.; Li, J.; Yu, N.; and Liu, T. 2021. Return-Based Contrastive Representation Learning for Reinforcement Learning. CoRR, abs/2102.10960.

Lowrey, K.; Rajeswaran, A.; Kakade, S. M.; Todorov, E.; and Mordatch, I. 2019. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control.

Lu, X.; Lee, K.; Abbeel, P.; and Tiomkin, S. 2020. Dynamics Generalization via Information Bottleneck in Deep Reinforcement Learning. CoRR, abs/2008.00614.
Lugaresi, C.; Tang, J.; Nash, H.; McClanahan, C.; Uböweja, E.; Hays, M.; Zhang, F.; Chang, C.; Yong, M. G.; Lee, J.; Chang, W.; Hua, W.; Georg, M.; and Grundmann, M. 2019. MediaPipe: A Framework for Building Perception Pipelines. CoRR, abs/1906.08172.

Nair, A.; Bahl, S.; Khazatsky, A.; Pong, V.; Berseth, G.; and Levine, S. 2019. Contextual Imagined Goals for Self-Supervised Robotic Learning. 100: 530–539.

Nair, A. V.; Pong, V.; Dalal, M.; Bahl, S.; Lin, S.; and Levine, S. 2018. Visual reinforcement learning with imagined goals. In Advances in Neural Information Processing Systems, 9191–9200.

Pérez, A.; Forestier, S.; Sigaud, O.; and Oudeyer, P.-Y. 2018. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. arXiv preprint arXiv:1803.00781.

Pitis, S.; Chan, H.; Zhao, S.; Stadie, B. C.; and Ba, J. 2020. Maximum Entropy Gain Exploration for Long Horizon Multi-goal Reinforcement Learning. ArXiv, abs/2007.02832.

Pong, V.; Gu, S.; Dalal, M.; and Levine, S. 2018. Temporal difference models: Model-free deep rl for model-based control. arXiv preprint arXiv:1802.09081.

Pong, V. H.; Dalal, M.; Lin, S.; Nair, A.; Bahl, S.; and Levine, S. 2019. Skew-fit: State-covering self-supervised reinforcement learning. arXiv preprint arXiv:1903.03698.

Popov, I.; Heess, N.; Lillicrap, T.; Hafner, R.; Barth-Maron, G.; Vecerik, M.; Lampe, T.; Tassa, Y.; Erez, T.; and Riedmiller, M. 2017. Data-efficient deep reinforcement learning for dexterous manipulation. arXiv preprint arXiv:1704.03073.

Salge, C.; Glackin, C.; and Polani, D. 2014. Empowerment: an introduction. In Guided Self-Organization: Inception, 67–114. Springer.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal value function approximators. In International conference on machine learning, 1312–1320.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. CoRR, abs/1707.06347.

Sermanet, P.; Lynch, C.; Chebotar, Y.; Hsu, J.; Jang, E.; Schaal, S.; Levine, S.; and Brain, G. 2018. Time-contrastive networks: Self-supervised learning from video. In 2018 IEEE International Conference on Robotics and Automation (ICRA), 1134–1141. IEEE.

Sharma, A.; Gu, S.; Levine, S.; Kumar, V.; and Hausman, K. 2020. Dynamics-Aware Unsupervised Discovery of Skills. Sukhbaatar, S.; Denton, E.; Szlam, A.; and Fergus, R. 2018. Learning goal embeddings via self-play for hierarchical reinforcement learning. arXiv preprint arXiv:1811.09083.

Tian, Q.; Liu, J.; and Wang, D. 2020. Learning transitional skills with intrinsic motivation.
Appendix

Here, we will provide additional experiments (Section 7), theoretical derivations (Section 8), implementation details (Section 9), and more results with respect the experiments in the main text (Section 10).

7 Additional Experiments

In this section, we make the additional experiments to

1. compare our method with DIAYN and its variant,
2. show the generalization when the dynamics and goal conditions are missing,
3. conduct the ablation study to analyze how the self-supervised loss over the perception-level affects the learned behaviors in terms of the generalization to new tasks,
4. explore the (automatic) goal distribution generation,
5. analyze the effect caused by the underspecified latent (support) space,
6. lift the Assumption 1 (assuming fixed initial state) in the main text,
7. evaluate the robustness of GPIM on stochastic MDP (with varying stochasticity).

7.1 Comparison with DIAYN and its Variant

In this section, we expect to clarify the difference and connection with DIAYN (Eysenbach et al. 2018) experimentally, and indicate the limitations of maximizing $I(s; \omega)$ and maximizing $I(s; g)$ separately in learning the goal-conditioned policy.

![Figure 9: 2D navigation with (a) DIAYN-Imitator, (b) DIAYN-Goal, and (c) our proposed GPIM. In (a), (b) and (c), the open (colored) circle denotes the user-specified goal in test, and the (colored) line with a solid circle at the end is the corresponding behavior. Note that the white lines in (a) are the learned skills in the training process, where we show 10 diverse skills. We can see that neither DIAYN-Imitator nor DIAYN-Goal approaches the corresponding goals, while our model succeeds in reaching the goal as shown in (c). In (d), we also show the generative factors $z$ of goals in GPIM.](image)

In DIAYN, authors show the ability of the model to imitate an expert. Given the goal, DIAYN uses the learned discriminator to estimate which skill was most likely to have generated the goal $g$:

$$\hat{\omega} = \arg \max_{\omega} q_\phi(\omega | g). \quad (10)$$

Here we call this model DIAYN-Imitator. We also directly substitute the perceptually-specific goals for the latent variable in DIAYN’s objective to learn a goal-conditioned policy. We call this model DIAYN-Goal:

$$\max_{s} I(s; g), \quad (11)$$

where $g$ is sampled from the prior goal distribution $p(g)$. Please note that we do not adopt the prior non-parametric distance as in DISCERN (Warde-Farley et al. 2019) to calculate the reward. We obtain the reward as in vanilla DIAYN using the variational approximator $q_\phi(g | s)$.

Figure 9 shows the comparison of our GPIM with DIAYN variants, including DIAYN-Imitator and DIAYN-Goal, where the 2D navigation task is considered. As observed, DIAYN-Imitator can reach seen goals in training (white lines) but not unseen goals in test in Figure 9(a), because it cannot effectively accomplish the interpolation between skills that are induced in training. And behaviors generated by DIAYN-Goal cannot guarantee consistency with the preset goals in Figure 9(b). The main reason is that such objective only ensures that when $g$ (or $\omega$) is different, the states generated by $g$ (or $\omega$) are different. However, there is no guarantee that $g$ (or $\omega$) and the state generated by the current $g$ (or $\omega$) keep semantically consistent. Our proposed GPIM method, capable of solving interpolation and consistency issues, exhibits the best performance in this 2D navigation task.

Moreover, when the user-specified goals are heterogeneous to the states, the learned discriminator $q_\phi$ in DIAYN is unable to estimate which skill is capable of inducing given goals. Specifically, when goals are visual inputs, and states in training is
feature vectors (e.g., joint angles), the learned discriminator is unable to choose the skills due to a lack of models for converting high-dimensional images into low-dimensional feature vectors. On the contrary, there are lots of off-the-shelf models to render low-dimensional feature vectors into perceptually-specific high-dimensional inputs (Tobin et al. 2017; Khirodkar, Yoo, and Kittair 2018).

### 7.2 Generalization on the Gridworld Task

Here we introduce an illustrative example of gridworld task and then show the generalization when the dynamics and goal conditions are missing in the gridworld task.

**Illustrative example.** Here, we start with a simple gridworld example: the environment is shown on the left of Figure 10, where the goal for the agent is to navigate from the middle to the given colored room. By training the latent-conditioned policy and the discriminator, our method quickly acquire four skills (ω = 1000; 0100; 0010; 0001) to reach different rooms. Each time the agent arrives in a room induced by πω, we train πθ conditioned on the room’s color (e.g., green), allowing the agent to be guided to the same room by the current reward function (e.g., qφ conditioned on ω = 1000). The results are shown on the right of Figure 10, where the upper and lower subfigures show the learning processes of latent-conditioned policy πθ and goal-conditioned policy πω respectively. It is concluded that the agent can automatically learn how to complete tasks given semantic goals in an unsupervised manner.

**Generalization on the gridworld task.** We further show the generalization when the dynamics and goal conditions are missing in the gridworld task, where there are four rooms in different colors: blue, green, orange and magenta. We consider the situation where we train the goal-conditioned policy in the blue room, the green room and the orange room, and test in the magenta room.

As shown in Figure 11, we quickly acquire four skills for different rooms through the training at the abstract level. After the agent reaches at the colored rooms, we relabel the corresponding room as a two-bit representation: fω('blue') = 10, fω('green') = 00, and fω('orange') = 01; we do not relabel the magenta room. We take the magenta room corresponding to 11 as the test goal to verify the generalization ability.

In lower part of Figure 11 we show the learning process of the goal-conditioned policy. We can find that the blue room task is completed quickly, and the green and orange room tasks are learned relatively slowly, and the agent is still able to complete the (new) test task successfully (the magenta room). Compared to the task in Figure 10 that relabel all the rooms as goals, the whole learning process in Figure 11 is much slower. We hypothesize that the main reason is that the agent needs to further infer the relationship between different goals. The lack of goal information (i.e., missing magenta 11) leads to a lower efficiency and unstable training.

### 7.3 Self-Supervised Loss over the Perception-Level

As indicated by Laskin, Srinivas, and Abbeel (2020), control algorithms built on top of useful semantic representations should be significantly more efficient. We further optimize the goal-conditioned policy πω with an associated self-supervised loss on the perception-level, so as to improve the learning efficiency and generalization ability of the goal-conditioned policy πω facing high-dimensional goals. As shown in Figure 12, we decompose πω into two components: the encoder network E parameterized by φE and the generative network G parameterized by φG. Hence, we optimize the overall objective function for the goal-conditioned policy πω with the control-level loss (SAC with reward \( \tilde{r}_t \)) and perception-level loss.

---

**Figure 10:** Gridworld tasks. Y-axis is the percentage of different rooms that the robot arrives in.

**Figure 11:** Gridworld tasks. The Y-axis is the percentage of different rooms that agent arrives in.

**Figure 12:** Optimization for the goal-conditioned policy πω with the control-level loss (SAC with reward \( \tilde{r}_t \)) and perception-level loss.
conditioned policy \( \pi_\theta \):

\[
\max_{\vartheta, E, \theta} \mathbb{E}_{p_m(\cdot)} \left[ \log q_\phi(\omega|s) \right] + \text{PrecLoss},
\]

(12)

where the self-supervised loss on the perception-level \( \text{PrecLoss} \) includes the reconstruction loss and a prior matching loss (with an isotropic unit Gaussian prior \( p(z) = \mathcal{N}(0; I) \)):

\[
\text{PrecLoss} \triangleq \alpha \cdot \mathbb{E}_{p(g)p_{\vartheta E}(z|g)} \left[ p_{\vartheta D}(g|z) \right] - \beta \cdot \mathbb{E}_{p(g)} \left[ KL(q_{\theta E}(z|g)||p(z)) \right],
\]

where \( p(g) = p(\omega)p(s|\omega; \mu)p(g|s; \kappa) \), \( \alpha \) and \( \beta \) are two hyperparameters. It is worth noting that the update of the encoder \( E \) is based on gradients from both \( \pi_\theta \) and \( \text{PrecLoss} \).

### 7.4 Automated Goal-Generation for Exploration

In general, for unsupervised RL, we would like to ask the agent to carry out autonomous "practice" during training phase, where we do not know which particular goals will be provided in test phase. In order to maximize the state coverage, the ability to automatically explore the environment and discover diverse goals is crucial. In this section, we will further analyze the goal distribution of three methods (RIG (Nair et al. 2018), DISCERN (Warde-Farley et al. 2019), GPIM) in a new 2D obstacle navigation task as shown in Figure 14(a), Figure 14(b) and Figure 14(c). The size of the environment is \( 10 \times 10 \), the initial state is set as \([5, 2]\), and there are two obstacles that prevent the agent from passing through each of which is \( 3 \times 6 \) in size.

DISCERN samples goals during training by maintaining a fixed sized buffer \( \mathcal{G} \) of past observations. We simply mimic the process of goal generation by taking random actions for 20 environment steps. As in Figure 14(b), we generate 100 goals (left) in a counterclockwise order, GPIM can successfully disentangle them and learn effective latent \( z \) (right), but w/o \( KL \) fails to keep the original order of goals (middle).

### Ablation study on mujoco tasks

We further study the impact of perception-level loss on mujoco tasks. The performance is reported in Figure 13(b). It is observed that w/o \( Pol \) performs worse than all other methods, which is consistent with the performance of RIG that trains VAE and policy separately. The main reason is that the process of learning generative factors become more difficult without the supervised reconstruction loss. While in 2D navigation task, the reconstruction loss has little impact on the performance. Even though that w/o \( KL \) has a similar performance with our full GPIM method, GPIM demonstrates better interpretability to behaviors as shown in Figure 9(d).
goal space. The decoded latent goals are shown in Figure 14(a), where we sample 30 goals. It is shown that the majority of goals are also between the two obstacles because the goals for training the VAE prior come from the same distribution as in DISCERN.

Our method, GPIM, obtains goals from the behaviors induced by the latent-conditioned policy. Maximizing \( I(s; \omega) \) encourages different skills to induce different states that are further relabeled to goals. This objective ensures that each skill individually is distinct and the skills collectively explore large parts of the state space (Eysenbach et al. 2018). As shown in Figure 14(c), our method provides better coverage of the state space than DISCERN and RIG.

Figure 14(d) shows the performance of the three methods, where we randomly sample goals from the whole state (or goal) space at test phase. We can see that our method significantly outperforms the baselines. The most common failure mode for prior methods is that the goal distribution collapses (Pong et al. 2019), causing that the agent can reach only a fraction of the state space, as shown in Figure 14(a) and 14(b).

Exploration is a well-studied problem in the field of RL, and there are many proven approaches with different benefits to improve the exploration (Campos et al. 2020; Colas, Sigaud, and Oudeyer 2018; Pitis et al. 2020). Note that these benefits are orthogonal to those provided by our straightforward GPIM, and these approaches could be combined with GPIM for even greater effect. We leave combing our method with sophisticated exploration strategies to future work.

### 7.5 Training with Underspecified Latent (Support) Space

Here we show that underspecified latent (support) space may induce a large performance gap for learning the goal-conditioned policy. Qualitatively, Theorem 1 explains the performance gap induced by \( p(s|\omega, \mu) \) and highlights that such gap is sensitive to the prior latent distribution \( p(\omega) \). We thus add additional experiments on the distribution of skills on 2D Navigation, Object manipulation and Fetch tasks, as shown in Table 1.

| Task       | Discrete; 5 skills | Discrete; 10 skills | Discrete; 50 skills | Discrete; 100 skills | Continuous |
|------------|--------------------|---------------------|---------------------|----------------------|------------|
| 2D Nav.    | 0.296 (+/- 0.091)  | 0.152 (+/- 0.066)   | 0.137 (+/- 0.052)   | 0.132 (+/- 0.037)   | 0.122 (+/- 0.044) |
| O-M        | 0.535 (+/- 0.110)  | 0.321 (+/- 0.082)   | 0.199 (+/- 0.057)   | 0.193 (+/- 0.064)   | 0.182 (+/- 0.060) |
| Fetch      | 0.376 (+/- 0.175)  | 0.320 (+/- 0.104)   | 0.257 (+/- 0.089)   | 0.162 (+/- 0.057)   | 0.149 (+/- 0.033) |
We found that the number of skills does affect the performance of the goal-conditioned policy. In simple tasks (2D navigation task), the effect is relatively small. But in complex environments, this effect cannot be ignored. We analyze the generated goal distributions in these experiments with different numbers of skills and found that goals generated by a small number of skills are extremely sparse, leading to the bias of the goal-conditioned policy. Nevertheless, such bias can be reduced, to some extend. As shown in 2D navigation task, learning with 10 skills can still produce comparable results. We attribute this result to the policy’s generalization ability and the perception-level loss (Appendix 7.3).

7.6 Training with Unfixed Initial State
In Assumption (the main text), we assume that the initial state of the environment is fixed. For granting valid training for each rollout of the goal-conditioned $\pi_\theta$, we need to keep the initial state fixed or filter out initial states (for goal-conditioned $\pi_\theta$, see Line 14 in Algorithm 1) that never lead to the relabeled goals, eg. constraining $\text{Distance}(s_0, s_0) \leq \epsilon$.

However, although randomly sampling initial states for $\pi_\mu$ and $\pi_\theta$ from the initial distribution (more than one fixed state) can not guarantee that the coming training (for $\pi_\theta$) must be valid (goals are reachable), a large number of samples will ensure that there exist cases where initial states obtained by two consecutive resets is close, which can enable valid training. We also validate such setting (more than one fixed initial state) in 2D navigation (x-y goal) and 2D navigation (color-shape goal) tasks, as shown in Figure 15. We can observe that 1) such performance is reduced compared to the fixed starting point setting, and 2) the performance of our method outperforms that of DISCERN with the same initial state distribution (more than one fixed state).

Figure 15: GPIM: training GPIM with the fixed initial state [0,0]. GPIM*: training GPIM with the initial state distribution (uniformly sampled from [0,0], [-0.25, -0.25], [-0.25, 0.25], [0.25, -0.25], [0.25, 0.25]). DISCERN*: training DISCERN with the initial state distribution (uniformly sampled from [0,0], [-0.25, -0.25], [-0.25, 0.25], [0.25, -0.25], [0.25, 0.25]).

7.7 The Robustness of GPIM on Stochastic MDP
Here we show the robustness of our GPIM on stochastic MDP. We add random noise to the dynamics of the environment 2D navigation, so as to build the stochastic MDP: \( p(s' | s, \pi(s)) \rightarrow p(s' | s, \pi(s) + \text{action-space.sample()} \cdot \Delta) \), where we sample an random action from the action space and then weigh the action and the action output from the policy with $\Delta$. Thus, we can use $\Delta$ to represent the stochasticity of the environment.

We shown our results in the stochastic environments in Figure 16. We can find that when $\Delta$ equals 0.1, the learning process is hardly affected. As the stochasticity of the environment increases, the performance of the method decreases. When $\Delta$ equals 0.5, the method barely completes the task.

Figure 16: Evaluation on the stochastic MDP.

8 Derivation

8.1 Derivation of the Variational Bound
\[
\mathbb{E}_{p_m(\cdot)} \left[ \log p(\omega | s) + \log p(\omega | \tilde{s}) - 2 \log p(\omega) \right] \\
= \mathbb{E}_{p_m(\cdot)} \left[ \log q_\phi(\omega | s) + \log q_\phi(\omega | \tilde{s}) - 2 \log p(\omega) \right] \\
+ \mathbb{E}_{p_m(\cdot)} \left[ KL(p(\omega | s) || q_\phi(\omega | s)) + KL(p(\omega | \tilde{s}) || q_\phi(\omega | \tilde{s})) \right] \\
\geq \mathbb{E}_{p_m(\cdot)} \left[ \log q_\phi(\omega | s) + \log q_\phi(\omega | \tilde{s}) - 2 \log p(\omega) \right]
\]
8.2 Proof of Theoretical Guarantees

Note that
\[ \eta(\pi_\theta) = \mathbb{E}_{p'(g,s;\theta)}[\log p'(g|s) - \log p'(g)], \]
and
\[ \hat{\eta}(\pi_\theta) = \mathbb{E}_{p(\omega,s,g,\tilde{s};\mu^*,\kappa,\theta)}[\log p(\omega|\tilde{s}) - \log p(\omega)], \]
where \( \mu^* = \arg\max_{\mu} \mathcal{I}(s; \omega) \). For deterministic \( \pi_\mu \), we have \( \mathbb{E}_{p(\omega,s;\mu^*)} \log p(s|\mu^*) = 0 \).

Thus, we have:
\[
\begin{align*}
\hat{\eta}(\pi_\theta) - \eta(\pi_\theta) & \equiv \frac{\partial}{\partial \mu^*} \log p(\omega,s,\tilde{s};\mu^*,\kappa,\theta) \left[ \log p'(g|\tilde{s}) - \log p'(g) \right] \\
\mathbb{E}_{p_m(\cdot)}[\log p'(g|\tilde{s}) - \log p'(g)] & \leq \mathbb{E}_{p_m(\cdot)}[\log p'(g|\tilde{s}) - \log p'(g) + \log p'(g)] \\
\mathbb{E}_{p_m(\cdot)}[\log p'(g|\tilde{s}) - \log p'(g)] & \leq \mathbb{E}_{p_m(\cdot)}[\log p'(g|\tilde{s}) - \log p'(g)] \\
\mathbb{E}_{p_m(\cdot)}[\log p'(g|\tilde{s}) - \log p'(g)] & \leq \mathbb{E}_{p_m(\cdot)}[\log p'(g|\tilde{s}) - \log p'(g)].
\end{align*}
\]

**Special case:** If we assume the prior goal distribution \( p' (g) \) optimizing \( \eta(\pi_\theta) \) matches the goal distribution \( p(g|\omega; \mu^*, \kappa) \) induced by \( \pi_\mu \) and \( f_\kappa \) for optimizing \( \eta(\pi_\theta) \), e.g., \( \mathbb{E}_{p}[p(g|\omega; \mu^*, \kappa)] = p'(g) \), thus we have \( p_m^*(\cdot) \equiv p(\omega,s;\mu^*)p(g,\tilde{s}|s;\kappa,\theta) = p(\omega,s;\mu^*)p'(g,\tilde{s};\theta) \). By inserting this equation into \( \hat{\eta}(\pi_\theta) - \eta(\pi_\theta) \), we obtain
\[
\text{TERM-II} = 0.
\]

**General case:** If \( \mathbb{E}_{p}[p(g|\omega; \mu^*, \kappa)] \neq p'(g) \), e.g., \( p_m^*(\cdot) \equiv p(\omega,s;\mu^*)p(g,\tilde{s}|s;\kappa,\theta) \neq p(\omega,s;\mu^*)p'(g,\tilde{s};\theta) \), we apply Holder’s inequality and Pinsker’s inequality over TERM-II. Then the following inequality holds:
\[
\begin{align*}
\text{TERM-II} & \leq \mathbb{E}_{p(\omega,s;\mu^*)}[\mathbb{E}_{p(g,\tilde{s};\kappa,\theta)}[R(g,\tilde{s})] - \mathbb{E}_{p'(g,\tilde{s};\theta)}[R(g,\tilde{s})]] \\
& \leq \mathbb{E}_{p(\omega,s;\mu^*)}[\mathbb{E}[R(g,\tilde{s})]_{\infty} \cdot \mathbb{E}[p(g,\tilde{s};s;\kappa,\theta) - p'(g,\tilde{s};\theta)]_1] \\
& \leq \mathbb{E}_{p(\omega,s;\mu^*)}\left[ \max_{g,\tilde{s}} R(g,\tilde{s}) \cdot \frac{1}{2} \mathbb{D}_{KL}(p(g,\tilde{s};s;\kappa,\theta) \| p'(g,\tilde{s};\theta)) \right] \\
& = \mathbb{E}_{p(\omega,s;\mu^*)}\left[ \max_{g,\tilde{s}} R(g,\tilde{s}) \cdot \frac{1}{2} \mathbb{D}_{KL}(p(g|s;\kappa) \| p'(g)) \right],
\end{align*}
\]
where \( R(g,\tilde{s}) = \log p'(g|\tilde{s}) - \log p'(g) \).

Assuming the goal space is the same as the state space and relabeling \( f_\kappa \) is bijective, we can ignore the relabeling term \( p(g|s;\kappa) \) and replace \( p'(g) \) with \( p'(s) \). Thus, we have
\[
\text{TERM-II} \leq \left( \max_{g,\tilde{s}} R(g,\tilde{s}) \right) \cdot \mathbb{E}_{p(\omega)} \left[ \frac{1}{2} \mathbb{D}_{KL}(p(s;\mu^*) \| p'(s)) \right].
\]
9 Implementation Details

9.1 Relabeling over the Environment

Applying our GPIM on 2D navigation (color-shape goal) may cause that the color and shape of the goal that relabeled from a state that does not represent a valid goal in the reset environment (since valid goals are only ones that can be reached by the agent reach after this reset). In Algorithm 1 (in the main text), the goals are claimed to be generated by \( \pi_{\theta} \) while there may be no corresponding target in the reset environment for training \( \pi_{\theta} \). Thus, at the step that we reset the environment for training \( \pi_{\theta} \) (line 14), we make this reset is associated with the generated goal by \( \pi_{\mu} \) and \( f_{\kappa} \). See Algorithm 2 for the details.

Algorithm 2: Learning process of our proposed GPIM (with relabeling over the environment)

```plaintext
1: while not converged do
2:     # Step I: generate goals and reward functions.
3:         Sample the latent variable: \( \omega \sim p(\omega) \).
4:         Reset Env. & sample initial state: \( s_0 \sim p_0(s) \).
5:         for \( t = 0, 1, \ldots, T - 1 \) steps do
6:             Sample action: \( a_t \sim \pi_{\mu}(a_t|s_t, \omega) \).
7:             Step environment: \( s_{t+1} \sim p(s_{t+1}|s_t, a_t) \).
8:             Relabel: \( g_t = f_\kappa(s_{t+1}) \). ▷ Record.
9:             Compute reward \( r(t) \) for policy \( \pi_{\mu} \) using (5).
10:            Update policy \( \pi_{\mu} \) to maximize \( r_t \) with SAC.
11:            Update discriminator \( (q_\phi) \) to maximize \( \log q_\phi(\omega|s_{t+1}) \) with SGD.
12:         end for
13:     # Step II: \( \pi_{\theta} \) imitates \( \pi_{\mu} \) with the relabeled goals and the associated rewards (for the same \( \omega \)). Here we only consider the static goal for \( \pi_{\theta} \).
14:         Reset Env. with the relabeled goals \( g_T \) & sample initial state: \( \tilde{s}_0 \sim p_0(\tilde{s}) \).
15:         for \( t = 0, 1, \ldots, T - 1 \) steps do
16:             \( g_t = g_T \) for fixed (static) goals.
17:             Sample action: \( a_t \sim \pi_{\theta}(a_t|s_t, g_t) \).
18:             Step environment: \( \tilde{s}_{t+1} \sim p(\tilde{s}_{t+1}|\tilde{s}_t, a_t) \).
19:             Compute reward \( \tilde{r}_t \) for policy \( \pi_{\theta} \) using (8).
20:             Update policy \( \pi_{\theta} \) to maximize \( \tilde{r}_t \) with SAC.
21:         end for
22: end while
```

9.2 Environment Details

We introduce the details of environments and tasks here, including the environment setting of 2D navigation (x-y goal) and color-shape goal), object manipulation, three atari games (seaquest, berzerk and montezuma revenge), and the mujoco tasks (swimmer, half cheetah and fetch).

**2D navigation tasks:** In 2D navigation tasks, the agent moves in each of the four cardinal directions, where the states denote the 2D location of the agent. We consider the following two tasks: moving the agent to a specific coordinate named x-y goal and moving the agent to a specific object with certain color and shape named color-shape goal.

- **2D Navigation (x-y goal):** The size of the environment is \( 10 \times 10 \) (continuous state space) or \( 7 \times 7 \) (discrete state space). The state is the location of the agent, and the goal is the location of the final location.
- **2D Navigation (color-shape goal):** The size of the environment is \( 10 \times 10 \). The state consists of the locations of the agent and three objects with different color-shape pairs (one real target and two distractors). The goal is described by the color and shape of the real target, encoded with one-hot.

**Object manipulation:** More complex manipulation considers a moving agent in 2D environment with one block for manipulation, and the other as a distractor. The agent first needs to reach the block and then move the block to the given location, where the block is described using color and shape. The size of the environment is \( 10 \times 10 \). The state consists of the locations of the agent and two blocks with different color-shape pairs (one real target and one distractor). The goal consists of the one-hot encoding of the color-shape of the target block that needs to be moved, and the 2D coordinate of the final location of the movement.

| Environments                  | k       |
|-------------------------------|---------|
| Seaquest-ram-v0              | 2, 3, 4, 5 |
| Berzerk-ram-v0               | 34, 36, 38, 40 |
| MontezumaRevenge-ram-v0      | 2, 3, 4, 5 |

**Atari games:** We test the performance on three atari games: seaquest, berzerk, and montezuma revenge. In order to reduce the difficulty of training, we adopt the RAM-environment (i.e., Seaquest-ram-v0, Berzerk-ram-v0, and MontezumaRevenge-ram-v0), where each state represents a 128-dimensional vector. Each action repeatedly performs for a duration of \( k \) frames, where \( k \) is uniformly sampled from Table 2.
Mujoco tasks: We consider to make diverse agents to fast imitate a given goal trajectory, including the imitation of behaviors of a swimmer, a half cheetah, and a fetch, where states in the trajectory denote positions of agents. Such experiments are conducted to demonstrate the effectiveness of our proposed method in learning behaviors over a continuous high-dimensional action space, which is more complicated in physics than the 2D map.

Note that the goals in all the experiments are images $50 \times 50 \times 3$ (3 channels, RGB) in size, except that the color-shape goal is encoded with one-hot.

9.3 Tracking Hand Movement

For the dynamic (time-varying) goals, we implement the hand movement tracking with MediaPipe (Lugaresi et al. 2019) to extracting features of hand from images. As shown in Figure 17, we adopt two cameras to get the up-view and the left-view of hand, respectively. Then the two views are processed by MediaPipe to locate the hand, which allows us to get the position in three dimensions. Finally, we make the panda robot track the extracted 3D coordinate with the trained goal-conditioned policy $\pi_\theta$.

9.4 Metrics, Network Architectures and Hyperparameters

Here we give a clear definition of our evaluation metric – “normalized distance to goal”:

I: When the goal is to reach the final state of the trajectory induced by $\pi_\mu$, the distance to goal is the L2-distance between the final state $\hat{s}_T^k$ induced by $\pi_\theta(\cdot | g^k)$ and the goal state $g^k$ randomly sampled from the goal (task) space:

$$\text{Dis} = \frac{1}{N} \sum_{k=1}^{N} \sum_{t=1}^{T} \frac{1}{T} L2(\hat{s}_t^k, g_t^k),$$

where $N$ is the number of testing samples. We set $N = 50$ for 2D navigation, object manipulation and atari games (seaquest, berzerk and montezuma revenge).

II: When the goal is to imitate the whole trajectory induced by $\pi_\mu$, the distance is the expectation of distance over the whole trajectory $\{\hat{s}_0^k, \hat{s}_1^k, ..., \hat{s}_T^k\}$ induced by $\pi_\theta(\cdot | g_t^k)$ and goal trajectory $\{g_0^k, g_1^k, ..., g_{T-1}^k\}$ randomly sampled from the trajectory (task) space:

$$\text{Dis} = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{T} \sum_{t=1}^{T} L2(\hat{s}_t^k, g_{t-1}^k),$$

where $N$ is the number of testing samples. We set $N = 50$ for mujoco tasks (swimmer, half cheetah and fetch).

The term “normalized” means that the distance is divided by a scale factor.

Note that, for three atari games (seaquest, berzerk, and montezuma revenge), the L2-distance for evaluation is the difference between the position of controllable agent and the target’s position, where the position is obtained by matching the pixel on the imaged state.

In our implementation, we use two independent SAC architectures (Haarnoja et al. 2018) for latent-conditioned policy $\pi_\mu$ and goal-conditioned policy $\pi_\theta$. We find empirically that having two networks share a portion of the network structure will degrade the experimental performance. We adopt universal value function approximates (UVFAs) (Schaul et al. 2015) for extra input (goals). For the latent-conditioned policy $\pi_\mu$, to pass latent variable $\omega$ to the Q function, value function and policy, as in D4AYN, we simply concatenate $\omega$ with the current state $s_t$ (and action $a_t$). For goal-conditioned policy $\pi_\theta$, we also concatenate $g_t$ with current state $\hat{s}_t$ (and action $a_t$). We update the $\theta_E$ using the gradients from both the Dis. loss and Q function’s loss of the goal-conditioned policy $\pi_\theta$.

The latent-distributions are provided in Table 3 and the hyper-parameters are presented in Table 4.

10 More Results

10.1 Comparison on the Atari tasks

In Figure 18, we provide more results on the Atari tasks.

---

3The reward function for our baseline L2 Distance still calculates the L2-distance directly on the original state space, instead of the distance of the agents’ positions after pixel matching here.
Table 3: The given latent distribution for each tasks in our experiments.

| Arrow       | 2D navigation | Object manipulation | Atari games |
|-------------|---------------|---------------------|-------------|
| Discrete; 3 skills | Continuous; [-1, 1]² | Continuous; [-1, 1]² | Continuous; [-1, 1]² |
| Half-cheetah Fetch; Panda Maze (Figure 8) 2D navigation (Appendix 1.4) |
| Discrete; 50 skills | Discrete; 10 skills |
| Swimmer Gridworld |
| Discrete; 50 skills | Discrete; 4 skills |

Figure 18: Performance (normalized distance to goals vs. actor steps) of our GPIM and baselines.

10.2 Learned Behaviors on temporally-extended tasks

More experimental results are given in Figure 19 to show the imitation on several temporally-extended tasks.

Figure 19: Expert behaviors and learned behaviors. The four expert trajectories are described parametrically as: 

\[
(x, y, z) = (\log_{10}(t + 1) + t/50, \sin(t)/5 + t/5, t/5),
\]

\[
(x, y, z) = (t/5, \cos(t)/5 - 1/5 + t/5, \sin(t)/5),
\]

\[
(x, y, z) = (\cos(t)/5 + t/50 - 1/1, \sin(t)/5 + t/5, t/5),
\]

and

\[
(x, y, z) = (\sin(t) - \sin(2t)/2, -t/5, \cos(t)/2 - \cos(2t)/2).
\]

10.3 Learned Behaviors from GPIM

More experimental results are given in Figure 20 to show the learned behaviors on 2D navigation, object manipulation, three atari games (seaquest, berzerk and montezuma revenge), and the mujoco tasks (swimmer, half cheetah and fetch). Videos are available under https://sites.google.com/view/gpim.

Broader Impact

The main concern in the research area of unsupervised RL is how to find numerous diverse goals as well as associated reward functions. Particularly, in practical application, the goal and state are often heterogeneous data with high variability of data types and formats, which further aggravates the difficulty of reward learning. Our model addresses these problems by introducing a latent-conditioned policy and the relabeling procedure, scoring current states with latent variables, instead of goals. Although the manifold spaces of goals and latent variables are inconsistent, we show that this relabeling procedure can still effectively learn policy.
However, by autonomous exploration of the environment, the agent is likely to generate some useful behaviors as well as the majority of useless skills. Of particular concern is that the use of autonomous exploration is likely to generate a strategy that will induce dire consequences, such as a collision skill in an autonomous driving environment. How to generate useful behaviors for user-specified tasks and how to use these induced skills are also open problems. An alternative solution is to use an extrinsic reward or the offline data (IRL, offline RL) to guide the exploration. While another issue comes from the exploration-exploitation trade-off. We would encourage further work to understand the limitations of GPIM interacting with the environment autonomously. We would also encourage research to understand the risks arising from autonomous robot learning.

Another limitation is that our learning framework needs an extra latent-conditioned policy training for relabeling goals and providing a reward function. This may require twice as much interaction time with the environment as learning a single policy network. For practical application, a simulation platform is preferred for such expensive training. Thus, one promising direction is to mitigate the difference between the simulation platform and the actual environment. We also encourage researchers to pursue the effective methods for transfer learning.
Table 4: Hyper-parameters

| Hyper-parameter      | value  |
|----------------------|--------|
| Batch Size           | 256    |
| Discount Factor      | 0.99   |
| Buffer Size          | 10000  |
| Smooth coefficient   | 0.05   |
| Temperature          | 0.2    |

| Learning Rate        | value  |
|----------------------|--------|
| 2D Navigation        | 0.0001 |
| x-y goal             | 0.0001 |
| color-shape goal     | 0.0001 |
| Object Manipulation  | 0.0001 |

| Learning Rate        | value  |
|----------------------|--------|
| Mujoco tasks         |        |
| Swimmer              | 0.0001 |
| HalfCheetah          | 0.0001 |
| Fetch                | 0.0001 |
| Seaquest             | 0.0003 |
| Berzerk              | 0.0003 |
| Montezuma Revenge    | 0.0003 |

| Learning Rate        | value  |
|----------------------|--------|
| Atari games          |        |
| x-y goal             | 20     |
| color-shape goal     | 20     |
| Object Manipulation  | 20     |

| Path Length          | value  |
|----------------------|--------|
| 2D Navigation        | 20     |
| x-y goal             | 20     |
| color-shape goal     | 20     |
| Object Manipulation  | 20     |

| Path Length          | value  |
|----------------------|--------|
| Mujoco tasks         |        |
| Swimmer              | 50     |
| HalfCheetah          | 50     |
| Fetch                | 100    |
| Seaquest             | 25     |
| Berzerk              | 25     |
| Montezuma Revenge    | 25     |

| Path Length          | value  |
|----------------------|--------|
| Atari games          |        |
| x-y goal             | 128    |
| color-shape goal     | 128    |
| Object Manipulation  | 128    |

| Hidden Size          | value  |
|----------------------|--------|
| 2D Navigation        | 128    |
| x-y goal             | 128    |
| color-shape goal     | 128    |
| Object Manipulation  | 128    |

| Hidden Size          | value  |
|----------------------|--------|
| Mujoco tasks         |        |
| Swimmer              | 256    |
| HalfCheetah          | 256    |
| Fetch                | 256    |
| Seaquest             | 256    |
| Berzerk              | 256    |
| Montezuma Revenge    | 256    |

| Hidden Size          | value  |
|----------------------|--------|
| Atari games          |        |
| x-y goal             | 2      |
| color-shape goal     | 2      |
| Object Manipulation  | 4      |

| Dimension of Generative Factor | value  |
|--------------------------------|--------|
| Mujoco tasks                   |        |
| Swimmer                        | 3      |
| HalfCheetah                    | 4      |
| Fetch                          | 4      |
| Seaquest                       | 16     |
| Berzerk                        | 16     |
| Montezuma Revenge              | 16     |

| Dimension of Generative Factor | value  |
|--------------------------------|--------|
| α                              | 1      |
| γ                              | 5      |
| δ_{pixel}                      | 255    |
Figure 20: Discovered goal-conditioned behaviors. (f-h): The left subfigure shows the expert behaviors (goals); The middle subfigure shows the learned behaviors by GPIM; The right subfigure is the stacked view of goals and GPIM behaviors.