# LEARN GOAL-CONDITIONED POLICY WITH INTRINSIC MOTIVATION FOR DEEP REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

It is of significance for an agent to learn a widely applicable and general-purpose policy that can achieve diverse goals such as images. Considering such perceptually-specific goals, the frontier of deep reinforcement learning research is to learn a goal-conditioned policy without hand-crafted reward. To learn this kind of policy, recent works usually take as reward the non-parametric distance to the given goal in an explicit embedding space. From a different viewpoint, we propose a novel unsupervised learning approach named as goal-conditioned policy with intrinsic motivation (GPIM), which jointly learns both abstract-level policy and goal-conditioned policy. The abstract-level policy is conditioned on a latent variable to provide intrinsic reward and discover diverse abstract skills that are further rendered into perceptually-specific goals for goal-conditioned policy, where both policies *share* an identical reward function. Experiments on a variety of robotic tasks show the effectiveness and efficiency of our proposed GPIM method, which substantially outperforms prior techniques.

## 1 INTRODUCTION

Reinforcement learning (RL) makes it possible to train agents capable of achieving sophisticated goals in complex and uncertain environments, from games (Badia et al., 2020; Berner et al., 2019) to robot control (Lee et al., 2018; Lowrey et al., 2018; Vecerik et al., 2019; Popov et al., 2017), which usually involves learning a specific policy for individual task relying on task-specific reward. However, autonomous agents are expected to exist persistently in the world and have the ability to solve diverse tasks, where we need to build universal reward function and mechanism to automatically generate the tasks.

Tasks in RL can be weakened to achieve different goals. Raw sensory inputs such as images have been considered as common goals for agents to practice and achieve (Watter et al., 2015; Florensa et al., 2019; Nair et al., 2018; 2019). Previous works make full use of a goal achievement reward function as an available prior (Kaelbling, 1993), such as the Euclidean distance. Unfortunately, these kind of measurement in original space are not very effective for visual tasks since the distance between images does not correspond to meaningful distance between states (Zhang et al., 2018). Further, the measure function is applied in the embedding space, where the representations of raw sensory inputs are learned by means of using a latent variable model like the VAE (Higgins et al., 2017b; Nair et al., 2018) or using the contrastive loss (Sermanet et al., 2018; Warde-Farley et al., 2018). We argue that these non-parametric approaches obtaining reward in original or embedding space as above may limit the repertoires of behaviors and impose a manual engineering burden (Pong et al., 2019).

In the absence of any prior knowledge about the measure function, unsupervised method in RL is one that learns latent-conditioned policy by maximizing an information theoretic objective or taking self-consistent trajectory autoencoder (Co-Reyes et al., 2018; Hausman et al., 2018). However, this procedures, similar to parameter initialization or hierarchical RL, need external reward function and cannot be applied directly to user-specified goals. Although the learned discriminator in Eysenbach et al. (2018) can be adopt to estimate which skill was most likely to have generated the states, the user-specified goals are usually heterogeneous to the states in training. Sharma et al. (2019) adopts model-based approaches while requires additional training of the model of the dynamics.

Different from previous works, we propose a novel unsupervised DRL scheme to learn goal-conditioned policy by jointly learning an extra abstract-level policy conditioned on latent variables. The abstract-level policy is trained to generate diverse abstract skills while the goal-conditioned policy is trained to efficiently achieve perceptually-specific goals that are rendered from states induced by the corresponding abstract skill. Both abstract-level and goal-conditioned policies *share an identical intrinsic reward function* that distinguishes the states induced by abstract skill. This shared reward function is generated within our framework and optimized in an unsupervised manner for the purpose of reliable exploration (Salge et al., 2014). In essence, the abstract skill can reproducibly influence the environment, and the goal-conditioned policy perceptibly *imitates* these influences. To improve the generalization ability of goal-conditioned policy in dealing with perceptually-specific inputs, a latent variable model is further considered in our GPIM to disentangle the goal into latent generative factors.

The main contribution of our work is a unsupervised RL method that can learns perceptually-specific goal-conditioned policy without extrinsic reward for autonomous agents. We propose a novel training procedure for this model, which provides interpretable reward function for various perceptual goals, e.g. images and text descriptions. Furthermore, we introduce the information bottleneck for learning the representation of high-dimensional goals, and demonstrate the potential of our model to generalize behavior across new tasks. Extensive experiments and detailed analysis demonstrate the effectiveness and efficiency of our proposed method.

## 2 PRELIMINARIES

**RL:** An agent interacts with an environment and selects actions in RL so as to maximize the expected amount of reward received in the long run (Sutton & Barto, 2018), which can be modeled as a Markov decision process (MDP) (Puterman, 2014). An MDP is defined as a tuple $\mathcal{M} = (S, A, p, R, \gamma)$, where $S$ and $A$ are the state and action spaces, $p(\cdot|s, a)$ gives the next-state distribution upon taking action $a$ in state $s$, $R(s, a, s')$ is a random variable representing the reward received at transition $s \xrightarrow{a} s'$, and $\gamma \in [0, 1)$ is a discount factor.

**Intrinsic Motivation:** RL with intrinsic motivation obtains the intrinsic reward by maximizing the mutual information between latent variables $\omega$ and agent's behaviors $\tau$: $\mathcal{I}(\omega; \tau)$, where the specific manifestation of $\tau$ can be the entire trajectory (Achiam et al., 2018), each state (Eysenbach et al., 2018) or the final state (Gregor et al., 2016); and the specific implementation includes *reverse* and *forward* forms (Campos et al., 2020).

**Disentanglement:** Given an observation $x$ with a dimension of $N$, $\beta$-VAE (Higgins et al., 2017a) is a latent model that pairs a top-down encoder $q_E(z|x)$ with bottom-up decoder network $p_D(x|z)$ by introducing a latent factor $z$. To encourage that the inferred latent factor $q_E(z|x)$ captures the generative factor of $x$ in a disentangled manner, an isotropic unit Gaussian prior $p(z) \sim N(0; I)$ is used as common to control the capacity of the information bottleneck (Burgess et al., 2018) by minimizing the KL divergence between $q_{\vartheta_E}(z|g_t)$ and $p(z)$.

## 3 METHOD

In this section, firstly we formalize the problem and introduce the framework. Secondly, we elaborate the process of how to jointly learn goal-conditioned policy and abstract-level policy. Disentanglement is then applied in our problem setting to improve the generalization ability.

### 3.1 OVERVIEW

Given perceptually-specific high-dimensional goal $g$, our objective is to learn a goal-conditioned policy $\pi_\theta(\tilde{a}|\tilde{s}, g)$ that inputs state $\tilde{s}$ and $g$ and outputs action $\tilde{a}$, where the reward is obtained within the proposed framework as shown in Figure 1. Abstract-level policy $\pi_\mu(a|s, \omega)$ takes as input state $s$ and a latent variable $\omega$ and outputs action $a$, where $\omega$ corresponds to diverse latent skills. Two policies $\pi_\mu$ and $\pi_\theta$ share the same reward network $q_\phi$, and all these three components are jointly optimized. The shared reward function is generated by distinguishing the states induced by different skill $w$, which is optimized for reliable exploration *at the abstract level*. Abstract-level policy is

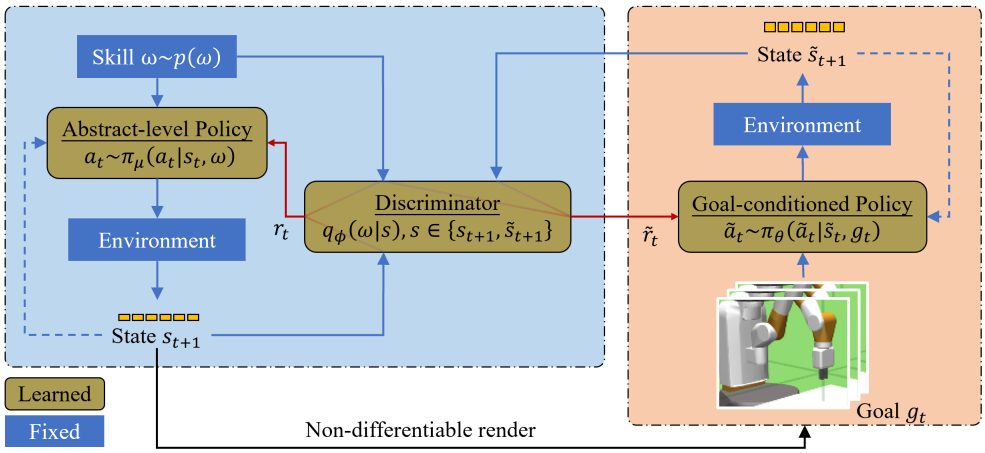



Figure 1: Framework of our GPIM method. Abstract-level policy $\pi_\mu$ and goal-conditioned policy $\pi_\theta$ share an identical reward network (i.e., discriminator) $q_\phi$. State $s_t$ (e.g., position of an agent) from $\pi_\mu$ is converted into a perceptually-specific goal $g_t$ (e.g., image showing the position of the agent) for $\pi_\theta$. Initial state $s_0$ of $\pi_\mu$ and $\tilde{s}_0$ of $\pi_\theta$ are sampled from same distribution.

able to generate diverse skills and state $s$ that can be further rendered as perceptually-specific goal $g = \textit{Render}(s)$ (Brockman et al., 2016). On this basis, goal-conditioned policy $\pi_\theta\left(\tilde{a}|\tilde{s}, g\right)$ can be learned by achieving diverse high-dimensional goals with only this intrinsic reward.

### 3.2 Proposed GPIM Method

In order to jointly learn abstract-level policy $\pi_\mu(a|s, \omega)$ and goal-conditioned policy $\pi_\theta\left(\tilde{a}|\tilde{s}, g\right)$, we maximize the mutual information between states $s$ and the latent variable $w$ for $\pi_\mu$, and simultaneously maximize the mutual information between states $\tilde{s}$ and high-dimensional goal $g$ for $\pi_\theta$. Consequently, the overall objective function can be expressed as follows

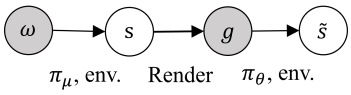



Figure 2: Graphical model.

$$\max \mathcal{F}(\mu, \theta) = \mathcal{I}(s; \omega) + \mathcal{I}(\tilde{s}; g). \tag{1}$$

For clarification, Figure 2 depicts the graphical model for the latent variable $\omega$, states $s$ induced by $\pi_\mu$, goal $g$ rendered from $s$ and states $\tilde{s}$ induced by $\pi_\theta$. As seen, the latent variable $\omega \sim p(\omega)$[1] is firstly used to generate states $s$ via the policy $\pi_\mu$ interacting with the dynamic environment. Then, we use states $s$ to generate high-dimensional goal $g$. After that, the goal-conditioned policy $\pi_\theta$ outputs action $\tilde{a}$ to interact with the environment to obtain states $\tilde{s}$ at next time step. In particular, $\omega$ is expected to generate diverse behavior modes through $\pi_\mu$ while $\pi_\theta$ behaving like $\tilde{s}$ is to imitate this behavior by taking as input the rendered goal $g$. Based on the context, the correlation between $\tilde{s}$ and $g$ is no less than that between $\tilde{s}$ and $w$ , so that $\mathcal{I}(\tilde{s}; g) \geq \mathcal{I}(\tilde{s}; \omega)$ (Beaudry & Renner, 2011). So we can obtain the lower bound of $\mathcal{F}(\mu, \theta)$:

$$\begin{aligned} \mathcal{F}(\mu, \theta) &\geq \mathcal{I}(s; \omega) + \mathcal{I}(\tilde{s}; \omega) \\ &= 2\mathcal{H}(\omega) - \mathcal{H}(\omega|s) - \mathcal{H}(\omega|\tilde{s}) \\ &\triangleq \mathbb{E}_{\omega, s, \tilde{s}}\left[\log p(\omega|s) + \log p(\omega|\tilde{s}) - 2\log p(\omega)\right]. \end{aligned} \tag{2}$$

Since it is difficult to exactly compute the posterior distribution $\log p(\omega|s)$ and $\log p(\omega|\tilde{s})$, Jensen's Inequality (Barber & Agakov, 2003) is further applied for approximation by using a learned discriminator network $q_\phi(\omega|\cdot)$. Thus,

$$\mathcal{F}(\mu, \theta) \geq \mathbb{E}_{\omega, s, \tilde{s}}\left[\log q_\phi(\omega|s) + \log q_\phi(\omega|\tilde{s}) - 2\log p(\omega)\right] \triangleq \mathcal{J}(\mu, \phi, \theta), \tag{3}$$

where it is worth noting that the identical discriminator $q_\phi$ with the parameter $\phi$ is used for the variational approximation of $\log p(\omega|s)$ and $\log p(\omega|\tilde{s})$. For states $s$ induced by skill $w$ and $\tilde{s}$ originating from $g$, the shared discriminator $q_\phi$ assigns a similarly high probability on $w$ for both states

[1] p($\omega$) denotes the prior distribution. Fixing $p(\omega)$ instead of learning it is to prevent $\pi_\mu$ from collapsing to sampling only a handful of skills (Eysenbach et al., 2018).

**Algorithm 1** Learning process of our proposed GPIM

**while** not converged **do**
  Sample skill $\omega \sim p(\omega)$.
  *# Step I*
  Sample initial state $s_0 \sim p_0(s)$.
  **for** $t \leftarrow 0$ **to** $steps_per_episode$ **do**
    Sample action $a_t \sim \pi_\mu(a_t|s_t, \omega)$.
    Step environment: $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$.
    Render goal: $g_t = \textit{Render}(s_{t+1})$.
    Compute reward $r_t$ for policy $\pi_\mu$ using (5).
    Update policy $\pi_\mu$ to maximize $r_t$ with SAC.
    Update discriminator ($\phi$) to maximize $q_\phi(\omega|s_{t+1})$ with SGD.
  **end for**
  *# Step II*
  Sample initial state $\tilde{s}_0 \sim p_0(\tilde{s})$.
  **for** $t \leftarrow 0$ **to** $steps_per_episode$ **do**
    Sample action $\tilde{a}_t \sim \pi_\theta(\tilde{a}_t|\tilde{s}_t, g_t)$.
    Step environment: $\tilde{s}_{t+1} \sim p(\tilde{s}_{t+1}|\tilde{s}_t, \tilde{a}_t)$.
    Compute reward $\tilde{r}_t$ for policy $\pi_\theta$ using (7).
    Update policy $\pi_\theta$ to maximize $\tilde{r}_t$ with SAC ($\theta = \{\vartheta_E, \vartheta_G\}$ when considering the disentanglement).
    Update $\vartheta_E$ and $\vartheta_D$ by maximizing *Dis_loss* in (8) using SGD.
  **end for**
**end while**

$s$ and $\tilde{s}$ associated with the same $\omega$. Therefore, $q_\phi$ can be regarded as a reward network shared by abstract-level policy $\pi_\mu(a|s, \omega)$ and goal-conditioned policy $\pi_\theta(\tilde{a}|\tilde{s}, g)$.

According to the objective function $\mathcal{J}(\mu, \phi, \theta)$ in (3), we propose an alternating optimization between $\pi_\mu$ and $\pi_\theta$ as follows:

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

$$\mathcal{J}(\mu, \phi) = \mathbb{E}_{\omega,s}\left[\log q_\phi(\omega|s)\right] + \underbrace{\mathbb{E}_{\omega,\tilde{s}}\left[\log q_\phi(\omega|\tilde{s}) - 2\log p(\omega)\right]}_{\text{Variable independent term}}. \tag{4}$$

According to (4), $\mathcal{J}(\mu, \phi)$ can be thus optimized by setting the reward at time step $t$ for $\pi_\mu$ as

$$r_t = \log q_\phi(\omega|s_{t+1}) - \log p(\omega), \tag{5}$$

where the term $-\log p(\omega)$ is added for agents to avoid artificial termination and reward-hacking issues (Amodei et al., 2016). We implement this optimization with soft actor-critic (SAC). On the other hand, the reward network $q_\phi$ can be updated by SGD to maximize $\log q_\phi(\omega|s_{t+1})$.

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

$$\mathcal{J}(\theta) = \mathbb{E}_{\omega,\tilde{s}}\left[\log q_\phi(\omega|\tilde{s})\right] + \underbrace{\mathbb{E}_{\omega,s}\left[\log q_\phi(\omega|s) - 2\log p(\omega)\right]}_{\text{Variable independent term}}. \tag{6}$$

According to (6), $\mathcal{J}(\theta)$ can be thus optimized by setting the reward at time step $t$ for $\pi_\theta$ as

$$\tilde{r}_t = \log q_\phi(\omega|\tilde{s}_{t+1}) - \log p(\omega), \tag{7}$$

where the term $-\log p(\omega)$ is added for the same reason as above and we also implement this optimization with SAC. Iterate two steps until the convergence. The overall GPIM method is summarized in Algorithm 1, which includes the disentanglement (*red*) in the next subsection.

### 3.3 Improve Generalization via Disentanglement

The goal $g_t$ can be further disentangled (Higgins et al., 2017a) so as to improve the generalization ability of goal-conditioned policy $\pi_\theta$. As shown in Figure 3, we decompose $\pi_\theta$ into two components: the encoder network E parameterized by $\vartheta_E$ and the generative network G parameterized by $\vartheta_G$. So we have $\pi_\theta(\tilde{a}_t|\tilde{s}_t, g_t) = p_{\vartheta_G}(\tilde{a}_t|\tilde{s}_t, q_{\vartheta_E}(z|g_t))$.

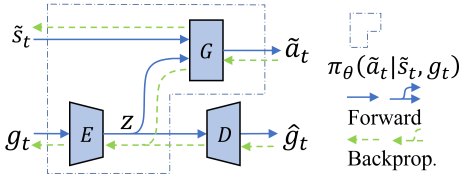



Figure 3: Disentanglement of policy $\pi_\theta$.

To encourage this disentangling property in the inferred $q_{\vartheta_E}(z|g_t)$, we adopt an prior $p(z) \sim N(0; I)$ to control the capacity of the information bottleneck by minimizing the KL divergence between $q_{\vartheta_E}(z|g_t)$ and $p(z)$. Hence, the overall objective function becomes

$$\max_{\vartheta_E,\vartheta_G,\vartheta_D} \quad \mathbb{E}_{\omega,\tilde{s}_t}\left[\tilde{r}_t\right] + \underbrace{\alpha \cdot \mathbb{E}_{g_t,z}\left[p_{\vartheta_D}(g_t|z)\right] - \beta \cdot \mathbb{E}_{g_t}\left[KL(q_{\vartheta_E}(z|g_t)||p(z))\right]}_{\triangleq Dis_loss}, \tag{8}$$

where $\alpha$ 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 *Dis_loss*. With this disentanglement, the overall GPIM method is given in Algorithm 1 as mentioned above.

## 4 Related Work

*Investigate goal distribution:* Many prior methods (Schaul et al., 2015; Andrychowicz et al., 2017; Pong et al., 2018; Hartikainen et al., 2019) assume an available distribution of goals during exploration for sparse-reward problems, where the resampled goals are restricted to states encountered along the trajectory (Andrychowicz et al., 2017; Levy et al., 2017) or in an embedding space (Nair et al., 2018). And the setting of goals for learning policy has been discussed in (Baranes & Oudeyer, 2013). Recently, several works (Colas et al., 2018; Warde-Farley et al., 2018; Florensa et al., 2019; Nair et al., 2018; Péré et al., 2018) adopt heuristics to design a goal distribution based on previously visited states. However, many of these (Nair et al., 2018; Pong et al., 2019; Warde-Farley et al., 2018) assume an ideal matching of the goal space with the state space.

*Learn goal-conditioned policy:* Several recent works propose various methods for agents to automatically learn policies: maximizing the mutual information between behaviors (e.g., latent variables) and tasks (e.g., goals). DISCERN (Warde-Farley et al., 2018) simultaneously learns a goal-conditioned policy and a goal achievement reward function that measures how similar a state is to the goal state. Skew-Fit (Pong et al., 2019) concurrently trains a goal-reaching policy and maximizes the entropy of the generated goal distribution. Given the difficulty of vision-based RL, Skew-Fit resorts to RIG (Nair et al., 2018), which obtains reward using prior non-parametric measure function in embedding spaces. Other unsupervised methods (Eysenbach et al., 2018; Sharma et al., 2019; Campos et al., 2020), while able to learn diverse skills, are difficult to apply directly to user-specified tasks.

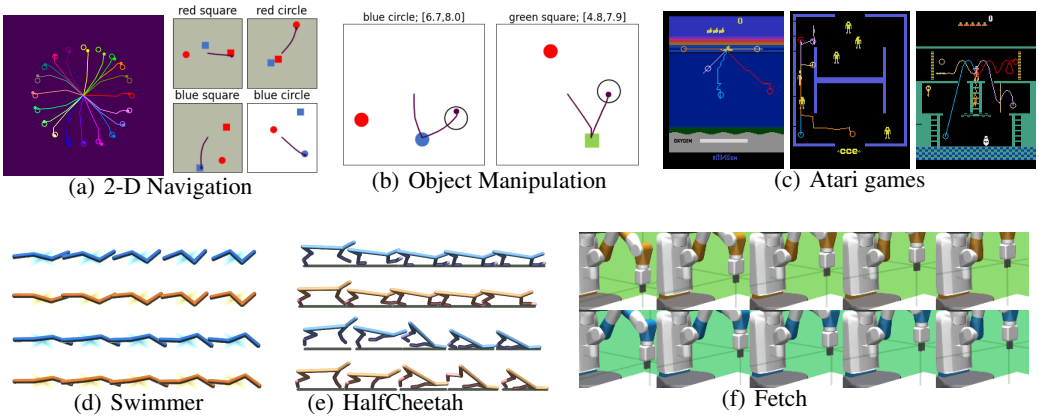
*Learn disentanglement representation:* Many prior works utilize unsupervised learning in DRL to acquire disentanglement representation (Bengio et al., 2013; Higgins et al., 2017a; Chen et al., 2016; Burgess et al., 2018). Some works take the disentanglement representation as a substitute for states (Higgins et al., 2017b; Ha & Schmidhuber, 2018; Jonschkowski et al., 2017; Finn et al., 2016; Lange et al., 2012; Watter et al., 2015; Srinivas et al., 2018; Nair et al., 2019; Srinivas et al., 2018) while others consider the no-parametric metric in the latent space to acquire a reward function (Nair et al., 2018; Sermanet et al., 2018). Besides, a few works directly embed manipulation tasks to generalize to long-horizon tasks (Goyal et al., 2019; Hausman et al., 2018; Mandlekar et al., 2020).

*Hindsight, self-play and knowledge distillation:* Our method is similar in spirit to goal relabeling methods, liking hindsight experience replay (HER) (Andrychowicz et al., 2017) which replays each episode with a different goal than the one the agent was trying to achieve. However, HER requires prior reward function and a hand-crafted goal space. By contrast, GPIM is unsupervised and finds its own goal space. This self-play (Sukhbaatar et al., 2017; 2018) is closely related to our model, leading to emergent autocurricula for training abstract-level policy and goal-conditioned policy whose tasks and reward function are both constructed at abstract level. As for knowledge distillation (Phuong & Lampert, 2019), we aim at the extraction of the relationship between two different tasks.

## 5 Experiments

Experiments are conducted to evaluate our proposed GPIM method and compare to state-of-the-art baselines. In this section, we mainly answer the following five questions[2]: 1) Does the discriminator $q_\phi$ provide effective reward function for learning goal-conditioned policy $\pi_\theta$? 2) Does our model learn effective behaviors conditioned on a variety of goals, including images and text descriptions? 3) What happens when the goals and states are heterogeneous? 4) Does the proposed GPIM on learning goal-conditioned policy outperform baselines? 5) How does the disentanglement of goals

[2]Due to the page limitation, more experimental analysis and results are included in the Appendix.




(a) 2-D Navigation (b) Object Manipulation (c) Atari games

(d) Swimmer (e) HalfCheetah (f) Fetch

Figure 6: Goals and learned behaviors by GPIM. Dots in 2D navigation (left, x-y goal) and atari games denotes the final goal state, and curves of the same color represent corresponding trajectories. Goals in 2D navigation (right, color-shape goal) and object manipulation are at the top of the diagram, wherein the purple lines imply the behaviors. In mujoco tasks, the first (swimmer, half cheetah and fetch) and third (swimmer and half cheetah) rows represent the target trajectories, and the following represent the corresponding behaviors.

in GPIM affect the learned behaviors in terms of the generalization to new tasks? Video is available under https://sites.google.com/view/gpim.

**Illustrative example.** We start with a simple gridworld example, the environment shown on the left of Figure 4, where the goal for the robot is to navigate from the middle to the given colored room. By abstract-level training, our method quickly acquire four skills to reach different rooms. Each time the robot arrive in a room, we target the color of the room, allowing the agent to be guided to the same room by the current reward function. The results are shown on the right of Figure 4, where the top and bottom of results show the learning processes of abstract-level policy and goal-conditioned policy respectively.

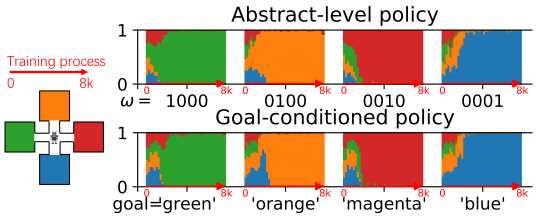



Figure 4: The Y-axis is the percentage of different rooms that the robot arrives in.

**Visualizing the learned reward function.** We use a simple "archery" task to visualize how the learned reward function (discriminator $q_\phi$) accounts for goal-conditioned behaviors in dynamics. The task, shown in Fig.5, requires choosing an angle at which to shoot an arrow given the target. The left figure shows that in a deterministic environment, given 3 different targets with different colors, the arrows reach the corresponding targets successfully under the instruction of learned reward function $q_\phi$. The reward as a function of the final location of arrows in the 3 tasks is shown in the right. We can find that the learned reward function is close to a quadratic function of a distance between the final state to the target, with the opening downwards and the vertices corresponding to the target. We observe that the maximum value of learned reward function is close to the given target and minimum value is far from the given target, which demonstrates the effectiveness of our learned reward function. Similarly, we can draw the same conclusion in the stochastic environment, wherein the angle of the arrow has a 50% chance of becoming a mirror symmetric angle, as shown in the bottom left.

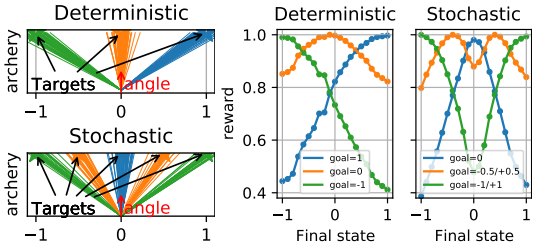



Figure 5: "Archery" tasks.

**Scaling to more complex tasks.** We now apply GPIM to the more complex tasks shown in Fig.6. 1) In 2D navigation tasks, the agent moves in each of the four cardinal directions. We consider the following two tasks: moving the robot to a specific coordinate named as *x-y goal* and moving

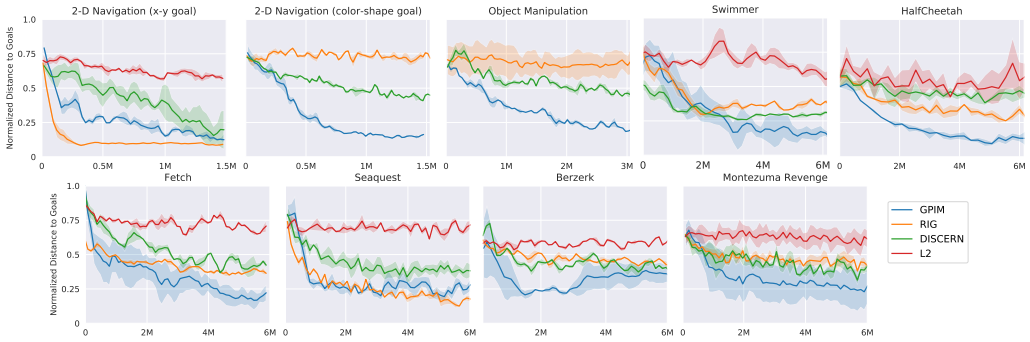



Figure 7: Performance of our GPIM and three baselines, where GPIM generally achieves the smaller distance to the goal in comparison to baselines.

the robot to a specific object with certain color and shape named as *color-shape goal*. 2) Object Manipulation considers a moving agent in 2D environment and one block for manipulation, where another block exists as a distractor. The manipulation might be arrived at the block and moved the block to the given location, where the block is described using color and shape. The description of the goal contains the *color-shape prior* of real block and the *x-y prior* of the target coordinate. 3) Three atari games (seaquest, berzerk and montezuma revenge) require the agent to imitate the final state. 4) We use three Mujoco robots (swimmer, half cheetah, and fetch) taken from OpenAI GYM (Brockman et al., 2016) to fast imitate a given expert trajectory. Specifically, the goals for $\pi_\theta$ in tasks 2D navigation and atari games are the rendered final states induced by abstract-level policy $\pi_\mu$: $g_t = \textit{Render}(s_T)$; the goals for $\pi_\theta$ in mujoco tasks are the rendered trajectories induced $\pi_\mu$: $\{g1, g2, ...\} = \textit{Render}(\{s_2, s_3, ...\})$. See Appendix F for more details.

The left chart of Figure 6(a) shows the learned behaviors of navigation with continuous action space given *x-y goal* which is noted as the small circle; and the right chart shows the trajectory of behaviors with given color-shape goal, discrete navigation can be found in Appendix E. As observed, the agent manages to learn navigation tasks by using GPIM. Figures 6 (c-f) show the behaviors of agents that reach at the corresponding final states (c) in a higher dimensional (action, state and goal) space and imitates the entire expert trajectories (d-f), respectively. Further, 2D navigation (color-shape goal) and object manipulation tasks show the effectiveness of our model facing heterogeneous goals and states. Specifically, Fig. 6(b) shows the behaviors of the agent on object manipulation, where the agent is asked to firstly arrive at a block (e.g. blue circle in 6(b) *left*) and then push it to the given location shown as a dark circle (e.g. [6.7,8.0] in 6(b) *left*), and the other object exists as a distractor.

**Performance Evaluation.** We compare our method to three baselines: **RIG** (Nair et al., 2018), **DISCERN** (Warde-Farley et al., 2018), and **L2 Distance**. L2 Distance measures the distance between states and goals, where the $L2$ distance $-||s_t - s_g||^2/\sigma_{pixel}$ is considered with a hyperparameter $\sigma_{pixel}$. Note that 2D navigation with *color-shape goal* and object manipulation using text description make the dimensions of states and goals different, so L2 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 another is to encode goals. We use normalized distance to goals as the evaluation metric, wherein randomly sampled 50 samples (tasks) from the goal space. Implementation details and further descriptions can be found in Appendix G.

We show the results of this experiment in Fig.7, plotting the normalized distance to goals as a function of the number of actor 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 the RIG in 2D-coordinate navigation 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) until object manipulation (consists of x-y goal and color-shape goal), GPIM converges faster than RIG and DISCERN 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 have difficulty in capturing the correlation of heterogeneous states and goals. Especially in high-dimensional action space and more exploratory tasks, our method substantially outperform the baselines, for which the effectively exploration for autonomous generating goals poses a major chal-

lenge (see next subsection and Appendix, where we shows the learning process at abstract-level enabling agent to explore environment autonomously).

**Explanation of GPIM.** To gain more intuition for our method, we recorded the distance ($\Delta r$) between the final state induced by $\pi_\theta$ and the goal rendered by abstract-level policy $\pi_\mu$ throughout training on the 2D Navigation (x-y prior) environment. In this experiment, we only update $\pi_\mu$ before 200k steps. Said another way, we ignore $\pi_\theta$ (actor and critic in SAC) for the first 200k steps of dynamic. As shown in Fig.8, $\Delta r$ steadily increases during the first 200k steps, suggesting that the abstract-level policy $\pi_\mu$ is exploring the environment (goal space) for making it easier to distinguish skills, and, as a result, generating diverse goals for training $\pi_\theta$. After around 1.5M training steps $\Delta r$ is almost going to 0: the goal-conditioned policy $\pi_\theta$ has learned the strategies to reach the rendered goals.

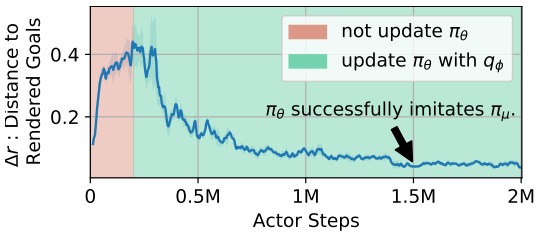



Figure 8: Abstract-level policy $\pi_\mu$ will gradually explore the environment, generating more difficult goals. $q_\phi$ encourages $\pi_\theta$ to gradually mimic $\pi_\mu$.

**Ablation study and disentanglement.** To investigate how the disentanglement affect the learned behaviors, we conducted the ablation study on 2D navigation task [3]. For convenience, we remove certain component from GPIM and define the new method as follows: *w/o Rec* - removing the reconstruction loss (i.e. $\alpha = 0$); *w/o KL* - removing the KL loss (i.e. $\beta = 0$); *w/o Pol* - removing the policy loss updating $\vartheta_E$.

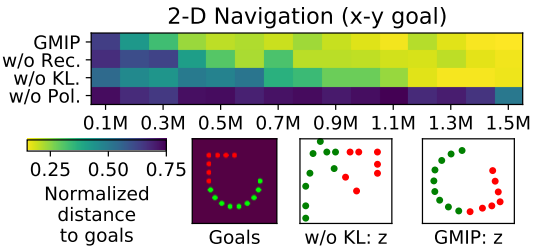



Figure 9: Ablation study and disentanglement.

The performance is reported in Figure 9. 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 disentanglement fails to figure out the required latent factors on given tasks. Moreover, although GPIM has a similar performance with other three methods on 2D navigation task, GPIM has better interpretability to behaviors. As shown at the bottom of Figure 9, considering a series of goals from the first red to the last green (*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*). By learning to reach diverse goals generated by abstract-level policy and then disentangle the goals, the agent learns the ability to infer new goals later encountered by the agent. For example, as in Figure 6(a) (*right*), learning three behaviors with the goal of red-square, red-circle and blue-square in a gray background makes the agent accomplish the new goal of blue-circle in a white background. More examples testing the generalization to unseen goals can be found in Appendix D.

## 6 Conclusion and Future Work

We propose a novel GPIM method to learn goal-conditioned policy in an unsupervised manner. In particular, the goal-conditioned policy is learned by sharing intrinsic reward with abstract-level policy, where abstract-level policy also provides distinguishing but diverse goals for goal-conditioned policy. Two policies and shared reward network are jointly optimized. From experiment results, our GPIM outperforms baselines in terms of accuracy as well as efficiency.

Although mutual information has become a common way to improve exploration in RL, it cannot be ignored that improvement (discovering diverse skills) Salge et al. (2014) provides a improvable convergence of the state space Campos et al. (2020). One promising direction of future work is to improve the exploration triggered by improving the empowerment of abstract-level policy and the exploitation of the distillation process between two policies.

[3] Ablation study on Mujoco tasks can be found in Appendix C.

## REFERENCES


Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, pp. 5048–5058, 2017.

Adria Puigdomenech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

David Barber and Felix V Agakov. The im algorithm: a variational approach to information maximization. In *Advances in neural information processing systems*, pp. None, 2003.

Normand J Beaudry and Renato Renner. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740*, 2011.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Victor A. Campos, A. Trott, Caiming Xiong, R. Socher, Xavier Giro i Nieto, and J. Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. *ArXiv*, abs/2002.03647, 2020.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. *arXiv preprint arXiv:1806.02813*, 2018.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054*, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519. IEEE, 2016.

Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Yoshua Bengio, and Sergey Levine. Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*, 2019.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2019.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. 2018.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017a.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1480–1490. JMLR. org, 2017b.

Rico Jonschkowski, Roland Hafner, Jonathan Scholz, and Martin Riedmiller. Pves: Position-velocity encoders for unsupervised learning of structured state representations. *arXiv preprint arXiv:1705.09805*, 2017.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993.

Sascha Lange, Martin Riedmiller, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. In *The 2012 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2012.

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2018.

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

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. *arXiv preprint arXiv:1910.11670*, 2019.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018.

NeurIPS2019. Neurips 2019 - robot open-ended autonomous learning (real). https://nips.cc/Conferences/2019/CompetitionTrack, 2019.

Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*, 2018.

Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, pp. 5142–5151, 2019.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pp. 67–114. Springer, 2014.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320, 2015.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. Learning goal embeddings via self-play for hierarchical reinforcement learning. *arXiv preprint arXiv:1811.09083*, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

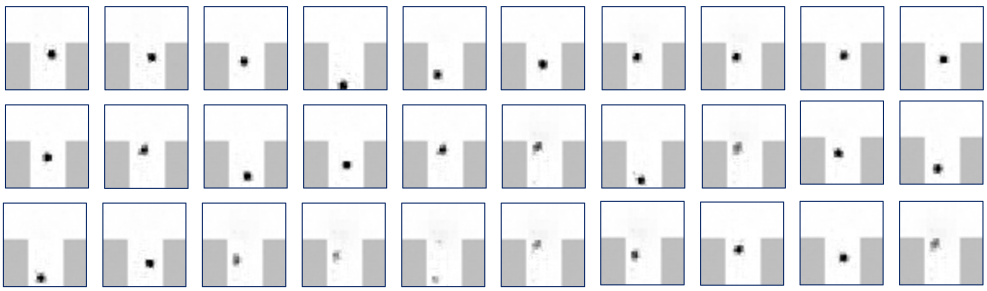
Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester, and Jon Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 754–760. IEEE, 2019.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

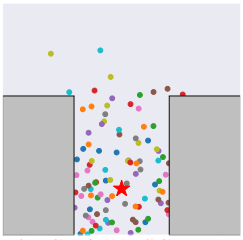
Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pp. 2746–2754, 2015.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

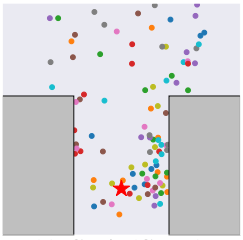
# A ADDITIONAL EXPERIMENT: AUTOMATED GOAL-GENERATION FOR EXPLORATION



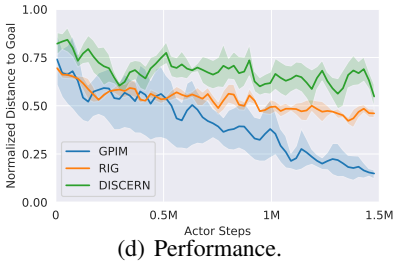
(a) Decoded latent goals (RIG).



(b) Goals (DISCERN).



(c) Goals (GPIM).




(d) Performance.

Figure 10: Distribution of sampled goals for goal-conditioned policy, where the initial state is [5,2] (the red star) and the actor step is 20 for each rollout. (a) The goals (images) are obtained by decoding 30 sampled latent goals $z_g$ in VAE framework; (b) The goals (colored dots) are sampled from the agent's behaviors by random exploration; (c) The goals (colored dots) are rendered from the states induced by our abstract-level policy. (d): Evaluation on reaching user-specified goals, where GPIM significantly outperforms baselines.

In general, for unsupervised RL, we would like to ask our the agent to carry out autonomous "practice" during training phase, where we do not know which particular goals will be provided in the 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 (DISCERN (Warde-Farley et al., 2018), RIG (Nair et al., 2018), GPIM) in a new 2D obstacle navigation task as shown in Figure A. 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 has the size of $3 \times 6$.

DISCERN samples goals during training by maintaining a fixed size buffer $\mathcal{G}$ of past observations. We simply mimic the process of goal generation by taking random action for 20 environment steps after initialization of the method. As in Figure 10(b), we generate 100 goals with different colors. We can see that the majority of goals locate between the two obstacles, which limits the further exploration of the environment.

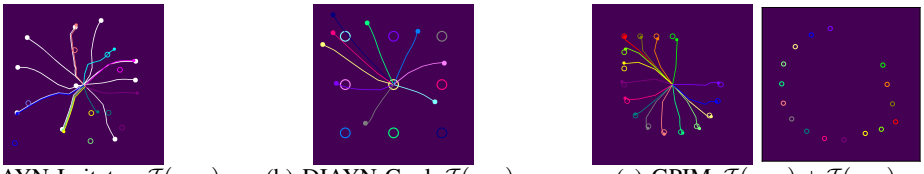
RIG samples representation (latent goals $z_g$) from the learned VAE prior, which represents a distribution over latent goals and state observation. The policy network takes the representation as a substitute for the user-specified goal. For a clear visualization of the distribution of the sampled latent goals, we further feed the sampled latent goals into the decoder to obtain the real goals in the user-specified space. The decoded latent goals are shown in Figure 10(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 VAE prior come from the same distribution as in DISCERN.

Our method, GPIM, obtains goals from the behaviors induced by abstract-level policy. Maximizing $\mathcal{I}(s; \omega)$ encourages different skills to induce different states that are further rendered to goals. This objective ensures that each skill individually is distinct and that the skills collectively explore large

parts of the state space (Eysenbach et al., 2018). As shown in Figure 10(c), our method provides better coverage of the state space than DISCERN and RIG.

Figure 10(d) shows the performance of the three methods, where we randomly sample goals from the whole state (or goal) space at test-time. 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), resulting in the agent learning to reach only a fraction of the state space, as shown in Figure 10(a) and 10(b).

## B Comparison to DIAYN Variants



(a) DIAYN-Imitator: $\mathcal{I}(s;\omega)$. (b) DIAYN-Goal: $\mathcal{I}(s;g)$. (c) GPIM: $\mathcal{I}(s;\omega)+\mathcal{I}(s;g)$

Figure 11: 2D navigation with (a) DIAYN-Imitator, (b) DIAYN-Goal, and (c) our proposed GPIM. In (a), (b) and (c-left), the open circles denote the user-specified goals, and the lines with a solid circle at the end are the corresponding behaviors. Specifically, the white lines in (a) are the learned skills in training, 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-left). In (c-right), we also show the generative factor $z$ by disentangling goals in (c-left), where the open circles with different colors in (c-right) correspond to the goals with corresponding colors in (c-left).

In DIAYN (Eysenbach et al., 2018), 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 $g$:

$$\hat{\omega} = \arg\max_{\omega} q_{\phi}(\omega|g). \tag{9}$$

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

$$\max \ \ \mathcal{I}(s;g), \tag{10}$$

where $g$ is sample from the prior goal distribution $p(g)$.

Figure 11 shows the comparison of our GPIM with DIAYN variants including DIAYN-Imitator and DIAYN-Goal, where 2D navigation task is considered. Given the limitation of DIAYN-Imitator that the state $s$ and user-specific goal $g$ must have the same representation, the state and goal in this experiment are set as coordinates.

As observed, DIAYN-Imitator can reach seen-goals but not for unseen-goals in Figure 11(a), because it cannot effectively accomplish the interpolation between skills that are induced in training. And behaviors generated by DIAYN-Goal cannot guarantee the consistency with the preset goals in Figure 11(b). The main reason is that this 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 $\omega$) have semantically consistent behavior information. Our proposed GPIM method, capable of solving interpolation and consistency issues, exhibits the best performance in this 2D navigation task.

## C Ablation Study

We further study the impact of disentanglement part in our framework. The performance is reported in Figure 12. It is observed that *w/o Pol* performs worse than all other methods, which is consistent

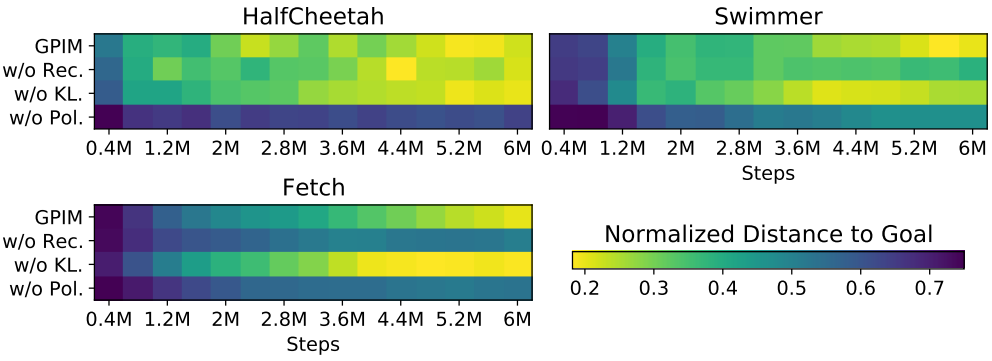



Figure 12: Ablation study on the state imitation task.

with the performance of 2D navigation task[4]. And we can find that when we remove the reconstruction loss ($\alpha = 0$), the performance of *w/o Rec* degrades in these three environments. The main reason is that the process of learning generative factor becomes 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 has better interpretability to behaviors as shown in Figure 11(c).

## D GENERALIZATION OF GPIM ON THE GRIDWORLD TASK

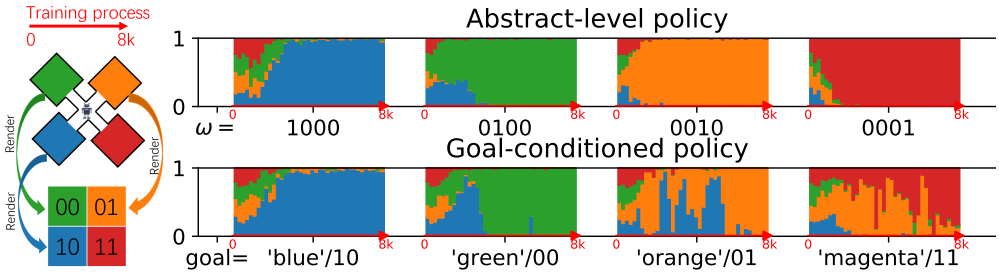



Figure 13: The Y-axis is the percentage of different rooms that robot arrives in.

Here we further test the generalization when there are missing dynamics or goal conditions on the gridworld task, where there are four rooms in different colors: blue, green, orange and magenta. We consider the case where we train the goal-conditioned policy in the blue room, the green room and the orange room, and let the magenta room serve as a test.

As shown in Figure 13, we quickly acquire four skills to different room through the training at the abstract level. After the robot reaches at the colored rooms, we rendered the corresponding room as a two-dimensional vector, Render('blue') = 10, Render('green') = 00, and Render('orange')=01; and we do not render the magenta room. We take the room corresponding to vector 01 as the test target to verify the generalization ability of the model.

In Figure 13 bottom, we show the learning process of the goal-conditioned policy. We can find that the blue room tasks are learned quickly, and the green and magenta rooms are learned relatively slowly, but the robot was still able to complete the test task successfully (the magenta room). Compared to the task in Figure 4, which renders all the rooms as goals, the whole learning process in Figure 13 is much slower.

## E LEARNED BEHAVIORS FROM GPIM

More experimental results are shown in Figure 14, including the visualization of learned goal-conditioned behaviors of each task. Videos are available under https://sites.google.com/view/gpim.

## F 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 robot to a specific coordinate named as *x-y goal* and moving the robot to a specific object with certain color and shape named as *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 and one block for manipulation, where another block exists as a distractor. The manipulation might arrive at the block and 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 is needed to move, and the 2D coordinate of the final location of the movement.

Table 1: The repetition length of the action.

| 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 state represents a 128-dimensional vector. Each action is repeatedly performed for a duration of k frames, where k is uniformly sampled from Table 1.

**Mujoco tasks:** We consider to make diverse agents to fast imitate a given goal trajectory, including the imitation of behaviors of a Swimmer, a HalfCheetah, and a Robot, where the states in the trajectory denote the 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 has more complicated physics than 2D navigation. State imitation considers moving a swimmer, a half-cheetah or a robot arm to a specified goal. The state is the configuration of the agent and the goal is an image.

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

[4]The results of 2D navigation task are shown in the full paper.

## G 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 at the final state of the trajectory induced by $\pi_\mu$ (Figures 6(a), 6(b) and 6(c)), the distance to goal is the L2-distance between the induced final state $\tilde{s}_T^k$ by $\pi_\theta(\cdot|\cdot, g^k)$ and the goal state $g^k$ which are randomly sampled from goal (task) space:

$$Dis = \frac{1}{50}\sum_{k=1}^{50} L2(\tilde{s}_T^k, g^k).$$

**II:** When the goal is to imitate the whole trajectory induced by $\pi_\mu$ (Figures 6(d), 6(e) and 6(f)), the distance is the expectation of distance over the whole trajectory $\{\tilde{s}_1^k, \tilde{s}_2^k, ..., \tilde{s}_T^k\}$ induced by $\pi_\theta(\cdot|\cdot, \{g_1^k, g_2^k, ..., g_3^k\})$ and goal trajectory $\{g_1^k, g_2^k, ..., g_3^k\}$ which are randomly sampled from the trajectory (task) space:

$$Dis = \frac{1}{50}\sum_{k=1}^{50}\left(\frac{1}{T}\sum_{t=1}^{T} L2(\tilde{s}_t^k, g_t^k)\right).$$

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[5] is the difference of the position of controllable agent, whose positions are obtained by matching pixel values on the imaged states. See the Python code below.

```
def obtainxy_seaquest(img):
  temp = np.where((img[:,:,0]==187)&(img[:,:,1]==187)&(img[:,:,2]==53))
  global seaquestxy
  if len(temp[0])==0:
    temp = seaquestxy
  else:
    seaquestxy = temp
  xy = np.array([np.mean(temp[0]), np.mean(temp[1])])/100
return xy

def obtainxy_montezuma_revenge(img):
  img[:20] = 0
  temp = np.where((img[:,:,0]==200)&(imgo[:,:,1]==72)&(img[:,:,2]==72))
  global montezumaxy
  if len(temp[0])==0:
    temp = montezumaxy
  else:
    montezumaxy = temp
  xy = np.array([np.mean(temp[0]), np.mean(temp[1])])/100
return xy

def obtainxy_berzerk(img):
  temp = np.array(img)
  temp[1:-1, 1:-1] = temp[:-2, :-2]/3+temp[2:,2:]/3+temp[1:-1,1:-1]/3
  temp = np.where((temp[:,:,0]==240)&(temp[:,:,1]==170)&(temp[:,:,2]==103
                                    ))
  global berzerkxy
  if len(temp[0])==0:
    temp = berzerkxy
  else:
    berzerkxy = temp
    xy = np.array([np.mean(temp[0]), np.mean(temp[1])])/100
  return xy
```

[5]The reward function for our baseline **L2 Distance** still calculates the L2-distance directly on the original state space, instead of the distance of the agent's position after pixel matching here.

In our implementation, we use two independent SAC architectures (Haarnoja et al., 2018) for abstract-level policy $\pi_\mu$ and goal-conditioned policy $\pi_\theta$. We found empirically that having two networks share a portion of the network structure will degrade the experimental performance. And we adopt universal value function approximates (UVFAs) (Schaul et al., 2015) for extra input (goals). For abstract-level policy $\pi_\mu$, to pass latent variable $\omega$ to the Q function, value function, and policy, as in DIAYN, we simply concatenate $\omega$ to the current state $s_t$ (and action $a_t$). For goal-conditioned policy $\pi_\theta$, we update the $\vartheta_E$ using the gradients from both the *Dis_loss* and Q function. And we also concatenate learned generative factor $z_t$ to current state $\tilde{s}_t$ (and action $\tilde{a}_t$).

The hyper-parameters are presented in Table 2.

## H Broader Impact

The main concern in the research area of goal-conditioned policy is how to find numerous diverse goals as well as obtain the reward function. 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 policy learning. Our model addresses these problems by allowing agents to interact with objects or environment, and learn the behaviors in a fully autonomous way on the basis of intrinsic motivations. Bridging with the render function between abstract level and goal-conditioned policy, we could obtain diverse and versatile policies. By optimizing the intrinsic reward, our GPIM makes an agent to have capability of solving diverse tasks, which fits for wide-range applications such as quick imitation on behavior states, interactive navigation and object manipulation and so on.

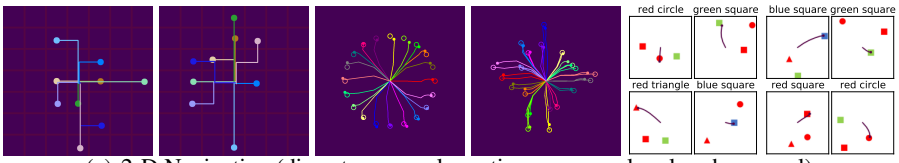
Moreover, our model provides an approach to address the Robot open-Ended Autonomous Learning (REAL) NeurIPS2019 (2019). And this framework is likely to increase progress of the general-purpose robots that can achieve complex tasks given the corresponding goals, and drive the development of autonomous robot learning in a life-long learning mode.

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 to guide the exploration or the offline reinforcement learning (Levine et al., 2020). While another issue comes from the exploration-exploitation trade-off. We would encourage further work to understand the limitations of REAL interacting with the environment autonomously. We would also encourage research to understand the risks arising from autonomous robot learning.

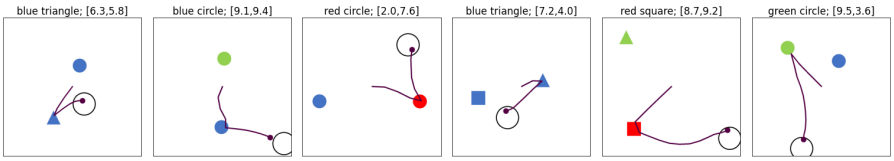
Another limitation is that our learning framework needs an extra abstract-level policy training for rendering goals and provides reward function. This requires twice as much interaction time with the environment as learning a single policy network. For practical application, a simulation platform is preferred for pre-training. We also encourage researchers to mitigate the difference between the simulation platform and the actual environment. And we also pursue the effective models of transfer learning.

Table 2: Hyper-parameters

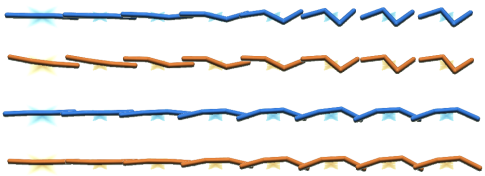
| Hyper-parameter | | | value |
|---|---|---|---|
| Batch Size | | | 256 |
| Discount Factor | | | 0.99 |
| Buffer Size | | | 10000 |
| Smooth coefficient | | | 0.05 |
| Temperature | | | 0.2 |
| Learning Rate | 2D Navigation | x-y goal | 0.001 |
| | | color-shape goal | 0.001 |
| | Object Manipulation | | 0.001 |
| | Mujoco tasks | Swimmer | 0.0001 |
| | | HalfCheetah | 0.0001 |
| | | Fetch | 0.0001 |
| | Atari games | Seaquest | 0.0003 |
| | | Berzerk | 0.0003 |
| | | Montezuma Revenge | 0.0003 |
| Path Length | 2D Navigation | x-y goal | 20 |
| | | color-shape goal | 20 |
| | Object Manipulation | | 20 |
| | Mujoco tasks | Swimmer | 50 |
| | | HalfCheetah | 50 |
| | | Robot | 100 |
| | Atari games | Seaquest | 25 |
| | | Berzerk | 25 |
| | | Montezuma Revenge | 25 |
| Hidden Size | 2D Navigation | x-y goal | 128 |
| | | color-shape goal | 128 |
| | Object Manipulation | | 128 |
| | Mujoco tasks | Swimmer | 256 |
| | | HalfCheetah | 256 |
| | | Robot | 256 |
| | Atari games | Seaquest | 256 |
| | | Berzerk | 256 |
| | | Montezuma Revenge | 256 |
| Dimension of Generactive Factor | 2D Navigation | x-y goal | 2 |
| | | color-shape goal | 2 |
| | Object Manipulation | | 4 |
| | Mujoco tasks | Swimmer | 3 |
| | | HalfCheetah | 4 |
| | | Robot | 4 |
| | Atari games | Seaquest | 16 |
| | | Berzerk | 16 |
| | | Montezuma Revenge | 16 |
| $\alpha$ | | | 1 |
| $\gamma$ | | | 5 |
| $\delta_{pixel}$ | | | 255 |



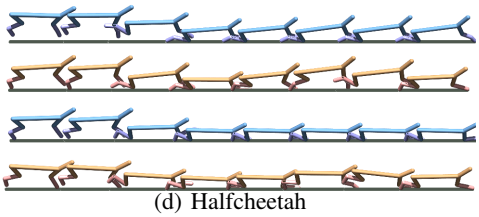

(a) 2-D Navigation (discrete x-y goal; continous x-y goal; color-shape goal)



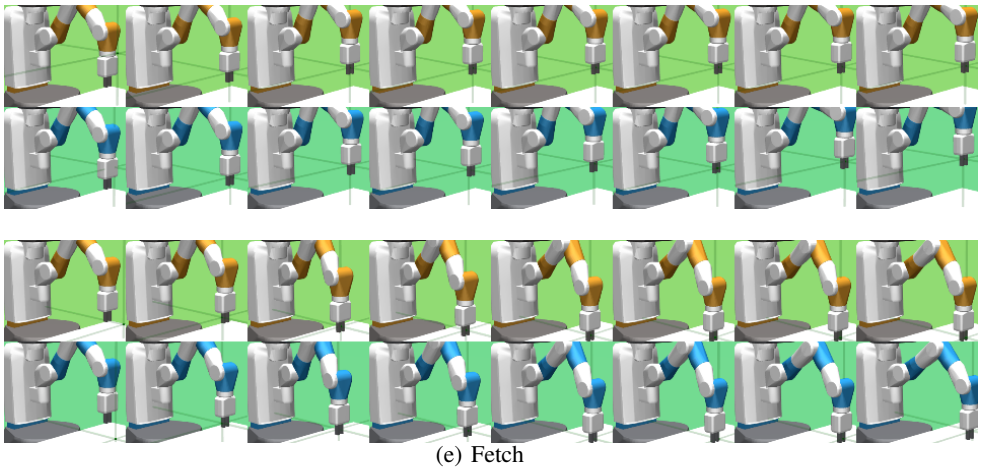

(b) Object Manipulation



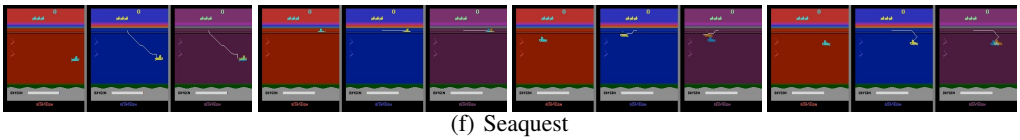
(c) Swimmer



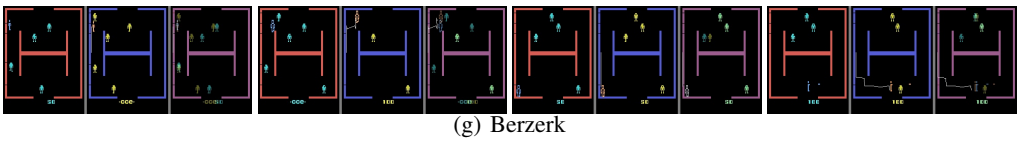
(d) Halfcheetah



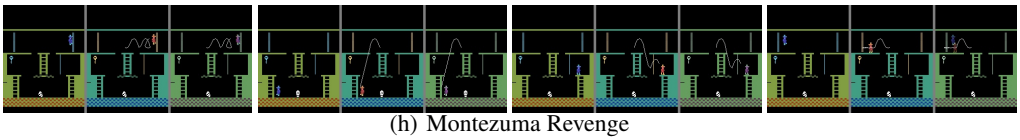
(e) Fetch



(f) Seaquest



(g) Berzerk



(h) Montezuma Revenge

Figure 14: Discovered goal-conditioned behaviors. (f-h): The left shows the goal behaviors; We shows the learned behaviors by GPIM in the middle; The right shows overlay display of goals and GPIM behaviors.