Compact Goal Representation Learning via Information Bottleneck in Goal-Conditioned Reinforcement Learning

Qiming Zou and Einoshin Suzuki

***Abstract*— We propose an Information bottleneck (IB) for Goal representation learning (InfoGoal), a self-supervised method for generalizable goal-conditioned reinforcement learning (RL). Goal-conditioned RL learns a policy from reward signals to predict actions for reaching desired goals. However, the policy would overfit the task-irrelevant information contained in the goal and may be falsely or ineffectively generalized to reach other goals. A goal representation containing sufficient task-relevant information and minimum task-irrelevant information is guaranteed to reduce generalization errors. However, in goal-conditioned RL, it is difficult to balance the tradeoff between task-relevant information and task-irrelevant information because of the sparse and delayed learning signals, i.e., reward signals, and the inevitable task-relevant information sacrifice caused by information compression. Our InfoGoal learns a minimum and sufficient goal representation with dense and immediate self-supervised learning signals. Meanwhile, InfoGoal adaptively adjusts the weight of information minimization to achieve maximum information compression with a reasonable sacrifice of task-relevant information. Consequently, InfoGoal enables policy to generate a targeted trajectory toward states where the desired goal can be found with high probability and broadly explores those states. We conduct experiments on both simulated and real-world tasks, and our method significantly outperforms baseline methods in terms of policy optimality and the success rate of reaching unseen test goals. Video demos are available at `infogoal.github.io`.**

***Index Terms*— Deep reinforcement learning (RL), goal-conditioned RL, Information bottleneck (IB), representation learning.**

I. Introduction

A GOAL-ORIENTED task, which requires an agent to reach desired goals, is ubiquitous not only in real-world applications, such as object manipulation [1], visual navigation [2], and object search [3], but also in policy learning paradigms, such as hierarchical reinforcement learning (RL) [4], skill learning [5], and imitation learning [6]. Goal-conditioned RL [7], [8], [9] maximizes an accumulated reward in a trajectory for learning diverse goal-reaching behaviors without making assumptions about the form of state and goal or the dynamic model of the environment. However, goal-conditioned RL can overfit task-irrelevant information contained in the goal input, which jeopardizes its ability to effectively generalize the learned behavior to reach new goals [10], [11], [12]. One potential way to circumvent the overfitting problem is to train the policy conditioned on a goal representation that is invariant to task-irrelevant information, i.e., a generalizable goal representation.

The generalization error of a learning algorithm is upper bounded by: 1) the mutual information (MI) between the input and the input representation and 2) the reciprocal of the number of learning samples [13]. Accordingly, there are two existing strategies for learning a generalizable goal representation: 1) reducing the MI between the goal and the goal representation [10] and 2) generating more learning samples with given training goals [1], [11], [12], [14], [15]. InfoBot [10] removes information contained in the goal representation while maximizing the cumulative reward for sufficient task-relevant information. However, in goal-conditioned RL, the reward signal is delayed, i.e., the reward is available at the end of each trajectory, and sparse, i.e., the reward is positive only when the desired goal is reached. Therefore, the information minimization term could easily overwhelm the cumulative reward maximization term, leading to a goal representation without sufficient task-relevant information. The policy taking the learned goal representation as input generates a goal-independent trajectory for the training and test goals, e.g., searching the entire environment to reach a desired goal. Consequently, InfoBot significantly improves the success rate of reaching test goals but loses the policy optimality. Self-supervised learning methods obtain "labels" from the data itself by using a "semiautomatic" process [16] and, thus, generate auxiliary learning samples. Self-supervised goal representation learning methods generate auxiliary learning samples with the given training goals via reconstruction tasks [1], [12], [14] or contrastive learning tasks [11], [15]. However, the generated auxiliary learning samples could fail to guarantee sufficient task-relevant information [1], [12], [14], [15] or be limited by the number of training goals, leading to the overfitting problem [11].

Motivated by the generalization error bound presented by Shamir et al. [13], we, for the first time, combine both the generation of learning samples and the compression of

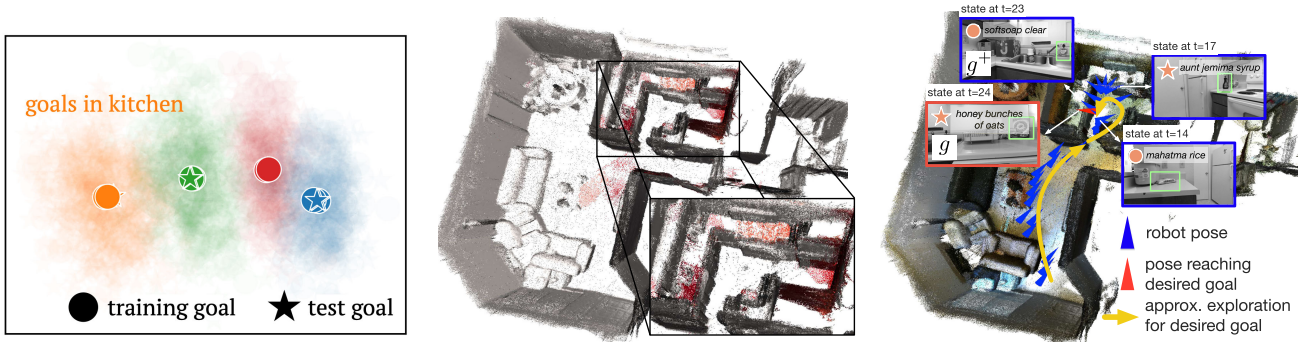
Manuscript received 10 June 2022; revised 14 January 2023, 1 August 2023, and 18 October 2023; accepted 10 December 2023. Date of publication 8 January 2024; date of current version 6 February 2025. This work was supported in part by the China Scholarship Council under Grant 202008050300 and in part by the Grant of the Graduate School of Information Science and Electrical Engineering (ISEE) of Kyushu University for Supporting Students' Participation in International Conferences. *(Corresponding author: Qiming Zou.)*

Qiming Zou is with the Graduate School of Systems Life Sciences, Kyushu University, Fukuoka 819-0395, Japan (e-mail: zou.qiming.847@s.kyushu-u.ac.jp).

Einoshin Suzuki is with the Graduate School and Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan (e-mail: suzuki@inf.kyushu-u.ac.jp).

This article has supplementary material provided by the authors available at https://doi.org/10.1109/TNNLS.2023.3344880.

Digital Object Identifier 10.1109/TNNLS.2023.3344880



(a) (b) (c)

Fig. 1. Motivation of InfoGoal. (a) Learned goal representation contains only task-relevant information, and thus, similar goals share similar representations. We model this goal representation using a Gaussian distribution, with the mean represented by a solid shape and representations sampled from the distribution represented by transparent shapes. (b) We define an intrinsic reward at a given state as the negative distance between the learned representation of the achieved goal at that state and the desired goal. The desired goal is located in the kitchen. The red region represents states that result in high intrinsic rewards, and the agent maximizing cumulative intrinsic reward would explore the red area in the kitchen. (c) In the trajectory visualization, we show both the achieved goals (denoted by circles) and the desired goal (denoted by a star) (targeted exploration for a test goal). Desired goal $\mathbf{g}$ and achieved goal $\mathbf{g}^+$ are reached within a time step. By combining the goal representation and intrinsic reward, the agent with a fully trained policy would be able to directly navigate to the kitchen and then broadly search for the test goal in the kitchen.

information through self-supervised Information bottleneck (IB) for Goal representation learning (InfoGoal) in RL. The difficulty is to balance a tradeoff between task-relevant and task-irrelevant information contained in the learned goal representation since maximizing information compression leads to an inevitable sacrifice of task-relevant information [17]. To overcome the difficulty, InfoGoal maximizes the MI between goals that were reached within a small time window by an arbitrary policy, which is theoretically proven to be sufficient to learn a near-optimal policy for training goals. In addition, we derive a lower bound for the MI, which allows for adaptively updating the weight of information compression, i.e., the Lagrange multiplier. Consequently, InfoGoal maximizes information compression with reasonable task-relevant information sacrifice. As depicted in Fig. 1(a), with our method, goals reached by similar optimal policies are grouped in the same cluster in the latent space. Besides, we derive an intrinsic reward at a given state as the negative representation distance between the desired and achieved goals, as shown in Fig. 1(b). The reward function encourages the agent to reach goals that are similar or identical to the desired goal. As shown in Fig. 1(c), the learned goal representation and the intrinsic reward function jointly encourage the policy to generate a targeted trajectory to states where the desired goal can be found in a high probability and searches broadly around the states.

In summary, we have made three major contributions to this work.

1) We introduce a self-supervised IB that guarantees sufficient task-relevant information with immediate and dense learning signals. In addition, we derive a lower bound for MI, enabling an adaptive Lagrange multiplier adjustment. Both components balance the tradeoff between task-relevant information and task-irrelevant information in the learned goal representation.
2) We present an intrinsic reward function based on the learned goal representation that promotes targeted goal-searching behavior.
3) We demonstrate the effectiveness of our method in various simulated and real-world goal-conditioned RL tasks, showing significant improvement in the policy optimality and success rate of reaching new test goals.

II. Related Works

Our InfoGoal generates immediate and dense learning signals for minimal and sufficient goal representation. In this section, we introduce works which also generate denser learning signals for goal-conditioned RL via hindsight experience replay (HER) [18], [19], [20], IB-based state representation learning [21], [22], [23], [24], [25], [26], [27], [28], and goal representation learning [1], [8], [10], [11], [12], [14], [15], [18].

HER-based goal-conditioned RL method [18], [19], [20] generates denser reward signals from collected trajectories as our method. The HER-based method replaces the desired goal with a state, i.e., pseudogoal, which is included in the collected trajectories, even though the desired goal is not reached. Relabeling a trajectory that failed to reach a desired training goal with a test goal can help the agent learn how to reach the test goal. However, the reward signals are still delayed. Moreover, with a limited number of goals, the learned policy would overfit the goals that have been seen and fail to generalize to a test goal unseen during the training phase, e.g., a goal to answer a new question about a seen object in an embodied question-answering task. Instead of using the delayed reward signals, our method learns a minimum and sufficient goal representation with immediate learning signals to improve the generalization ability of the policy.

IB-based state representation learning method minimizes MI between a state and its representation (or a state sequence and a state representation sequence) while maximizing the task-relevant information. These methods guarantee sufficient task-relevant information by maximizing the cumulative reward [21], [22], [23] or MI between adjacent states [24], [25], [26], [27], [28]. However, in a goal-conditioned task, which information is relevant or irrelevant depends not only on the current state but also on the goal [11]. Therefore, our work learns a compact goal representation, which is complementary to state representation learning.

Goal representation learning, which is the focus of this work, is proposed to extract task-relevant information contained in the goal input for training sample efficient or generalizable goal-conditioned RL. A typical goal-conditioned RL algorithm updates goal representations with reward signals in an end-to-end fashion [8], [18]. However, positive rewards

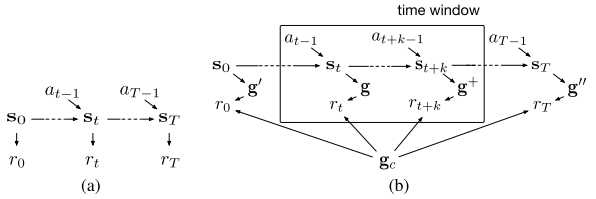


Fig. 2. Graphical models for the MDP of (a) RL and (b) MDP of goal-conditioned RL. A time window is a period with a fixed length and can be slid along a trajectory.

are often delayed and sparse, and thus, even a large number of trajectories might not provide meaningful learning signals. An autoencoding model learns a goal representation with an auxiliary goal reconstruction task in which the representation and a decoder network are optimized jointly to reconstruct the goal [1], [14]. A recent study [12] empirically demonstrated that using a discrete goal representation can improve the generalization ability of an RL agent for tasks with image goals, as the discretization helps similar goal groups. This approach is effective when a single goal contains sufficient task-relevant information that differentiates it from other irrelevant goals, e.g., image goals. Contrastive goal grouping (COGOAL) [15] considers goals that are reached adjacently as positive pairs and clusters these pairs in the latent space. Goal-conditioned bisimulation (GCB) [11] regards the initial state and goal image in a task as a positive pair and learns a goal-state encoder that maps similar pairs to the same cluster in the latent space. Both methods utilize the dynamic information of the environment to cluster similar goals in the latent space. However, these methods do not explicitly remove task-irrelevant information as our method does, which may lead to an overfitting problem, especially with limited training tasks.

IB-based goal representation learning, which is most related to our work, is much less explored. As far as we know, InfoBot [10] is the only previous work that extends the IB principle to goal representation learning in RL. InfoBot [10] learns a goal representation with minimum goal-specific information. The policy taking the learned goal representation as input tends to perform a goal-independent behavior for reaching a new goal, e.g., looking for seasoning in the entire indoor environment. Our InfoGoal learns a goal representation containing minimum and sufficient task-relevant information, leading to a trajectory targeting the desired goal while still allowing for broad searching around states that are promising for finding the desired goal, e.g., looking for seasoning in the kitchen.

III. Background

A. Reinforcement Learning

RL is typically formalized as a Markov decision process (MDP) [29]. A graphical model formulation of MDP is illustrated in Fig. 2(a). An MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $P: \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes the state transition function. The agent is initialized to initial state $\mathbf{s}_0$. At time step t, the agent at state $\mathbf{s}_t \in \mathcal{S}$ executes action $a_t \in \mathcal{A}$, and then, the state transits to $\mathbf{s}_{t+1} \in \mathcal{S}$ with probability $P(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$. Reward r_t is drawn from reward distribution $R: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Interaction record $\tau = \{(\mathbf{s}_t, a_t, r_t)\}_{t=0,1,\ldots,T^\tau-1}$ is called a trajectory, where T^τ is the length of τ. The discount factor γ penalizes future rewards in order to encourage the agent to maximize cumulative rewards as efficiently as possible.

The objective is to find optimal policy π^* that maximizes its expected discounted sum of rewards $V^\pi(\mathbf{s})$, in one episode. Q-learning [30] is a commonly used method to achieve this objective when the action space is discrete. In Q-learning, the value of a state is defined as a conditional expectation[1] of the discounted sum of rewards given π and $\mathbf{s}_0$ over τ, i.e.,

$$V^\pi(\mathbf{s}) = \mathbb{E}_\tau\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \pi, \mathbf{s}_0 = \mathbf{s}\right] \tag{1}$$

where γ is a discount factor ($0 < \gamma < 1$). Similarly, action-value (or Q value) function $Q^\pi: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is

$$Q^\pi(\mathbf{s}, a) = \mathbb{E}_\tau\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \pi, \mathbf{s}_0 = \mathbf{s}, a_0 = a\right]. \tag{2}$$

The optimal Q value function is defined as

$$Q^*(\mathbf{s}, a) = \arg\max_{\pi \in \Pi} Q^\pi(\mathbf{s}, a) \quad \forall \mathbf{s} \in \mathcal{S}, \quad a \in \mathcal{A} \tag{3}$$

where Π is a set of policies over $\mathcal{S}$ and $\mathcal{A}$. In Q-learning, given $Q^*(\mathbf{s}, a)$, it is proven that π^* can be obtained by choosing actions greedily [30]

$$\pi^*(\mathbf{s}) = \arg\max_{a \in \mathcal{A}} Q^*(\mathbf{s}, a) \quad \forall \mathbf{s} \in \mathcal{S}. \tag{4}$$

The deep Q network (DQN) [31] directly estimates $Q^*(\mathbf{s}, a)$ with a deep neural network and obtains π^* with (4).

B. Goal-Conditioned RL

Goal-conditioned RL is a special case of RL, which considers a problem of reaching goal $\mathbf{g}$ drawn from goal space $\mathcal{G}$. We illustrate a graphical model formulation for the MDP of goal-conditioned RL in Fig. 2(b).

In each episode, the agent learns to reach the desired goal $\mathbf{g}_c$. The common goal-conditioned RL setting assumes that the goal space is a subset of the state space, but we set a more general assumption that there is a known mapping from state to goal, i.e., $\mathbf{g} = m(\mathbf{s})$. For example, in the visual object search task, the agent can recognize the objects in images. In this example, mapping function m could be a pretrained object detection network, e.g., YOLO [32].

The reward function is defined as

$$r(\mathbf{s}) := \begin{cases} 1, & \text{if } \mathbf{g}_c = m(\mathbf{s}) \\ 0, & \text{else.} \end{cases} \tag{5}$$

We define $T^\tau_{\mathbf{g}_c}$ as the number of time steps to reach $\mathbf{g}_c$ in trajectory τ, i.e., the first hit time of goal $\mathbf{g}_c$ in trajectory τ. An episode is terminated when $\mathbf{g}_c$ is reached, i.e., $T^\tau = T^\tau_{\mathbf{g}_c}$. A universal value function [8] approximates different goal-conditioned value functions or action-value functions with a single model, e.g., the deep neural network

$$\hat{V}^\pi(\mathbf{s}, \mathbf{g}_c) = \mathbb{E}_\tau\left[\gamma^{T^\tau_{\mathbf{g}_c}} \middle| \pi, \mathbf{s}_0 = \mathbf{s}\right] \tag{6}$$

$$\hat{Q}^\pi(\mathbf{s}, a, \mathbf{g}_c) = \mathbb{E}_\tau\left[\gamma^{T^\tau_{\mathbf{g}_c}} \middle| \pi, \mathbf{s}_0 = \mathbf{s}, a_0 = a\right]. \tag{7}$$

The state and the goal are encoded by two separate neural networks $\psi: \mathcal{S} \to \mathbb{R}^n$ and $\phi: \mathcal{G} \to \mathbb{R}^n$, where n is the dimension of the state representation and the goal representation, respectively. The goal representation and the state representation are concatenated into vector $\psi(\mathbf{s}) \oplus \phi(\mathbf{g})$, where $\oplus$ denotes the concatenation operator. The fully connected network h maps $\psi(\mathbf{s}) \oplus \phi(\mathbf{g})$ to an action probability distribution.

[1]The conditional expectation is denoted as $\mathbb{E}_Z[X|Y]$ that represents the expected value of X given Y over Z.

C. Information Bottleneck

The IB was introduced in [33] as an information-theoretic framework for learning. It considers extracting information about a target signal (label) $\mathbf{y}$ through a correlated observable input $\mathbf{x}$. The extracted information is quantified by variable (representation) $\mathbf{z}$, defined by parametric encoder $p(\mathbf{z}|\mathbf{x};\theta)$, thus forming Markov chain $\mathbf{y} \leftrightarrow \mathbf{x} \leftrightarrow \mathbf{z}$. The objective is to find $\mathbf{z}$ that minimizes $I(\mathbf{x};\mathbf{z})$, while maximizing $I(\mathbf{y};\mathbf{z})$. Specifically, the IB problem for a pair $(\mathbf{x},\mathbf{y})$ is given as

$$\min_{\theta} \beta I(\mathbf{z};\mathbf{x}) - I(\mathbf{z};\mathbf{y}) \tag{8}$$

with the introduction of Lagrange multiplier β. This formulation provides a natural approximate version of minimal sufficient statistics [34]. The IB principle is appealing since it defines what we mean by a good representation, in terms of the tradeoff between having a concise representation and a representation with good predictive power [35].

IV. Information Bottleneck for Goal Representation

In this section, we first introduce the concept of sufficient goal representation. Next, we propose an information maximization objective function that not only generates dense learning signals but also ensures that the learned representation is sufficient. To address the challenge of balancing the infomax and infomin terms in the IB loss using a Lagrange multiplier, we suggest a lower bound for MI, which can be utilized in an adaptive Lagrange multiplier adjustment strategy. Finally, we utilize the learned goal representation to derive an intrinsic reward function, which encourages the agent to explore states where the achieved goal is close to the desired goal in the latent space.

A. Sufficient Goal Representation

A sufficient goal representation guarantees that the optimal policy in the learned latent goal space is the same as the optimal policy in the original goal space, i.e., the optimality of the learned policy [36]. We formalize this intuition by: 1) defining the goal representation as a stochastic function that takes a goal as input and outputs a probability distribution over a vectored goal representation; 2) giving a specific condition that the goal representation should fulfill (termed the *sufficiency* condition); and 3) proving that a goal representation that meets the condition guarantees the optimality of the learned policy.

We quantify the mutual dependence with MI, which is defined over random variables, and thus, we formalize a stochastic representation ϕ as a mapping from $\mathbf{g}$ to a probability distribution $p(\mathbf{z}|\mathbf{g})$. Specifically, ϕ takes goal $\mathbf{g}$ as input and outputs $(\boldsymbol{\mu},\boldsymbol{\sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ represent the mean and the standard deviation of a normal distribution, and then, goal representation $\mathbf{z}$ is sampled from normal distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma})$. In order to update ϕ end-to-end, we utilize the reparameterization trick [37], which approximates sampling $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma})$ with the following equation:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \tag{9}$$

where $\boldsymbol{\eta}$ is a noise sampled from a unit normal distribution and $\odot$ denotes the elementwise multiplication.

Intuitively, if two goal representations are the same, the corresponding goals should share the same optimal Q-function of any state-action pairs, which is formalized in the following definition.

Definition 1 (Sufficiency): Goal representation $\phi(\mathbf{g})$ is sufficient if $\phi(\mathbf{g}) = \phi(\mathbf{g}^+) \implies \forall \mathbf{s}, a, \quad Q^*(\mathbf{s},a,\mathbf{g}) = Q^*(\mathbf{s},a,\mathbf{g}^+)$.

In the following theorem, we prove that the goal representation that meets the condition defined in Definition 1 is sufficient for an RL agent to learn the same optimal policy in the original goal space.

Theorem 1: If goal representation ϕ is sufficient, i.e., $\forall \mathbf{g},\mathbf{g}^+,\ \phi(\mathbf{g}) = \phi(\mathbf{g}^+) \implies \forall \mathbf{s},a,\ Q^*(\mathbf{s},a,\mathbf{g}) = Q^*(\mathbf{s},a,\mathbf{g}^+)$, then ϕ guarantees the convergence of goal-conditioned Q-learning to the optimal policy in the original goal space.

Proof: See Appendix A in the Supplementary Material. □

Therefore, we aim to find a representation meeting the condition defined in Definition 1.

B. Information Bottleneck Loss

In this section, we first bridge the connection between learning sufficient goal representation and MI maximization. Then, we propose an MI maximization objective function, which does not require the optimal policy.

We prove that maximizing $I(\phi(\mathbf{g}),\mathbf{s_t},a_t;\phi(\mathbf{g}^+),\mathbf{s_{t+k}})$ $(\mathbf{g} = m(\mathbf{s_t}),\mathbf{g}^+ = m(\mathbf{s_{t+k}}))$ guarantees the sufficiency of a goal representation, where k is smaller than w, which is the size of a small time window. A time window is a period with a fixed length and can be slid along a trajectory, as shown in Fig. 2. For clarity, we introduce the following denotations:

$$\mathbf{x} := (\mathbf{s_t},a_t,\mathbf{g}), \quad \mathbf{x}^+ := (\mathbf{s_{t+k}},\mathbf{g}^+)$$
$$\mathbf{z} := (\mathbf{s_t},a_t,\phi(\mathbf{g})), \quad \mathbf{z}^+ := (\mathbf{s_{t+k}},\phi(\mathbf{g}^+)).$$

Theorem 2: If $I(\mathbf{z},\mathbf{z}^+)$ is maximized, then ϕ is sufficient.

Proof: See Appendix B in the Supplementary Material. □

Based on Theorem 2, we propose a new form of the IB loss function given as

$$\mathcal{L}_{\mathrm{IB}} = \beta \overbrace{I(\phi(\mathbf{g});\mathbf{g})}^{\text{infomin}} - \overbrace{I\left(\mathbf{z};\mathbf{z}^+\right)}^{\text{infomax}} \tag{10}$$

where the optimization variable is the parameter of the goal encoder ϕ. We refer to the MI minimization term as *infomin* and the MI maximization term as *infomax*. Compared to the original IB loss function in (8), our IB loss function guarantees the sufficiency of the goal representation, which alleviates the need for labels.

C. MI Estimation and Adaptive Lagrange Multiplier Adjustment

In this section, we first derive a lower bound for MI between $\mathbf{z}$ and $\mathbf{z}^+$ in loss function (10). Based on the derived lower bound, we propose an adaptive adjustment strategy for Lagrange multiplier β in loss function (10).

1) MI Estimation: The calculation of MI between high dimensional continuous random variables is difficult, and thus, we adopt approximations for both infomax and infomin terms in (10). For the infomin term, we adopt an upper bound derived in [38], i.e.,

$$I(\phi(\mathbf{g});\mathbf{g}) \le \mathrm{KL}(\phi(\mathbf{g}),\mathcal{N}(\mathbf{0},\mathbf{I})). \tag{11}$$

For the infomax term, we derive a lower bound which is beneficial for an adaptive Lagrange multiplier adjustment.

Theorem 3: Given random variable $\mathbf{x_0}$, $\mathbf{x_1}$, $\mathbf{x}_{\text{neg}} \in \mathcal{X}$, and assuming that $\mathbf{x}_{\text{neg}}$ is independent of $\mathbf{x_0}$, we have

$$I(\mathbf{x_0}; \mathbf{x_1}) \geq \underbrace{\mathbb{E}_{\mathcal{X}}\left[\log \frac{f(\mathbf{x_0}, \mathbf{x_1})}{\sum_{\mathbf{x}_{\text{neg}} \in \mathcal{X}} f\left(\mathbf{x_0}, \mathbf{x}_{\text{neg}}\right)}\right]}_{\text{our lower bound}} \tag{12}$$

where f is an auxiliary score function jointly optimized during the training process.

Proof: See Appendix C in the Supplementary Material. □

For our infomax term, $\mathbf{x_0} := \mathbf{z}$, $\mathbf{x_1} := \mathbf{z}^+$, and negative sample $\mathbf{x}_{\text{neg}}$ is defined as $\mathbf{z}' := (\mathbf{s_{t+k}}, m(\mathbf{s_{t+k}})), \forall \tau, k > w$. For sampling $\mathbf{z}'$, we define set $\mathcal{G}^{\mathbf{g}}_{\text{neg}}$, which contains goals that are dissimilar to $\mathbf{g}$. Note that, in this context, "dissimilar goals" refer to goals sampled from $\mathcal{G}$ that cannot be reached within a small time window in any trajectory. We initialize $\mathcal{G}^{\mathbf{g}}_{\text{neg}}$ as $\mathcal{G}$ and progressively remove similar goals. As the trajectory generation progresses, goals within $\mathcal{G}^{\mathbf{g}}_{\text{neg}}$ become dissimilar to $\mathbf{g}$. We obtain negative samples via $\mathbf{z}' := (\mathbf{s_{t+k}}, m(\mathbf{s_{t+k}})), m(\mathbf{s_{t+k}}) \in \mathcal{G}^{\mathbf{g}}_{\text{neg}}$. We focus on tasks with discrete goal spaces, i.e., $\mathcal{G}$ is a set of goals. For a continuous goal space, the size of $\mathcal{G}^{\mathbf{g}}_{\text{neg}}$ would be infinite. We can train a classifier that takes the desired goal and another goal as input and predicts whether these two goals are similar. We can then sample a goal from the replay buffer, which is predicted to be dissimilar to $\mathbf{g}$. Compared to the random sampling [39], this method can successfully sample dissimilar goals with higher probabilities.

Finally, we put everything together to obtain the following objective function, which we try to minimize:

$$\hat{\mathcal{L}}_{\text{IB}}\left(\phi, \mathbf{s}, \mathbf{s}^+\right) = \mathbb{E}_\tau\left[\log \frac{f\left(\mathbf{z}, \mathbf{z}^+\right)}{\sum_{\substack{\mathbf{z}' := (\mathbf{s}', m(\mathbf{s}')), \\ m(\mathbf{s}') \in \mathcal{G}^{\mathbf{g}}_{\text{neg}}}} f(\mathbf{z}, \mathbf{z}')} + \beta \text{KL}(\phi(\mathbf{z}), \mathcal{N}(\mathbf{0}, \mathbf{I}))\right] \tag{13}$$

where $\mathbf{z} := (\mathbf{s}, a, \phi(m(\mathbf{s})))$, $\mathbf{z}^+ := (\mathbf{s}^+, a^+, \phi(m(\mathbf{s}^+)))$; $\mathbf{s}$ and $\mathbf{s}^+$ are adjacent states.

2) Adaptive Lagrange Multiplier Adjustment: During training, the Lagrange multiplier should be increased if the goal representation contains sufficient task-relevant information. Otherwise, the Lagrange multiplier should be decreased. Therefore, the Lagrange multiplier is updated at each iteration of goal representation learning.

Based on (13), we can quantify the task-relevant information with the following metric:

$$\sigma(\mathbf{z}) = \mathbb{E}_\tau \mathbb{1}\left(f\left(\mathbf{z}, \mathbf{z}^+\right) > f\left(\mathbf{z}, \mathbf{z}'\right)\right) \tag{14}$$

where indicator function $\mathbb{1}$ outputs 1 if the condition is satisfied, otherwise outputs 0; $f(\cdot)$ is optimized with (13) for assigning a high value to pair $(\mathbf{z}, \mathbf{z}^+)$ and a low value to pair $(\mathbf{z}, \mathbf{z}')$, i.e., discriminating similar and dissimilar goals. Since the output of $\mathbb{1}(\cdot)$ is either 0 or 1, we have $\sigma(\mathbf{z}) \in [0, 1]$. Intuitively speaking, $\sigma(\mathbf{z})$ is the accuracy of discriminating similar and dissimilar goals based on information contained in $\mathbf{z}$.

The value of β is adaptively updated during the goal representation learning process with the following equation:

$$\beta = \beta + \rho(\sigma(\mathbf{z}) - \alpha) \tag{15}$$

where ρ is a learning rate, which is usually a small number, and α is a threshold close to 1. Intuitively speaking, we increase β when $\sigma(\mathbf{z})$ is higher than α. Otherwise, we decrease β. In other words, we remove more information when $\mathbf{z}$ contains sufficient information for discriminating similar and dissimilar goals. Compared with β that needs to tradeoff between infomin and infomax terms, our setting of ρ and α is easy and straightforward.

D. Local Searching With Intrinsic Reward

When there is no adequate training goal, the agent cannot learn an efficient searching strategy from a sparse reward signal. In this section, we introduce an intrinsic reward function that encourages the agent to locally search state regions where the desired goal can be found in a high probability.

The intrinsic reward function is defined as a negative distance between the learned goal representations via the InfoGoal loss given in (13)

$$r_{\text{in}}(\mathbf{s_t}, \mathbf{g}) = d(\phi(\mathbf{g}), \phi(m(\mathbf{s_t}))) \tag{16}$$

where $d(\cdot)$ is the Euclidean distance function. Intuitively, this reward function encourages the agent to explore states where the achieved goals share similar task-relevant information with the desired goal.

E. Practical Algorithm

The complete algorithm alternates between collecting new trajectories, minimizing InfoGoal loss, and goal-conditioned RL, as shown in Algorithm 1.

We denote the goal encoder with ϕ, the state encoder with ψ, and the policy with π. Buffer $\mathcal{G}^{\mathbf{g}}_{\text{neg}}$ restores goals that are dissimilar to $\mathbf{g}$ and initialized with goal space $\mathcal{G}$. At the start of each episode, desired goal $\mathbf{g}_c$ and initial state $\mathbf{s}_0$ are sampled from $\mathcal{G}$ and $\mathcal{S}$, respectively (lines 3 and 4). In each episode, policy π interacts with the environment to collect transitions $\tau = \{(\mathbf{s}_t, a_t, r(\mathbf{s}_t), \mathbf{s}_{t+1}, \mathbf{g}_c)\}_{t=1,\ldots,T}$ (lines 5–12). Note that policy π outputs action a_t conditioned on state representation $\psi(\mathbf{s}_t)$ and goal representation $\psi(\mathbf{g}_t)$, instead of the raw state and goal inputs.

For goal representation learning, we construct positive state pair $(\mathbf{s}, \mathbf{s}^+)$ (lines 16 and 17) and get positive goal pair $(\mathbf{g}, \mathbf{g}^+)$ via $\mathbf{g} = m(\mathbf{s})$ and $\mathbf{g}^+ = m(\mathbf{s}^+)$ (line 18). These goals are reached by π within a small time window, i.e., $|t^+ - t| \leq w$. We remove $\mathbf{g}^+$ from $\mathcal{G}^{\mathbf{g}}_{\text{neg}}$ since $\mathbf{g}$ and $\mathbf{g}^+$ can be reached within a small time window (line 19). As the trajectory generation progresses, $\mathcal{G}^{\mathbf{g}}_{\text{neg}}$ contains only goals dissimilar to goal $\mathbf{g}$. We sample negative goal pair $(\mathbf{g}, \mathbf{g}')$, where $\mathbf{g}' \in \mathcal{G}^{\mathbf{g}}_{\text{neg}}$ (line 20). We update goal encoder ϕ via minimizing the loss function in (13) (line 21). Moreover, based on the accuracy of the MI estimator to classify the positive and negative pairs, we update the Lagrange multiplier β for balancing the infomax and infomin terms (line 21). Finally, we update goal encoder ϕ, state encoder ψ, and policy π jointly with an off-the-shelf RL algorithm (line 22). Note that this algorithm is complementary to other state representation learning methods, e.g., self-supervised exploration with DB-bonus (SSE-DB) [24], and RL methods, e.g., DQN [8], HER [18].

V. Experiments

In this section, we aim to answer three questions.

1) How does the goal influence the learning performance of an RL method?
2) Does our method improve the generalizability of an RL agent?
3) What are the gains from our method?

Algorithm 1 Goal-Conditioned RL With InfoGoal

Input : Goal encoder ϕ, state encoder ψ, policy π, environment $\mathcal{M}$, set of dissimilar goals of the desired goal $\mathcal{G}_{\text{neg}}^{\mathbf{g}_c}$ (initialized with $\mathcal{G}$), empty replay buffer $\mathcal{B}$, time window size w

Output: Fully trained ϕ, ψ and π

for *epoch = 1 to # of training epochs* **do**
  /* Collect new trajectories */
  for *episode = 1 to # of episodes per epoch* **do**
    Uniformly sample goal $\mathbf{g}_c \in \mathcal{G}$.
    Reset the agent to random initial state $\mathbf{s}_0 \in \mathcal{S}$.
    for *t = 1 to max. time steps per episode* **do**
      Output action $a_t = \pi(\psi(\mathbf{s}_t), \phi(\mathbf{g}_c))$
      Update state $\mathbf{s}_{t+1} \sim P(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$
      Calculate reward $r(\mathbf{s}_t)$ with Eq. (5).
      if $\mathbf{g_c} = m(\mathbf{s}_{t+1})$ **then**
        Break.
      end if
    end for
  end for
  Adding $\tau = \{(\mathbf{s}_t, a_t, r(\mathbf{s}_t), \mathbf{s}_{t+1}, \mathbf{g}_c)\}$ to $\mathcal{B}$.
  for *iteration = 1 to # of updates per epoch* **do**
    /* InfoGoal */
    Sample $(\mathbf{s}_t, \mathbf{s}_{t^+})$, $|t^+ - t| \leq w$, from $\mathcal{B}$.
    Construct positive state pair $(\mathbf{s}, \mathbf{s}^+)$ via $\mathbf{s} = \mathbf{s}_t$ and $\mathbf{s}^+ = \mathbf{s}_{t^+}$.
    Construct positive goal pair $(\mathbf{g}, \mathbf{g}^+)$ via $\mathbf{g} = m(\mathbf{s})$ and $\mathbf{g}^+ = m(\mathbf{s}^+)$.
    Remove $\mathbf{g}^+$ from $\mathcal{G}_{\text{neg}}^{\mathbf{g}}$.
    Construct negative pair $(\mathbf{g}, \mathbf{g}')$, $\mathbf{g}' \in \mathcal{G}_{\text{neg}}^{\mathbf{g}}$.
    Update ϕ, β via Eq. (13), Eq. (15), respectively.
    /* Goal-conditioned RL */
    Update ϕ, π, and ψ end-to-end with an off-the-shelf RL algorithm.
  end for
end for

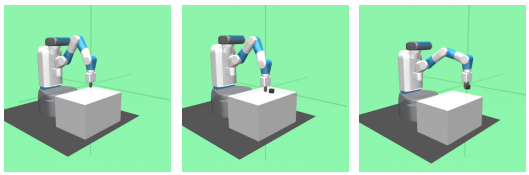


Fig. 3. Fetch [40] environments: FetchReach, FetchPush, and FetchPickAndPlace.

A. Benchmark Environments

In an environment, the RL agent is trained to reach test goals starting from an initial state. The overview of the environment settings is reported in Appendix D in the Supplementary Material. The detailed environment information is given in the following.

1) Fetch [40]: The fetch environments are based on a 7-DoF robotic arm that has a two-fingered parallel gripper. We adopt three tasks as benchmarks (see Fig. 3).

1) *FetchReach:* The task is to move the gripper to a target position.

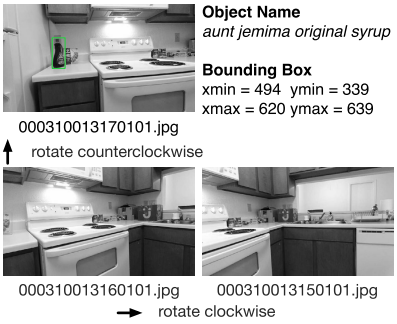


Fig. 4. Object search environment. Example of simulating an embodied movement with discretely captured images. An image is labeled with detected object names and the corresponding bounding boxes.

2) *FetchPush:* A box is placed on a table in front of the robot, and the task is to move it to a target location on the table.
3) *FetchPickAndPlace:* The task is to grasp a box and move it to the target location, which is located either on the table surface or in the air above it.

The observation includes the state of the robot and an object, i.e., the end-effector position and the object position. The goal in the original environments from [40] is 3-D and describes the desired position of the object (or the end-effector). The condition of reaching a goal is that the Euclidean distance between the object (or the end-effector) and the desired position is smaller than 5 cm. To evaluate the importance of the goal representation, we replace the original goal with the corresponding image goal. The image goal is an RGB image captured when the object (or the end-effector) is at the desired position.

In real-world applications, environments usually evolve spatially and temporally, requiring policies to be robust to shifts in the goal space. In the case of a visual goal input, shifting is often reflected in changes to foreground textures and colors, and the background of the goal input. For example, in the fetch environment, the goal inputs may be generated by humans manually controlling the robot arm. However, the robot arm may be located in a dynamic environment such as a factory where people are constantly working in the background or obscuring the camera. Meanwhile, the lighting may keep changing. Consequently, the foreground and background of the goal input change frequently. To simulate the various levels of noise that are widespread in real-world environments and evaluate the generalization ability of our method, we corrupt the training and test goal images by overlaying images randomly sampled from the CIFAR10 dataset. Specifically, goal image $\mathbf{x}$ is corrupted by sampled background image $\mathbf{x}_{\text{bkg}}$ following $\mathbf{x_c} = (1 - w_{\text{bkg}})\mathbf{x} + w_{\text{bkg}}\mathbf{x}_{\text{bkg}}$, where w_{bkg} controls the degree of corruption and $\mathbf{x_c}$ is the corrupted goal image. This is a common practice in RL research [41].

2) Object Search [42]: This task contains an automobile-wheeled robot that is commanded to search several objects in an indoor environment with a real first-person image. We simulate the embodied motion by converting the Active Vision Dataset [42] into a connectivity graph with discrete points and moving along the edges of the graph (similar to [43]). For example, as shown in Fig. 4, two images are connected by a directed edge with an action type attribution.

At each time step, the robot receives a gray image as an observation. In addition to the observation, a target object (randomly sampled from 24 object categories in each episode)

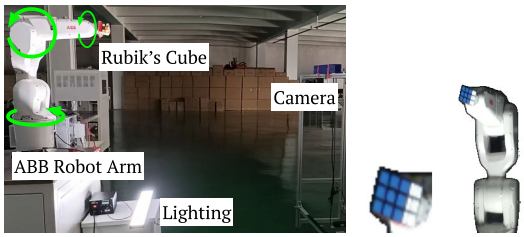


Fig. 5. Object manipulation environment. Left: hardware setting for the dataset collection. Right: example of a state.

is given to the robot in the text. The text is the name of the target object, e.g., "Coca-Cola glass bottle." SBERT [44] transforms the text into a 384-D vector, which is then fed into the goal encoder for further processing. The robot reaches the goal if the target object can be detected in the observation. Note that each image in the Active Vision Dataset [42] is labeled with detected object names and the corresponding bounding boxes (as shown in Fig. 4). As a result, there is no need for an object detector. Besides, we choose the bounding box of an object so that the robot has a clear view of the target object. The environment layout, an example of a state, a desired goal, an achieved goal, and a trajectory are shown in the right plot of Fig. 1.

3) Object Manipulation: We collect a dataset based on a 6-DoF ABB robot arm of which the version is IRB 1200. This dataset contains RGB images in which the robot arm rotates a 3×3 Rubik's cube around the pitch, yaw, and roll axes, respectively. In the task, the robot arm is commanded to rotate the Rubik's cube to a desired pose given in the RGB image.

Fig. 5 (left) shows the hardware setting with which we collect the dataset. The Rubik's cube is fixed to the end-effector of the robotic arm, and an eye-to-hand RGB camera captures the image of the arm and the cube. By rotating the one, two, and six joints of the robot arm, the Rubik's cube can be rotated around the pitch, yaw, and roll axes, respectively. At each time step, the robot rotates one joint 15° clockwise or counterclockwise, and the camera captures an RGB image as the state of the agent. In Fig. 5 (right), we visualize an example of a state. To help the agent focus on the Rubik's cube, we enlarge the cube in the state and put the enlarged image patch in the left-bottom of the image state.

The goal is an RGB image in which the Rubik's cube is in the desired pose. The condition of reaching a goal is the pose difference is smaller than 10°. We record the pose of the cube in each state and an image goal, and then, we can calculate the pose difference. In real-world tasks, we can estimate the object pose with an off-the-shelf 6-D pose estimation model. To evaluate the generalization ability of our method, we corrupt the training and test goal in the same way as in the fetch environment. Examples of the corrupted image goals are shown in Fig. 6.

4) Embodied QA With Real Quadcopter: To evaluate our method in a real-world task, we conduct experiments on a real quadcopter, TELLO drone by Ryze Robotics (see Fig. 7, quadcopter). The goal is to successfully navigate to a location that contains sufficient information for answering a given question about an object on a shelf. We highlight that the policy should be generalized to handle new questions because it is impossible to train the policy on every possible question that a human may ask.

As shown in Fig. 7, the quadcopter is connected to a PC (MacBook Pro, 14-in, 2023) through the transmission control protocol (TCP). The quadcopter is facing a shelf where 24 different objects are placed. The built-in front-facing camera of the quadcopter captures 1280×720 RGB images at a frame rate of 30 frames/s (fps). An example first-person image is shown in Fig. 7 (first-person view). The first-person RGB images are sent to the PC as the policy observations. Then, the policy running on the PC sends commands (move left, right, up, and down with a step length of 20 cm) to the quadcopter to execute. An additional third-person RGB camera (with a resolution of 2532×1170 and 30 fps) is connected to the PC with a cable. The camera records the quadcopter trajectory from a third-person view. An example third-person image is shown in Fig. 7 (third-person view). A trajectory will be terminated when the trajectory length reaches 50 or the desired goal has been reached.

The goal space includes questions about objects on the shelf shown in Fig. 7 (third-person view). The questions are generated by gpt-3.5-turbo.[2] Specifically, we use the prompt "Print a Python list which contains 5 questions about [object]." We replace "[object]" with the objects on the shelf. For all the generated questions and video demos, please refer to the project website.[3] The external reward function (distinct from the intrinsic reward functions defined by InfoGoal and InfoBot [10]) only returns a reward of 1 when the desired goal is reached. The condition of reaching the goal is that the first-person RGB image contains sufficient information to answer the given question. Since all the questions are asked about objects, we make an assumption that if the object mentioned in the question can be detected in the first-person image, the goal has been achieved. We train a YOLOv8 [32] object detection model with a collected labeled dataset, which includes 24 objects. With the data augmentation methods implemented in YOLOv8, the trained model running on the PC can reliably detect objects in the first-person images captured by the quadcopter.

Learning a policy via trial and error with a real robot is unsafe and time-consuming. Therefore, we adopt offline RL, which trains the policy on a precollected dataset and then evaluates the policy with the real robot in the real environment [45]. To avoid the distribution shift problem of offline RL [45], i.e., the mismatch between the state and action distribution in the dataset and the distribution encountered during the test, we collect the training dataset by manually controlling the quadcopter to cover the entire state and action space.

B. Evaluation Settings

1) Train–Test Split: For the object search task, we need to guarantee that a testing goal is located close to at least a training goal; otherwise, the trained policy is unlikely to find the test goal with a random search. Therefore, we randomly sample two objects in each room as training goals and leave the other objects as test goals, i.e., the train–test split is 8/16. This split strategy is reasonable since we assume that objects located in the same room can be found by similar goal-reaching policies.

For the object manipulation task, we discretize the Rubik's cube pose space via uniformly sampling poses ranging from $-180°$ to 180° every 15° for the three axes (pitch, yaw, and roll). We uniformly sample eight poses as the training goals

[2]https://platform.openai.com/docs/models.

[3]https://qiming-zou.github.io/infogoal.github.io/.

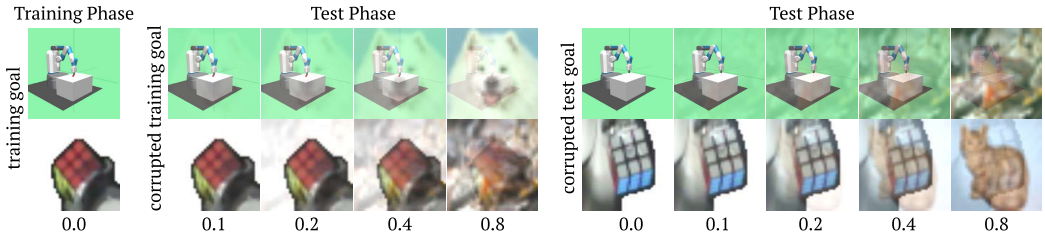


Fig. 6. Examples of image goals in the fetch and object manipulation environments during the training and test phases. During the test phase, the agent is commanded to reach both a corrupted test goal corresponding to a new goal state and a corrupted training goal corresponding to an observed goal state in the training phase. The degree of corruption is controlled by $w_{\text{bkg}} \in [0, 1]$.

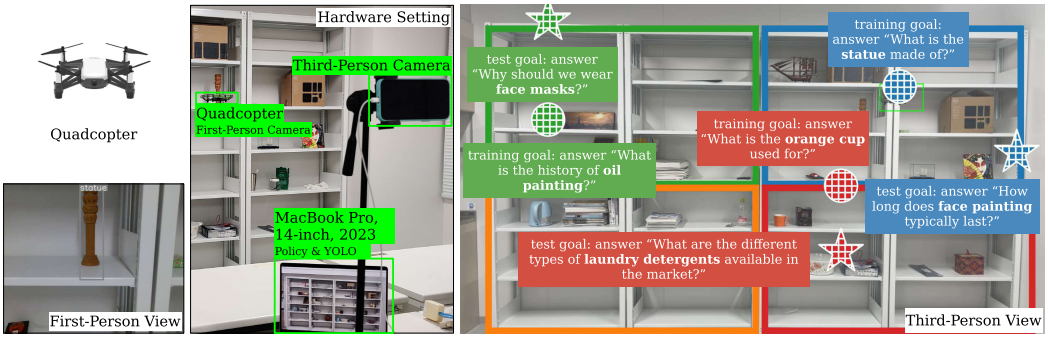


Fig. 7. Hardware setting, all the training goals (circles), and example test goals (stars) for EmboidedQA task. The position where the goals can be reached is visualized in the third-person view. Note that the shelf is divided into four regions indicated by the four bounding boxes with orange, green, blue, and red. The color of the circles and stars corresponds to the region where the corresponding objects (which the question, i.e., goal, asked about) can be detected.

and sample 32 test poses in the neighborhood of the training goals (the maximum degree difference to the closest training goal is 90°). Similarly, for the fetch environment, we uniformly sample eight positions in the goal space as training goals and sample 32 test positions in the neighborhood of the training goals (the maximum Euclidean distance to the closest training goal is 20 cm).

For the embodied QA task, the policy is trained to reach four training goals. The training goals are related to four objects uniformly placed on the shelf, enabling the policy to learn to explore the entire state space uniformly. We generate four additional questions for each of the 24 objects as test goals. We visualize the example goals and corresponding goal states in Fig. 7 (third-person view). Note that during the training phase, the target object related to the test goals can be detected, but the specific questions that may be asked about these objects are unknown. Therefore, the test object names are considered as goals during training for desired goal relabeling and goal representation learning.

2) Test Settings: For the object search task, we compare the generalization ability of our method to baseline methods with test goals. For the fetch task and the object manipulation task, we compare the generalization ability of our method to baseline methods with three types of test goals (as shown in Fig. 6).

1) *Test Goal:* A goal with a goal state, i.e., the state at which the goal can be reached, which is different from the goal states of training goals.
2) *Corrupted Test Goal:* A test goal with different degrees of corruption.
3) *Corrupted Training Goal:* A training goal with different degrees of corruption.

Note that we corrupt the image goal by overlaying the random background image, as shown in Fig. 6.

For the embodied QA task, the fully trained policy is evaluated in a real-world environment. There are 20 test objects, each corresponding to four test questions, i.e., goals. We visualize four example test goals in Fig. 7 (third-person view).

C. Baseline Methods

We compare our InfoGoal with the following representation learning methods.

1) *Variational Autoencoder (VAE) [1], [14]:* This is a commonly used method to embed goals by learning a goal representation that can easily reconstruct the goal input.
2) *Discrete Goal-Conditioned Reinforcement Learning (DGRL) [12]:* This work converts a goal representation that can reconstruct the goal input into a discrete representation and improves the generalization of an RL agent to unseen test goals.
3) *Reinforcement Learning With Augmented Data (RAD) [46]:* This is an RL algorithm that randomly adjusts the state inputs to learn a robust RL policy. We augment the goal inputs via random translation (for image goals) and random amplitude scaling (for text goals), which have been empirically proved efficient for policy generalization [47].
4) *Transfer and Exploration via the Information Bottleneck (InfoBot) [10]:* This method adopts an auxiliary loss function that minimizes the MI between the goal representation and the goal input. Combining with the learned goal representation and an intrinsic reward, this method learns a general policy for the training goals.
5) *COGOAL [15]:* This method adopts an auxiliary loss function that minimizes a contrastive loss between representations of adjacent goals.

TABLE I
TEST SUCCESS RATE OF INFOGOAL AND BASELINE METHODS

	Test Goals						Corrupted Test Goals				Corrupted Training Goals			
	ObjSearch	ObjMani	Reach	Push	PickAndPlace	EmbodiedQA	ObjMani	Reach	Push	PickAndPlace	ObjMani	Reach	Push	PickAndPlace
InfoGoal (our)	**0.93**	**0.92**	**0.91**	0.84	**0.90**	**1.0**	**0.95**	**0.84**	0.74	**0.82**	**1.0**	**0.91**	0.88	0.82
InfoBot	0.85	0.84	0.70	**0.95**	0.72	**1.0**	0.79	0.8	**0.94**	0.76	0.88	0.76	0.92	0.69
COGOAL	0.45	0.42	0.34	0.52	0.17	/	0.18	0.0	0.17	0.0	0.48	0.12	0.31	0.17
GCB	0.41	0.53	0.59	0.78	0.61	0.15	0.35	0.1	0.28	0.25	0.75	0.13	0.38	0.53
VAE	0.21	0.27	0.22	0.61	0.37	/	0.19	0.14	0.51	0.24	**1.0**	0.84	**1.0**	0.91
DGRL	0.33	0.41	0.38	0.7	0.38	/	0.25	0.33	0.5	0.31	**1.0**	0.9	0.98	**0.95**
RAD	0.19	0.63	0.33	0.65	0.33	/	0.44	0.16	0.46	0.15	0.66	0.62	0.62	0.46
DQN/HRAC/CQL	0.18	0.4	0.06	0.48	0.19	0.05	0.3	0.09	0.25	0.23	0.75	0.67	0.56	0.62
HER	0.51	0.88	0.59	0.75	0.71	/	0.52	0.23	0.48	0.38	0.69	0.44	0.53	0.44
CHER	0.53	0.83	0.48	0.76	0.67	/	0.5	0.21	0.52	0.35	0.69	0.34	0.59	0.5
GHRL	0.55	0.84	0.61	0.81	0.68	0.65	0.48	0.22	0.48	0.32	0.72	0.42	0.78	0.41

6) *GCB [11]:* This method learns a state-goal encoder that helps the policy generalize to unseen state-goal pairs. GCB groups the representations of state-goal pairs that lead to similar future rewards and states.

We also compare our method to an IB-based state representation learning method, **SSE-DB** [24], to demonstrate its complementary relationship with state representation learning. SSE-DB learns a state representation by maximizing the MI between temporally adjacent states and uses an intrinsic reward to encourage the agent to explore states with high information gain. Different from our work, the target of SSE-DB is to facilitate exploration in order to improve the sample efficiency of standard RL.

We compare our method to standard goal-conditioned RL methods. Since our InfoGoal adopts the idea of HER, in particular, we compare it to HER-based methods.

1) *DQN [8], [31]:* This is a standard value-based algorithm for discrete action tasks. For a goal-conditioned task, we use a universal value function approximator [8] that takes a state and a goal as input and outputs a predicted state-goal value.
2) *Conservative Q-Learning (CQL) [45]:* This is a commonly used offline RL method. Following [48], we adopt CQL for offline goal-conditioned RL by modifying it to take both the state and goal as input.
3) *Hierarchical Reinforcement Learning With k-Step Adjacency Constrain (HRAC) [49]:* This hierarchical RL method utilizes a high-level policy to assign subgoals to a low-level policy, which subsequently executes these subgoals in order to achieve the desired overall goal.
4) *HER [18]:* This is a commonly used goal-conditioned RL method. HER utilizes failed goal-reaching trajectories by relabeling the trajectories with reached states.
5) *Curriculum-Guided Hindsight Experience Replay (CHER) [19]:* This work adaptively selects the failed experiences for replay according to the proximity to the desired goal and the diversity of pseudogoals.
6) *Generalized Hindsight for Reinforcement Learning (GHRL) [20]:* This work utilizes inverse RL to infer the pseudogoal that best matches the failed experience. Then, the failed experience is relabeled with this new goal.

To accurately compare the performance of various methods, we apply them in combination with DQN on object search that has a discrete action space. For fetch and object manipulation tasks, which have continuous action spaces, we apply state and goal representation learning methods in combination with HRAC. We need to evaluate the test success rate after the agent has converged on the training goals. To improve the sample efficiency in the training phase, we pretrain the low-level policy of HRAC that takes the subgoal, i.e., a target state, as input and generates a trajectory toward the subgoal. For the embodied QA task, we apply state and goal representation learning methods in combination with the offline RL method, i.e., CQL.

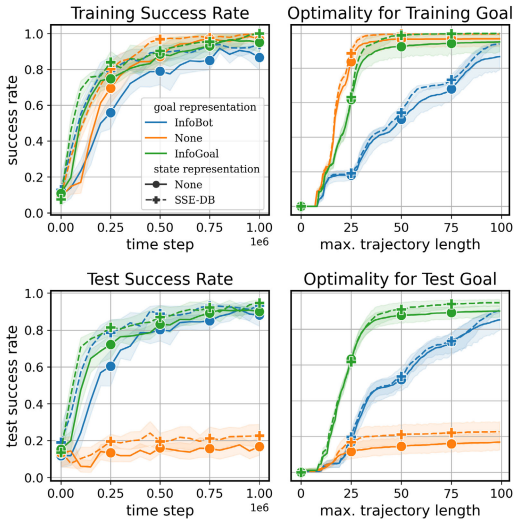


Fig. 8. Effect of IB-based state representation learning, i.e., SSE-DB, in terms of (left) the sample efficiency and (right) the optimality of the learned policy. The goal and state representation learning methods are combined with vanilla DQN. The label "None" represents that DQN is not combined with the representation learning method. The optimality is evaluated by the training and test success rate versus the maximum trajectory length. The optimal policy tends to achieve a high success rate with the minimum trajectory length.

D. Comparison Results

We evaluate our method on a standard goal-conditioned fetch environment that contains three tasks, i.e., FetchReach, FetchPush, and FetchPickAndPlace. We also apply our method to real-world tasks, i.e., object search, object manipulation, and EmbodiedQA.

In Table I, we quantify the test performance by the test success rate on test goals, corrupted training goals, and corrupted test goals. The best and second-best results are highlighted with bold fonts and are underlined, respectively. For the EmbodiedQA task, since it is time-consuming to evaluate all the baseline methods, we evaluate vanilla CQL and combine CQL with methods that have consistently performed well in the previous sets of experiments. We use slashes in the table to represent no value. The corrupted goals have a random

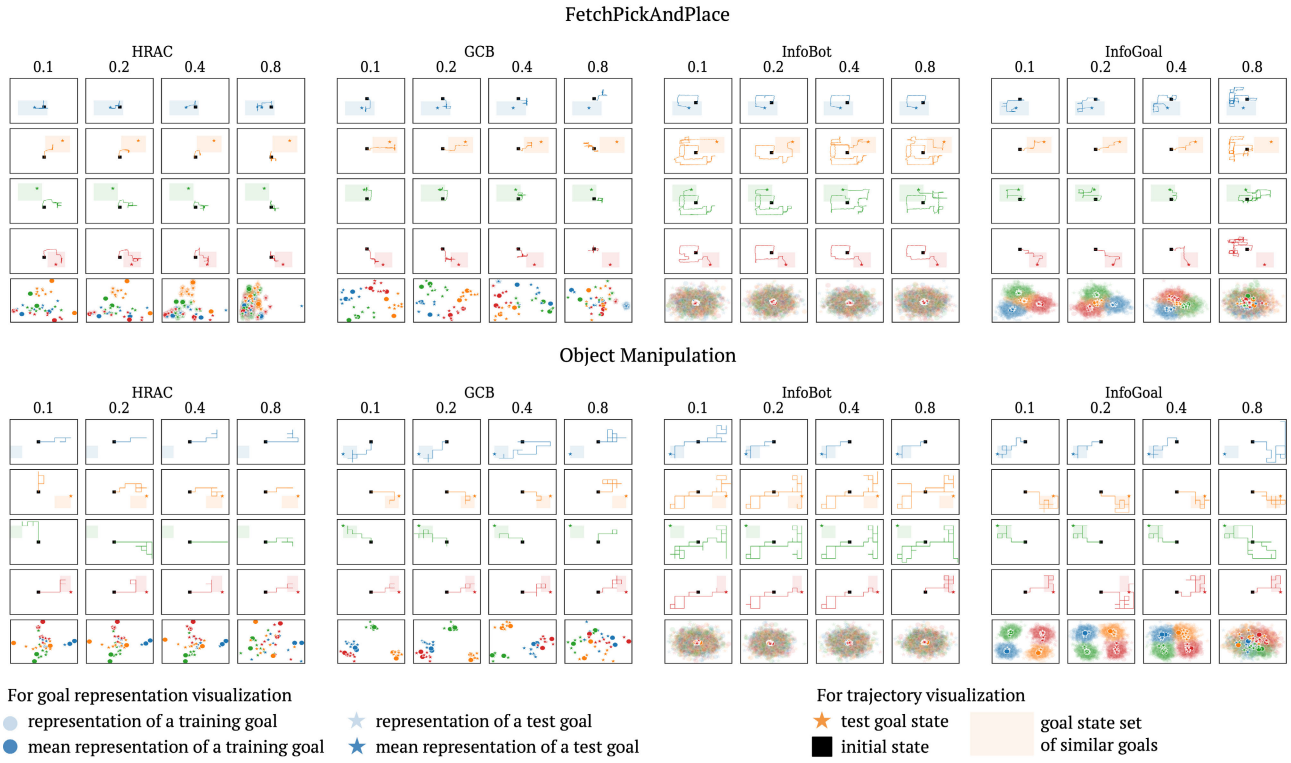


Fig. 9. Learned goal representations and trajectories with only external sparse rewards for FetchPickAndPlace (upper plots) and object manipulation (lower plots). We visualize the learned goal representations of our method and baseline methods under different degrees of corruption. The degree of corruption is quantified using the weight of the overlaid random background image, i.e., 0.1, 0.2, 0.4, and 0.8. Since the goal representation follows a Gaussian distribution, we visualize the mean of the learned Gaussian distribution, i.e., solid shapes, and samples from the distribution, i.e., shapes with faded colors. Shapes with the same color correspond to similar goals and should be grouped in the same cluster. We visualize the learned trajectories for corrupted test goals. Note that we plot the trajectories in the first two dimensions of the object position (for FetchPickAndPlace), i.e., the x-axis and the y-axis, or the pose (for object manipulation), i.e., pitch and yaw, which helps us to visualize and analyze the movement patterns.

background overlay with a weight of $w_{\text{bkg}} = 0.4$. We evaluate the test success rates after the convergence of the methods on the training goals. From Table I, we observe that HER-based methods, such as HER, CHER, and GHRL, achieve better test performance compared to vanilla goal-conditioned RL methods, i.e., DQN and HRAC. The reason is that these methods relabel a trajectory with a test goal, which helps the agent learn how to reach the test goal. However, HER-based methods tend to overfit the goal-reaching tasks and achieve significantly worse success rates of reaching corrupted goals.

For a test goal and a corrupted test goal, from Table I, we observe that our method performs the best. The first reason our method outperforms baseline methods is the IB-based goal representation that increases the overlapping between similar training and test goals in the latent space. The second reason is that our intrinsic reward function encourages the agent to search around the goal states of similar training goals. For corrupted training goals, VAE and DGRL perform well. These two methods learn goal representation by reconstructing the goal input. Since the image goal contains rich task-relevant information, i.e., similar goals have similar visual features, VAE and DGRL encode task-relevant information in the goal representation. To further justify the superior performance of our method, in Section V-E, we visualize the learned goal representation of our method and several related baseline methods. We also visualize the trajectories for a subset of test goals to show the importance of the learned searching behavior.

Since IB-based state representation learning is complementary to InfoGoal, we compare our method to SSE-DB [24] in terms of the sample efficiency, success rate, and learned policy optimality. We conduct the comparison in the object search task in which a state corresponds to an image, which necessitates representation learning. As shown in Fig. 8 (left column), SSE-DB consistently improves the sample efficiency of DQN, InfoBot, and InfoGoal, i.e., it achieves a high training and test success rates with fewer time steps. However, as shown in Fig. 8 (bottom left), DQN without goal representation still achieves a low test success rate when combined with SSE-DB. Therefore, the improvement introduced by SSE-DB over the test success rate is not as significant as our InfoGoal or InfoBot. Besides, SSE-DB cannot improve the policy optimality of InfoBot. As shown in Fig. 8 (right column), the trained policy still generates a long trajectory to reach the desired training or test goal. The reason is that the state space is shared by all the training and test goals in an environment, and the agent deals with the same state space despite the desired goals (or goal representations) being similar or dissimilar.

E. Understanding Gains From InfoGoal via Representation Visualization

To understand how InfoGoal improves policy performance, we visualize the learned goal representations with principal component analysis (PCA) [50] and the trajectories for the test goals and the corrupted test goals in Figs. 9–11. To explain

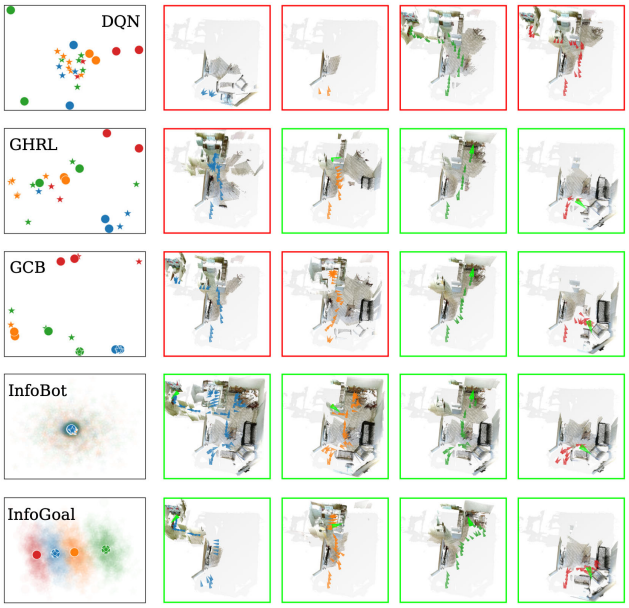


Fig. 10. Learned goal representations and trajectories for test goals in the object search environment. The arrow indicates the robot's position and orientation at each time step. The green arrow represents the pose where the robot reaches the desired goal. The color of the bounding box represents whether the robot has reached the desired goal (green: success; red: failure).

Fig. 11. Learned goal representations (left column) and trajectories (right columns) for test goals in EmboidedQA task. Each row corresponds to a method. The test and training goal representations and corresponding desired quadcopter positions are indicated by a star and a circle, respectively. The shapes and trajectories are colored according to the adjacency of goal states. In other words, goals with the same color can be reached adjacently.

how InfoGoal learns to search in the state region, we visualize the intrinsic rewards in Fig. 12.

Fig. 9 can be divided into two groups of plots: the upper plots visualize results for FetchPickAndPlace, while the bottom plots visualize results for object manipulation. The upper and bottom plots contain four columns of results, respectively, where each column corresponds to the results of a goal-conditioned RL method. For each method, we display the learned goal representations and trajectories under different degrees of corruption, specifically 0.1, 0.2, 0.4, and 0.8, across four columns. For each corruption level, we visualize four

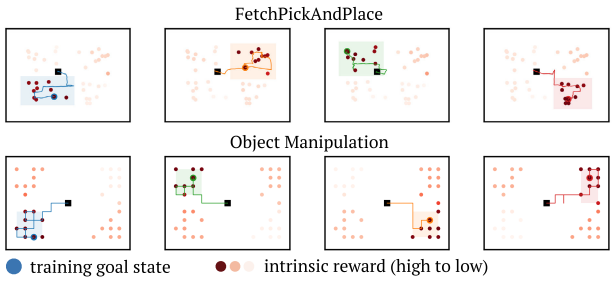


Fig. 12. Intrinsic reward for training goals. Each plot visualizes the intrinsic rewards and a learned trajectory for a training goal located in a state region.

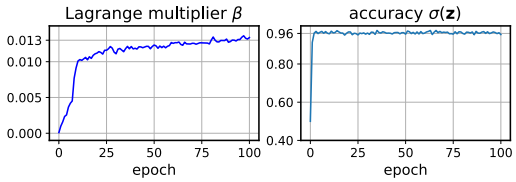


Fig. 13. Evolution of the Lagrange multiplier and the accuracy with our adaptive Lagrange multiplier adjustment in the FetchPickAndPlace task. We report the converged β for all tasks in Table II in the Supplementary Material.

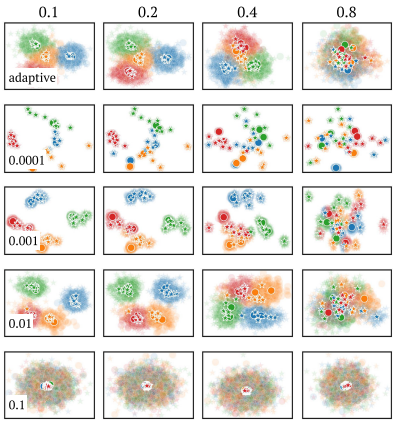


Fig. 14. Learned goal representations of InfoGoal with different Lagrange multiplier values (shown in each row) under different corruption degrees (shown in each column) for the FetchPickAndPlace task.

trajectories whose desired goals are sampled from different regions in the goal space for each method. We have five rows of plots to present these trajectories, along with the representation of training and test goals in the entire goal space. We assume that the target poses that are located closely (points with the same color) tend to share similar optimal goal-reaching policies. Therefore, ideally, the points with the same color should be grouped into a cluster in the goal representation.

From the visualization of goal representations in Figs. 9–11, we observe that, without auxiliary goal representation learning, the test goal representation would be mixed, and the agent would generate trajectories leading to states that are irrelevant to the goal. GCB groups similar goals in the latent space, and thus, the agent can directly go to the state region where the test goal can be found. However, since there is little overlap between goal representations, the agent learns policies that are specific to each training goal, and thus, for a test goal, the agent goes to the most similar training goals without searching around the test goal. InfoBot is designed to reduce specific information in the training goal representations and learn a

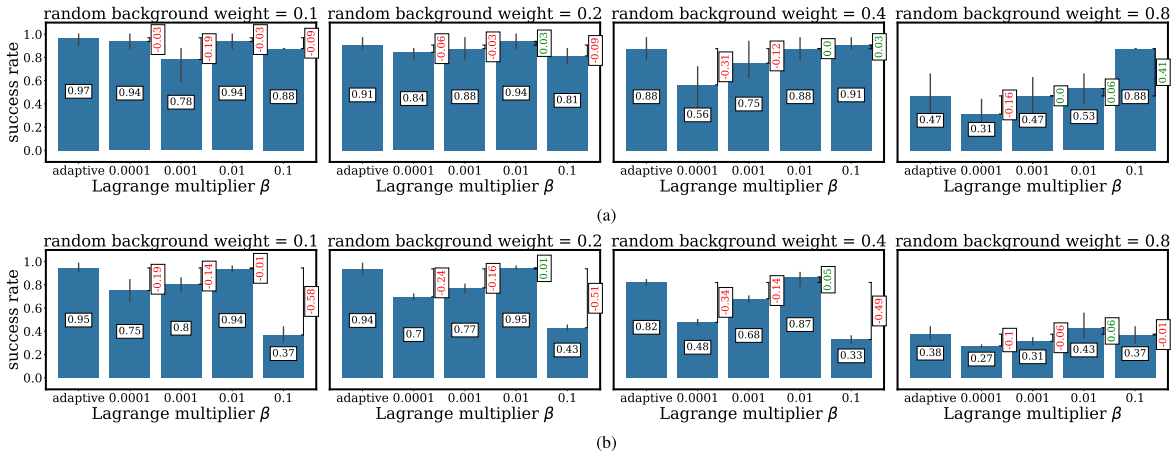


Fig. 15. Test success rate with adaptive Lagrange multiplier adjustment and fixed Lagrange multiplier value ranging from 0.0001 to 0.1. The mean and standard deviation of the success rate are over three fetch environments. (a) Success rate of reaching corrupted training goals. (b) Success rate of reaching corrupted test goals.

general policy for all the training goals. InfoBot learns a goal representation that is close to a standard Gaussian distribution. Since the agent takes similar goal representations as input but is commanded to reach diverse goals, the agent learns a policy that searches the entire environment to reach each goal. This general policy can significantly improve the test success rate but harms the policy optimality, i.e., the policy generates long trajectories to reach the test goals. The InfoBot policy would easily drain the battery of a robot if it operates in a larger environment.

As shown in Figs. 9–11, with InfoGoal, similar goals share similar goal representation (a Gaussian distribution). Besides, as shown in Fig. 12, the intrinsic reward of a (state, desired goal) pair is high when the achieved goal at that state is close to the desired goal in the latent space. From the figure, we observe that, for a training goal, the intrinsic rewards are high in the state region where the training goal is located. As a result, the learned policy searches broadly in the state region where goals that are similar or identical to the desired goal can be found.

F. Effect of Adaptive Lagrange Multiplier Adjustment

We compare the test performance of InfoGoal under different settings of the Lagrange multiplier β. Specifically, we compare our adaptive method to fixed β with values 0.1, 0.01, 0.001, and 0.0001.

In Fig. 13, we visualize the evolution of β and $\sigma(\mathbf{z})$, i.e., the accuracy of discriminating similar and dissimilar goals based on information contained in $\mathbf{z}$. According to Algorithm 1, each epoch consists of 100 updates for both accuracy and the Lagrange multiplier. The training agent will use the goal representation learned with the updated β in each epoch. We report the average β value (around 0.013) once the updating has reached convergence. As shown in Fig. 13 (left), our method adaptively increases the Lagrange multiplier to achieve a more significant reduction in information while maintaining accuracy at approximately 0.96 [as shown in Fig. 13 (right)], thereby guaranteeing sufficient task-relevant information. Note that the accuracy does not need to be 1.0 since, for a goal-reaching task, the goal representation is sampled from the learned Gaussian distribution at every time step. Therefore, the agent will work well if the representation contains sufficient task-relevant information in most of the time steps.

In Fig. 14, we visualize the learned goal representations with different β settings. As shown in the visualization, the goal representation is sensitive to β. When β is relatively high, the infomin term would overwhelm the infomax term, and thus, the goal representation is close to the standard Gaussian distribution, which contains minimum information. When β is relatively low, the infomax term would overwhelm the infomin term, and thus, the task-irrelevant information cannot be removed. As a result, the representation of similar goals would overlap each other to a smaller degree. As shown in Fig. 15, for three fetch tasks, the mean test success rate increases between $\beta = 0.0001$ and $\beta = 0.01$, followed by a drop. Since β balances the tradeoff between the infomin term and the infomax term, it is difficult to set β. We circumvent this difficulty by adaptively adjusting β along the optimization of the infomax and infomin terms. Consequently, as shown in Fig. 14, in the latent space, our method groups the representation of similar goals into clusters in which the representations overlap each other. Therefore, without setting a specific value of β, our method successfully balances the infomax and infomin terms. As shown in Fig. 15, for three fetch tasks, the mean success rate of our method is comparable to the best performance achieved with a fixed β.

VI. Conclusion

In this work, InfoGoal learns a minimal and sufficient goal representation that significantly improves the generalization ability of goal-conditioned RL. We demonstrated the performance of InfoGoal in simulated tasks, i.e., an object search task and robot arm fetching tasks, and real-world tasks, including an object manipulation task and an embodied question-answering task. The experimental results show that InfoGoal enables representations of similar training and test goals to follow a similar Gaussian distribution. As a result, the learned behavior for reaching the training goal generalizes well to reach new test goals with near-optimal trajectories. In addition, we observed that the intrinsic reward function derived from the learned goal representation encourages the policy search around states where the desired goal can be

found with a high probability. Based on the learned targeted searching behavior, InfoGoal significantly improves the success rate of reaching test goals with uninformative inputs, i.e., text goals, and goals with varying degrees of corruption, i.e., corrupted image goals. InfoGoal relies on the adjacency relationship between goals to learn representations, which means that the performance may suffer if the goals are sparsely located. A promising direction for future work is to learn goal representations that capture long-distance relationships between goals using specific neural network architectures, such as transformers [51].

REFERENCES

[1] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *Proc. NIPS*, 2018, pp. 9209–9220.

[2] Y. Zhu et al., "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3357–3364.

[3] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8846–8852.

[4] X. Yang et al., "Hierarchical reinforcement learning with universal policies for multistep robotic manipulation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 4727–4741, Sep. 2022.

[5] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giró-i-Nieto, and J. Torres, "Explore, discover and learn: Unsupervised discovery of state-covering skills," in *Proc. ICML*, vol. 119, 2020, pp. 1317–1327.

[6] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proc. CoRL*, 2019, pp. 1025–1037.

[7] L. P. Kaelbling, "Learning to achieve goals," in *Proc. IJCAI*, 1993, pp. 1094–1099.

[8] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. ICML*, 2015, pp. 1312–1320.

[9] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," 2022, *arXiv:2201.08299*.

[10] A. Goyal et al., "InfoBot: Transfer and exploration via the information bottleneck," in *Proc. ICLR*, 2019.

[11] P. Hansen-Estruch, A. Zhang, A. Nair, P. Yin, and S. Levine, "Bisimulation makes analogies in goal-conditioned reinforcement learning," in *Proc. ICML*, 2022, pp. 8407–8426.

[12] R. Islam et al., "Discrete compositional representations as an abstraction for goal conditioned reinforcement learning," in *Proc. NIPS*, 2022, pp. 3885–3899.

[13] O. Shamir, S. Sabato, and N. Tishby, "Learning and generalization with the information bottleneck," *Theor. Comput. Sci.*, vol. 411, nos. 29–30, pp. 2696–2711, Jun. 2010.

[14] V. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine, "Skew-fit: State-covering self-supervised reinforcement learning," in *Proc. ICML*, 2020, pp. 7783–7792.

[15] Q. Zou and E. Suzuki, "Contrastive goal grouping for policy generalization in goal-conditioned reinforcement learning," in *Proc. ICONIP*, 2021, pp. 240–253.

[16] X. Liu et al., "Self-supervised learning: Generative or contrastive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 857–876, Jan. 2023.

[17] Z. Pan, L. Niu, J. Zhang, and L. Zhang, "Disentangled information bottleneck," in *Proc. AAAI*, 2021, pp. 9285–9293.

[18] M. Andrychowicz et al., "Hindsight experience replay," in *Proc. NIPS*, 2017, pp. 5048–5058.

[19] M. Fang, T. Zhou, Y. Du, L. Han, and Z. Zhang, "Curriculum-guided hindsight experience replay," in *Proc. NIPS*, 2019, pp. 12602–12613.

[20] A. C. Li, L. Pinto, and P. Abbeel, "Generalized hindsight for reinforcement learning," in *Proc. NIPS*, 2020, pp. 7754–7767.

[21] M. Igl et al., "Generalization in reinforcement learning with selective noise injection and information bottleneck," in *Proc. NIPS*, 2019, pp. 13956–13968.

[22] V. Pacelli and A. Majumdar, "Learning task-driven control policies via information bottlenecks," in *Proc. RSS*, 2020.

[23] B. Eysenbach, R. Salakhutdinov, and S. Levine, "Robust predictable control," in *Proc. NIPS*, 2021, pp. 27813–27825.

[24] C. Bai et al., "Dynamic bottleneck for robust self-supervised exploration," in *Proc. NIPS*, vol. 34, 2021, pp. 17007–17020.

[25] K. Lee et al., "Predictive information accelerates learning in RL," in *Proc. NIPS*, 2020, pp. 11890–11901.

[26] B. Mazoure, R. T. des Combes, T. Doan, P. Bachman, and R. D. Hjelm, "Deep reinforcement and InfoMax learning," in *Proc. NIPS*, 2020, pp. 3686–3698.

[27] R. Y. Tao, V. François-Lavet, and J. Pineau, "Novelty search in representational space for sample efficient exploration," in *Proc. NIPS*, 2020, pp. 8114–8126.

[28] K. Rakelly, A. Gupta, C. Florensa, and S. Levine, "Which mutual-information representation learning objectives are sufficient for control?" in *Proc. NIPS*, 2021, pp. 26345–26357.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[30] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, U.K., 1989.

[31] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] G. Jocher, A. Chaurasia, and J. Qiu. (Jan. 2023). *YOLO by Ultralytics*. [Online]. Available: https://github.com/ultralytics/ultralytics

[33] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," in *Proc. Allerton Conf. Commun., Control, Comput.*, 1999, pp. 368–377.

[34] E. L. Lehmann and H. Scheffé, "Completeness, similar regions, and unbiased estimation-Part I," *Sankhya, Indian J. Statist.*, vol. 10, no. 4, pp. 305–340, 1950.

[35] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Apr. 2015, pp. 1–5.

[36] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for MDPs," in *Proc. ISAIM*, 2006, pp. 531–539.

[37] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014.

[38] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *Proc. ICLR*, 2017.

[39] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[40] M. Plappert et al., "Multi-goal reinforcement learning: Challenging robotics environments and request for research," 2018, *arXiv:1802.09464*.

[41] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *Proc. ICLR*, 2021.

[42] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg, "A dataset for developing and benchmarking active vision," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1378–1385.

[43] P. Anderson et al., "Vision-and-Language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3674–3683.

[44] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 3980–3990.

[45] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. NIPS*, 2020, pp. 1179–1191.

[46] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," in *Proc. NIPS*, 2020, pp. 19884–19895.

[47] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *Proc. ICLR*, 2021.

[48] R. Yang et al., "Rethinking goal-conditioned supervised learning and its connection to offline RL," in *Proc. ICLR*, 2022.

[49] T. Zhang, S. Guo, T. Tan, X. Hu, and F. Chen, "Generating adjacency-constrained subgoals in hierarchical reinforcement learning," in *Proc. NIPS*, 2020, pp. 21579–21590.

[50] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[51] A. Vaswani et al., "Attention is all you need," *Proc. NIPS*, vol. 30, 2017, pp. 5998–6008.

Qiming Zou received the B.E. degree from the Anhui University of Technology, Ma'anshan, China, in 2017, and the M.E. degree from the Harbin Institute of Technology, Harbin, China, in 2019. He is currently pursuing the Ph.D. degree with Kyushu University, Fukuoka, Japan.

His research interests include reinforcement learning, representation learning, and robotics.

Einoshin Suzuki received the B.E., M.E., and Dr.Eng. degrees from The University of Tokyo, Tokyo, Japan, in 1988, 1990, and 1993, respectively.

He has been a Professor with Kyushu University, Fukuoka, Japan, since 2006. His research interests include data mining, machine learning, and artificial intelligence.

Dr. Suzuki received the Best Paper Award at Pacific Rim International Conference on Artificial Intelligence (PRICAI) 2021. He has served in several chair positions at international conferences, such as the Vice-Chair of IEEE International Conference on Data Mining (ICDM) Program Committee (PC) in 2004 and 2014 and the Area Chair of European Conference on Machine Learning (ECML)/European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) in 2010 and 2017 and from 2019 to 2021.