Discrete Compositional Representations as an Abstraction for Goal-Conditioned RL

Anonymous Author(s)
Affiliation
Address
email

Abstract

Goal-conditioned reinforcement learning (RL) is a promising direction for training agents that are capable of solving multiple tasks and reach a diverse set of objectives. How to *specify* and *ground* these goals in such a way that we can both reliably reach goals during training as well as generalize to new goals during evaluation remains an open area of research. Defining goals in the space of noisy and high-dimensional sensory inputs poses a challenge for training goal-conditioned agents, or even for generalization to novel goals. We propose to address this by learning compositional representations of goals and processing the resulting representation via a discretization bottleneck, for coarser specification of goals, through an approach we call DGRL. We show that applying a discretizing bottleneck can improve performance in goal-conditioned RL setups, by experimentally evaluating this method on tasks ranging from maze environments to complex robotic navigation and manipulation tasks. Additionally, we prove a theorem lower-bounding the expected return on out-of-distribution goals, while still allowing for specifying goals with expressive combinatorial structure.

1 Introduction

Reinforcement Learning is a popular and highly general framework [22, 49] focusing on how to select actions for an agent to yield high long-term sum of rewards. An important question is how to control the desired behavior of an RL agent during both training and evaluation [21]. One way to control this behavior is by specifying a reward signal [44, 46]. While this approach is very general, the reward signal can be hard to design and may not be the most informative form of feedback. The credit assignment problem in RL can become difficult when the reward signal is sparse [51, 52, 28, 29, 47], such as policy gradients becoming nearly flat in regions where reward is almost never achieved. Generalization can also suffer if the agent only learns one way to achieve a high reward rather than learning a diverse set of skills for coping with novel challenges [16].

One potential way to flexibly specify and ground the desired behavior of RL agents is by training agents that receive a reward when they reach a goal specified explicitly to them [19]. In this approach, called *Goal-Conditioned RL*, a single agent is trained to reach a diverse set of goals, and is given a reward only when it reaches the goal that it was instructed to reach [50, 40, 36]. This provides a richer signal for the agent than simply collecting more samples oriented around a single goal, as reaching multiple goals requires the agent to learn a more diverse and robust set of skills. It also allows for more flexible and tightly constrained control over the desired behavior of a learned agent [10, 3, 24]. Moreover, the diversity of goals seen during training should help improve both credit assignment and generalization [40, 35].

While this framework is promising, it introduces two new challenges: *goal grounding* [6, 1] and *goal specification* [2]. Goal grounding refers to defining the goal space and goal specification refers to selecting what goal the agent should try to reach in a given context. The agent is only rewarded in goal-conditioned RL when it reaches the reward which it was specified to reach, whereas in goal-free RL a reward is provided regardless of any such specification, which makes the nature of the agent's task fundamentally different.

What makes grounding and specifying goals challenging? Consider trying to train a goal-conditioned RL agent to pick up various fruits from a table. For example, we may want it to pick up a red apple or a green pear (illustrated in Figure 1). The number of possible goals of interests may be fairly small, such as the set of all valid combinations of fruits and their colors, while the number of possible observations of goals is extremely large when working in a rich observation space (e.g images from a camera). *Goal Grounding* refers to this challenge of relating high-dimensional observations and the space of relevant goals. *Goal Specification* refers to picking a suitable goal for the agent to reach and computing an appropriate reward when it is reached. It also implies specifying goals reachable in the agent's current context [31, 23]. Goal specification could be done either manually by a developer or by another RL agent, such as a high-level agent which generates goals that a lower-level agent tries to reach [10, 3, 24, 18]. Goals specified in language are an excellent fit for these desiderata, as language is a compressed discrete representation which is useful for out-of-distribution generalization, while being compositional and expressive [15, 9, 18, 14]. At the same time, connecting language feedback for an agent is non-trivial (requiring special assumptions or a labeling framework) [7].

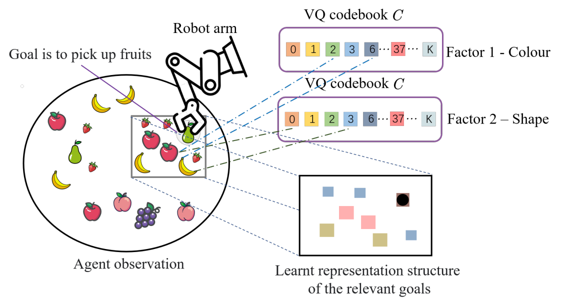

Figure 1: Illustration of learning *discrete* and *compositional* goal representations.

We propose to learn the goal representations with self-supervised learning (either by itself, or trained jointly with the downstream RL objective) while forcing the goal representations to be *discrete* and *compositional*. To perform this discretization, we use Vector-Quantization [53, 39, 26] which discretizes a continuous representation using a codebook of discrete and learnable codes. The approach proposed here (called DGRL) serves two complementary purposes. First, it provides a structured representation of the raw visual goals. By representing the visual goals as a *composition* of discrete codes from a learned dictionary, it makes it easier to ground unseen goals (i.e., goals not seen during training) using (novel) compositions of the discrete codes learned during the training process. We show empirically that this improves generalization performance of goal-reaching policies to unseen goals while remaining expressive enough. Second, the learned discrete codes can be used by another agent (like a higher-level policy in hierarchical RL) to specify sub-goals to an agent (i.e.. a lower-level policy) to complete the task (i.e., reach the goal). In this case goal-inference is learned end-to-end. The effectiveness of goal-conditioned HRL relies on the specification of semantically meaningful sub-goals. Using compositional discrete sub-goals allows the higher-level policy to specify semantically meaningful objectives to the lower-level policy.

2 Preliminaries

Goal-conditioned RL. We consider a goal-conditioned Markov Decision Process, where the goals $\mathcal{G}$ lie in the state space $\mathcal{S}$, i.e., $\mathcal{G} = \mathcal{S}$ (or in observation space $\mathcal{O}$). We denote a goal-conditioned policy as $\pi(a|s, g)$ (either stochastic or deterministic), and its expected total return as $J(\pi) = \mathbb{E}\Big[\sum_{t=0}^{T} R(s_t, g, a)\Big]$ where the goal g is either sampled from a distribution ρ_g or provided by another higher level policy $\pi^{h}_{\theta_h}(g \mid s)$. The value function V^{π} is additionally conditioned on goals, and is trained to predict the expected sum of future rewards conditioned on states and goals; $V^{\pi}(s, g) = \mathbb{E}\Big[\sum_{t=0}^{T} R(s_t, g, a) \mid s_0 = s; \pi\Big]$. As in standard RL, the objective in goal-conditioned RL is to maximize the expected discounted returns induced by the goal-conditioned policy.

Hierarchical Reinforcement Learning. We consider goal-conditioned settings in which the goals are specified in the observation space. In the hierarchical reinforcement learning (HRL) setup, goals are provided by a higher level policy $\pi^{h}_{\theta_h}(g|s_t)$. The higher level policy operates at a coarser time scale and chooses a goal $g_t \sim \pi^{h}_{\theta_h}(g|s_t)$ to reach for the lower level policy every K steps. The lower level policy executes primitive actions $\pi^{l}_{\theta_l}(a|s_t, g_t)$ to reach the goals specified by the high-level policy and is trained to maximize the intrinsic reward provided by the high-level policy. The higher level policy is trained to maximize the external reward i.e., the reward function specified by the MDP. Both the higher and lower level policies can be trained with any standard RL algorithms, such as Deep Q-Learning (DQN) [28] or policy optimization based algorithms [42, 43]. Alternately, one can also consider another setup for goal-conditioned RL, where the goals are provided by the environment $g_1, \ldots g_L$ and are part of the state or observation space. At each episode of training, one of the goals is sampled from the distribution of goals ρ_g and the policy is trained to reach the sampled goal. At test time, the agent can be evaluated either on its ability to reach goals within the distribution ρ_g, or for its out-of-distribution generalization capability to reach new kinds of goals.

In this work, we consider both the HRL and goal-conditioned setups, and evaluate the significance of learning a compositional representation of discrete latent goals in a series of complex goal-conditioned tasks.

Vector Quantized Representations. VQ-VAE [53, 39, 26] discretizes the bottleneck representation of an auto-encoder by adding a codebook of discrete learnable codes. The input is passed through an encoder. The output of the encoder is compared to all the vectors in the codebook, and the codebook vector closest to the continuous encoded representation is fed to the decoder. The decoder is then tasked with reconstructing the input from this quantized vector.

Self-supervised learning of representations. Several papers [25, 48, 45, 27] have demonstrated the benefits of using a pre-training stage where the representations of raw observations are learned using self-supervised objectives in a task-agnostic fashion. After the pre-training stage, the representations can be used for (and potentially also fine-tuned on) downstream tasks. These self-supervised representations have been shown to improve sample efficiency.

3 Discrete Goal-Conditioned Reinforcement Learning (DGRL)

In this section, we provide technical details on the proposed framework, DGRL, which consists of three parts: (a) learning representations of raw visual observations through self-supervised representation objectives, (b) processing the resulting representations via a learned dictionary of discrete codes, and (c) using the resulting discrete representations for downstream goal-conditioned and HRL tasks. With the learnt discrete goal representations, we describe in Section 5 how they can accelerate learning in complex navigation and manipulation tasks. The goal representation can be learned at the same time as the downstream-RL objective or pre-trained with self-supervised learning and then used as a fixed representation for RL.

3.1 Self-Supervised Goal Representation Learning

One can use any off-the shelf self-supervised method for learning representations of the raw state and the goal observations. We denote by ϕ the encoder network that takes as input the raw observation and maps it to a continuous embedding: $z_e = \phi(o_t)$. Here, we explore two different self-supervised techniques for learning representations. For simpler environments, we use a simple autoencoder with the reconstruction objective. For more complex environments, we use the Deep InfoMax approach [27] which optimizes for a contrastive objective as a proxy to maximize the mutual information between representations of nearby states in the same trajectory.

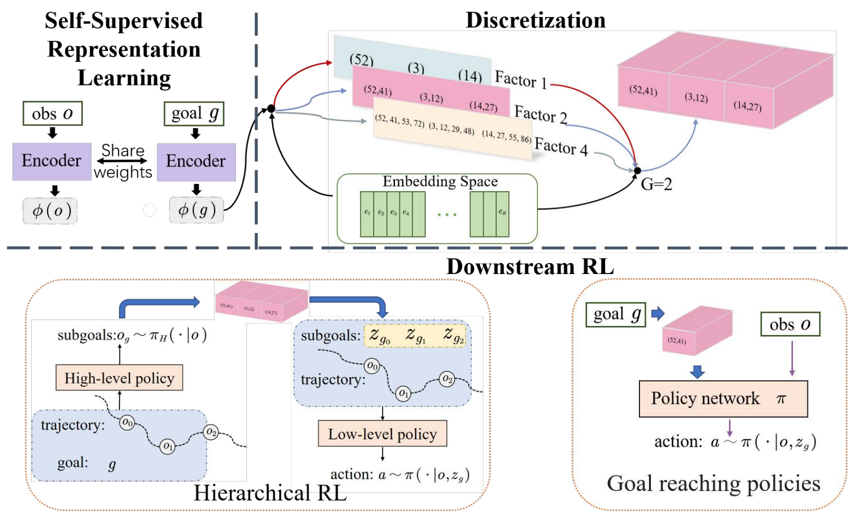

Figure 2: **Summary of Proposed DGRL Model** for improving *goal grounding* and *goal specification* by making goal representations *discrete* and *compositional*. We learn a latent representation for both observations and goals using a self-supervised learning method (sec. 3.1). We convert the learnt latent representation into discrete latents based on a VQ-VAE quantization bottleneck with multiple factor outputs (sec. 3.2). We use the resulting discrete representations for downstream RL tasks: (i) to train a goal-conditioned policy or value function, and (ii) in the context of goal-conditioned hierarchical reinforcement learning (sec. 3.3).

3.2 Processing continuous representations via a discrete codebook

We quantize the continuous representation into discrete latent variables, following the codebook method from [26], which generalizes the vector quantization used in VQ-VAE [53] to a list of discretized codes instead of a single discretized code. The discretization process for each vector $z_e \in \mathcal{H} \subset \mathbb{R}^m$ is described as follows. First, vector z_e is divided into G segments $c_1, c_2, \ldots, c_G$ with $z_e = \text{CONCATENATE}(c_1, c_2, \ldots, c_G)$, where each segment $c_i \in \mathbb{R}^{m/G}$ (such that $\frac{m}{G} \in \mathbb{N}^+$). Each continuous segment c_i is mapped separately to a discretized latent space vector $e \in \mathbb{R}^{L\times(m/G)}$ where L is the size of the discrete latent space (i.e., an L-way categorical variable):

$$e_{o_i} = \text{DISCRETIZE}(c_i), \quad \text{where } o_i = \underset{j\in\{1,\ldots,L\}}{\arg\min} ||c_i - e_j||.$$

These discrete codes, which we call the factors of the continuous representation z_e, are concatenated to obtain the final discretized vector z_q:

$$z_q = \text{CONCATENATE}(\text{DISCRETIZE}(c_1), \text{DISCRETIZE}(c_2), ..., \text{DISCRETIZE}(c_G)). \tag{1}$$

The loss for vector quantization is: $\mathcal{L}_{\text{discretization}} = \frac{\beta}{G}\sum_i^G ||c_i - \text{sg}(e_{o_i})||_2^2$.

The training procedure closely follows both [26] and [53]. Here, sg refers to a stop-gradient operation that blocks gradients from flowing into its argument, and β is a hyperparameter which controls the reluctance to change the code. We update e_{o_i} with an exponential moving average to encourage it to become close to the selected output segment c_i. This update sets the new value of e_{o_i} to be equal to $\eta e_{o_i} + (1-\eta)c_i$, where the value of η is a fixed hyperparameter controlling how quickly the moving average updates. The term $\sum_i^G ||c_i - \text{sg}(e_{o_i})||_2^2$ is the commitment loss, which only applies to the target segment c_i and trains the encoder that outputs c_i to make c_i stay close to the chosen discrete latent vector e_{o_i}. We trained the codebook together with other parts of the model by gradient descent. When there were multiple z_e vectors to discretize in a model, the mean of the codebook and commitment loss across all z_e vectors was used.

Summary. The multiple steps described above can be summarized by $z_q = q(z_e, L, G)$, where $q(\cdot)$ is the whole discretization process using the codebook, L is the codebook size, and G is the number of factors per vector. We train the representations for both the state and goal observations with a discretization bottleneck on the continuous representations resulting from the self-supervised pre-training. The number of factors G is a hyper-parameter. In our experiments, we try with different number of discrete factors $G = 1, 2, 4, 8, 16$, and found that $G = 16$ worked the best. Using discretization with more factors slightly increases computation but reduces the number of model parameters due to the codebook embeddings being reused across the different factors.

3.3 Using representations for downstream RL

We use the discrete representations for downstream RL tasks: (i) to train a goal-conditioned policy, and (ii) in the context of hierarchical reinforcement learning.

Goal-conditioned RL. Defining goals in the space of noisy, high-dimensional sensory inputs poses a challenge for generalization to novel goals because the encoder that maps the goal observations to the low dimensional latent representation may fail to generalize. One way to address this is to embed the continuous latent representation into a discrete representation such that the representation of the novel goal is mapped to the fixed set of latent discrete codes, and hence facilitate generalization to new combinations of these codes while making it easy for downstream learning to figure out the meaning of each discrete code. In this setup instead of feeding the continuous state and goal embedding we used their discretized versions, thus grounding goal representations in the input space.

We use the resulting representations for training a goal-conditioned policy $a_t \sim \pi^l_{\theta_l}(a|s_t, g_t)$ or a goal-conditioned action value function $Q(s_t, a_t, g_t)$. At each episode of training, a goal is sampled from the distribution of goals ρ_g and the agent gets rewarded for reaching the sampled goal. This reward can either be an *extrinsic reward* which is defined as part of the environment or an *intrinsic reward* which is defined as part of the algorithm. In DGRL, we define the intrinsic reward as the fraction of discrete factors which match in the respective representations of the goal observation and the state observation. At test time, the agent can either be evaluated to reach goals within the distribution ρ_g, or for its generalization capability to reach goals not seen during training.

Hierarchical RL. The higher level policy $g_t \sim \pi^h_{\theta_h}(g \mid s_t)$ outputs a continuous representation of goals g by conditioning on the states every K time-steps, it can also output a sub-goal s_g by conditioning on both states s and environment goals g, i.e., $\pi^h_{\theta_h}(s_g \mid s_t, g_t)$. The effectiveness of goal-conditioned HRL relies on the specification of semantically meaningful sub-goals. Learned codebooks (section 3.2) consisting of a set of discrete codes can be used by a higher level policy to *specify* which goal to reach to a lower level policy. The use of learned codebooks ensures that the goal specified by the higher level policy is grounded in the space of raw-observations.

In section 5, we empirically show the benefits of the proposed approach for training goal-reaching policies or goal-conditioned value functions, as well as in a goal-conditioned hierarchical RL setup.

4 Theoretical Analysis

In this section, the goal discretization is shown to improve generalization to novel goals by enhancing the concentration of the goal distribution within each neighborhood of discretized goal values; i.e., by decomposing the goal probability $p(g)$ into $p(g) = \sum_k p(g|g \in \mathcal{G}_k)p(g \in \mathcal{G}_k)$ with the neighborhood set $\{\mathcal{G}_k\}_k$, it improves the overall performance in $p(g)$ by increasing the concentration in $p(g|g \in \mathcal{G}_k)$. Intuitively, this is because the discretization removes varieties of possible goal values $g \in \mathcal{G}_k$ for each neighborhood $\mathcal{G}_k$. To state our result, we define $\varphi_\theta(g) = \mathbb{E}_{s_0}[V^\pi(s_0, g)]$, where $\theta \in \mathbb{R}^m$ is the vector containing model parameters learned through goals observed during training phase, $g_1, \ldots, g_n$. We denote the discretization of goal g by $q(g)$, and the identity function by id as $\mathrm{id}(x) = x$. Let $\mathcal{Q} = \{q(g) : g \in \mathcal{G}\}$ and $\hat{d}$ be a distance function. We use $\mathcal{Q}_i$ to denote the i-th element of $\mathcal{Q}$ (by ordering elements of $\mathcal{Q}$ with an arbitrary ordering). We also define $[n] = \{1, \ldots, n\}$, $\mathcal{G}_k = \{g \in \mathcal{G} : k = \arg\min_{i \in [|\mathcal{Q}|]} \hat{d}(q(g), \mathcal{Q}_i)\}$, $\mathcal{I}_k = \{i \in [n] : g_i \in \mathcal{G}_k\}$, and $\mathcal{I}_Q = \{k \in [|\mathcal{Q}|] : |\mathcal{I}_k| \geq 1\}$. We denote by c a constant in $(n, \theta, \Theta, \delta, S)$.

The following theorem (proof in Appendix E) shows that the goal discretization improves the lower bound of the expected sum of rewards for unseen goals $\mathbb{E}_{g \sim \rho_g}[(\varphi_\theta \circ \varsigma)(g))]$ by the margin of $\omega(\theta)$:

Theorem 1. *For any $\delta > 0$, with probability at least $1-\delta$, the following holds for any $\theta \in \mathbb{R}^m$ and $\varsigma \in \{\mathrm{id}, q\}$:*

$$\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g))] \geq \frac{1}{n}\sum_{i=1}^{n}(\varphi_\theta \circ \varsigma)(g_i) - c\sqrt{\frac{2\ln(2/\delta)}{n}} - \mathbb{1}\{\varsigma = \mathrm{id}\}\omega(\theta)$$

where $\omega(\theta) = \frac{1}{n}\sum_{k\in\mathcal{I}_Q}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\varphi_\theta(g_i) - \mathbb{E}_{g\sim\rho_g}[\varphi_\theta(g)|g \in \mathcal{G}_k]\right)$. Moreover, for any compact $\Theta \subset \mathbb{R}^m$, if $\varphi_\theta(g)$ is continuous at each $\theta \in \Theta$ for almost all g and is dominated by a function χ as $|\varphi_\theta(g)| \leq \chi(g)$ for all $\theta \in \Theta$ with $\mathbb{E}_g[\chi(g)] < \infty$, then the following holds:

$$\sup_{\theta\in\Theta}|\omega(\theta)| \xrightarrow{P} 0 \quad \textit{when} \quad n \to \infty.$$

Proof. Detailed proof provided in the appendix E □

Without the goal discretization, we incur an extra cost of $\omega(\theta)$, which is expected to be strictly positive since $\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\varphi_\theta(g_i)$ is maximized during training while $\mathbb{E}_{g\sim\rho_g}[\varphi_\theta(g)|g \in \mathcal{G}_k]$ is not. Thus, the goal discretization can improve the expected sum of rewards for unseen goals by the degree of $\omega(\theta)$, which measures the concentration of the goal distribution in each neighborhood. This extra cost $\omega(\theta)$ goes to zero when the number of goal observations n approaches infinity.

5 Experiments

The main goal of our experiments is to show that goal discretization can lead to sample efficient learning and generalization to novel goals, in goal-conditioned RL. First, we directly study this by training on environments with a set of goals (such as 8 positions within a gridworld) and then evaluating the agent's ability to reach a position within the gridworld which it was not tasked with reaching during training. Second, we consider hierarchical goal-conditioned RL, in which a higher-level agent generates goals that a lower-level agent is tasked with reaching. In this case, the task of reaching novel goals occurs *organically* as the higher-level model selects new goals. This setup also shows the advantages of DGRL for *goal specification*. A secondary goal of our experiments is to show that using many discrete factors is often critical for optimal performance, which proves the value of *compositionality* in *grounding goals*.

We evaluate our proposed method DGRL by integrating it into existing state-of-the-art goal-conditioned and hierarchical RL tasks. Experimentally, we analyse DGRL on several challenging goal-conditioned testbeds that have previously been used in the goal-conditioned RL community. DGRL in principle can be applied to any existing downstream goal-conditioned RL tasks, we demonstrate improvements on five distinct goal-conditioned RL tasks. We considere maze navigation tasks where images are used as observations and we show improved generalization to novel goals. We integrate DGRL to an existing goal-conditioned baseline for navigating procedurally-generated hard exploration Minigrid environments [8] and find that it outperforms state-of-the-art exploration baselines. We also show improvements with DGRL on continuous control (Ant) navigation and manipulation tasks, where goals come from a high-level controller. Finally, we show that discrete representations also significantly improve sample efficient learning on a challenging vision-based robotic manipulation environment.

Learning to Reach Diverse and Novel Goals. We study a gridworld navigation task in which an agent is trained to reach a goal from a small finite set of training goals, and during evaluation is tasked with reaching a novel goal unseen during training. This is a navigation task with a pixel-level observation space showing the position of the agent and the goal in a gridworld. We consider two mazes spiral and single-loop topology. Experiment setup given in Appendix C.2.

For this task, we train a goal-conditioned Deep Q-Learning (DQN) agent, and use a pre-trained representation $\phi(\cdot)$ where the encoder is trained using data from a random rollout policy. Because the gridworld is small the random rollout policy achieves good coverage of the state space, so we found this was sufficient for learning a good goal representation. At each episode, a specific goal is randomly sampled from a distribution of goals, and the DQN agent is trained to reach the specified goal for that episode. During evaluation, we test the learned agent on goals either from the training distribution, or not seen during training.

Furthermore, for this task, we additionally use an intrinsic reward for exploration of the goal-DQN agent. Since we learn a discrete compositional representation of the goal, we compute an exploration bonus based on the discrete latent codebooks; i.e., we embed the states and goals using the learned codes and then compute an intrinsic exploration bonus based on the fraction of learned factors that match. For the baseline goal-DQN agent, we provide an additional reward bonus based on the cosine distance between continuous embeddings of the state observation and goal. Figure 3 shows that DGRL significantly outperform a continuous baseline goal DQN agent, when trained on either four goals or eight goals. We evaluate generalization to 4 novel goals unseen during training (Figure 4) and demonstrate improved generalization to novel goals.

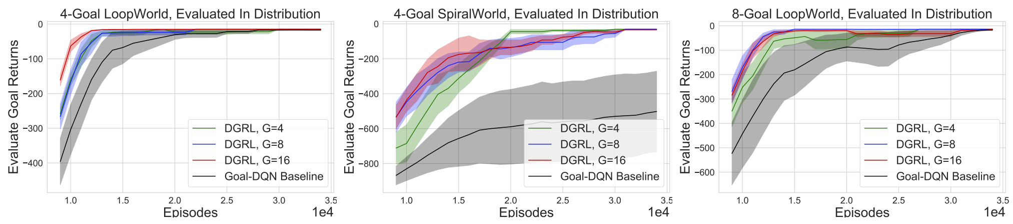

Figure 3: Loopworld maze environment. We show that for different discrete factors of $4, 8, 16$, DGRL outperforms a goal-DQN baseline agent with continuous goal representations. As we increase the number of factors G to 16, the expressivity of the discrete goal representation increases, lowering the odds of the factors being the same. This provides a better intrinsic reward signal for exploration, resulting in faster convergence for DGRL integrated on a goal-DQN agent.

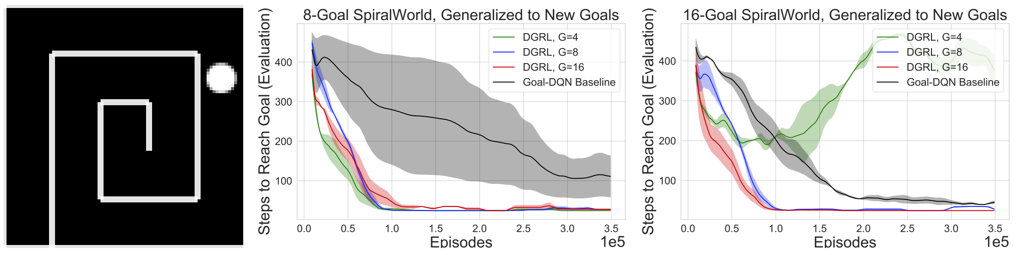

Figure 4: SpiralWorld environment (left). Generalization to test distribution of 4-goals in a SpiralWorld environment (left). We show the total number of steps to solve all test set goals, when trained on either an 8-goal or 16-goal training distribution.

In the previous experiment, we evaluated the generalization ability of DGRL by showing that learning discrete compositional representations of goals can improve generalization to novel goals. Now, we consider various setups in which a goal generating agent specifies goals using the learned codebook and the goal-conditioned agent is tasked with reaching the goals specified by the goal generating agent. We test various settings, where the goal generating agents is parameterized as an adversarial teacher [4], or as a higher-level policy in the case of hierarchical RL.

Procedurally Generated MiniGrid Exploration Task. We follow the experimental setup of [4] and [38] and evaluate DGRL on procedurally generated MiniGrid environments [8]. In [4], a goal-generating teacher proposes goals to train a goal-conditioned "student" policy. We integrate DGRL on top of AMIGO [4] and compare DGRL on a hard exploration task with state-of-the-art exploration baselines. Experimental results are summarized in Table 1 and more details provided in Appendix C.3. Note that unlike RIDE and RND, we do not provide an additional exploration bonus to DGRL, and find that DGRL can still solve this hard exploration task more efficiently.

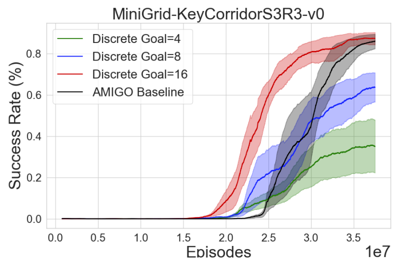

Figure 5: Performance comparison of the Amigo baseline [4] with adversarially intrinsic goals, and adding DGRL for discretization of the goals.

Model	KCmedium
AMIGO + DGRL, G=16	.96 ± .01
AMIGO + DGRL, G=8	.70 ± .16
AMIGO	.93 ± .06
RIDE	.90 ± .00
RND	.89 ± .00
ICM	.42 ± .21

Table 1: We added DGRL on top of the Amigo baseline implementation provided by the authors.

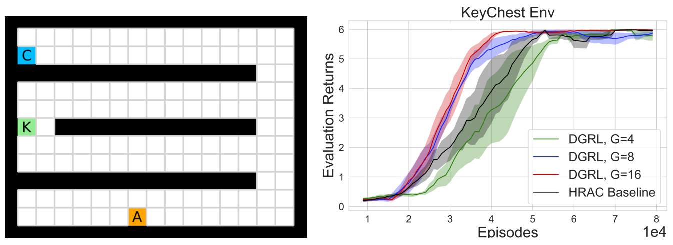

Figure 6: In KeyChest, the agent (A) starts from a random stochastic position, picks up the key (K) and then uses the key to open the chest (C). We find DGRL improved sample efficiency over the HRAC baseline.

Goal Grounding in KeyChest Maze Navigation Domain. We consider a simple discrete state action KeyChest maze navigation task, following [58], where discrete goals in the state space are provided by a higher level policy. For this task, to integrate DGRL, we learn an embedding $\phi(\cdot)$ of the goals, then discretize the representation with a learned codebook. We compare with a baseline HRAC [58] agent (details in Appendix C.1). Figure 6 shows an illustration of the KeyChest environment and a performance comparison of DGRL with different group factors G. Using fewer factors ($G = 4$) performs worse than the HRAC baseline, whereas using a larger number of factors ($G = 8$ or $G = 16$) improves the sample efficiency of the goal reaching agent, providing evidence for the benefits of compositionality.

Ant Manipulation Control Domains. We employed DGRL on three different continuous control tasks: AntMazeSparse, AntFall and AntPush task. We emphasize that these tasks are the more challenging counterparts of AntGather and AntMaze tasks, typically used in the hierarchical RL community [30, 32]. Figure 7 provides illustration of these tasks. We evaluate goal discretization by integrating DGRL to the state-of-the-art HRAC baseline. Details of the experimental setup are provided in Appendix. Figure 7 shows that specifying the goals using the learned codebook helps DGRL achieve a higher success rate compared to the HRAC baseline.

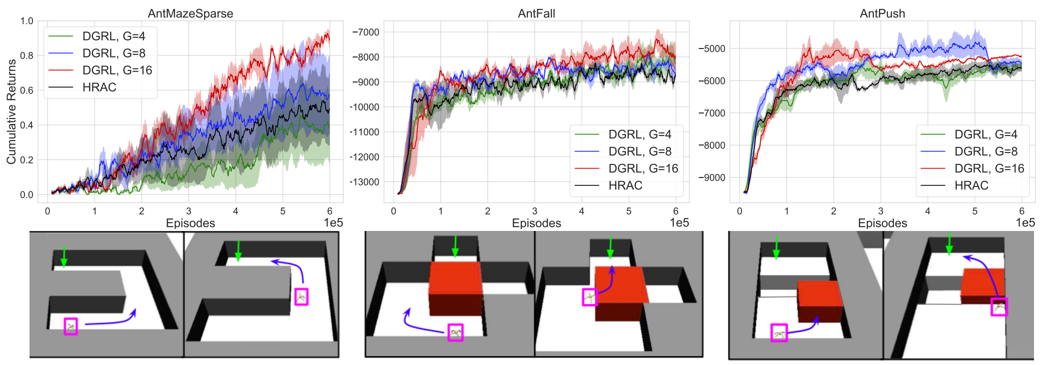

Figure 7: Comparison of DGRL with baseline HRAC [58] on 3 different navigation tasks.

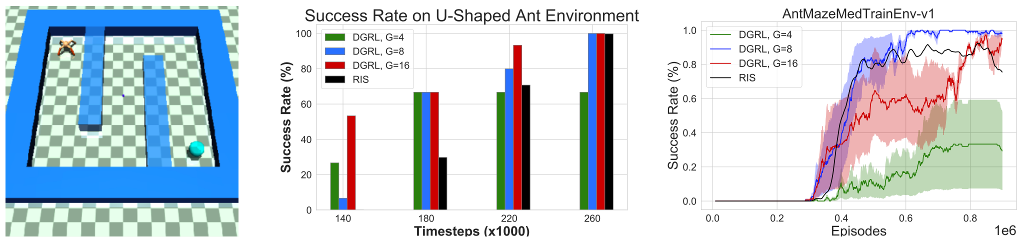

Figure 8: Performance comparison with the success rate of reaching goal positions during evaluation in an extended temporal configuration of the U-shaped and maze-shaped Ant navigation tasks. We find that integrating DGRL with RIS can lead to more sample efficient convergence on these tasks, while baselines such as SAC and TDM (not shown) fail completely on both AntU and AntMaze as reported by [5]. The RIS baseline is based on raw data provided by the authors.

Ant Navigation Maze Tasks. We consider Ant navigation tasks that require extended temporal reasoning, following the setup in Reinforcement learning with Imagined Subgoals [5, RIS]: a U-shaped maze, and an S-shaped maze (the S-shaped maze is shown in Figure 8). The ant navigating in the maze is trained to reach any goal in the environment. The agent is evaluated for generalization in an extended temporal setting with a difficult configuration, we compare the success rate of DGRL integrated on top of RIS with several baselines. We emphasize the difficulty of these tasks, where existing baselines like soft actor critic [16, SAC] and temporal difference models [37, TDM] fail completely. Results in Figure 8 show that DGRL improves the sample efficiency over the RIS baseline. Additional experimental setup and environment configurations are provided in Appendix C.4.

Vision Based Robotic Manipulation. Finally, we assess DGRL on a hard vision-based robotic manipulation task, and use the same setup as in section 5 to integrate DGRL with the state-of-the-art RIS baseline on the Sawyer task in Figure 9. This manipulation task is adapted from [34], where the baseline RIS is already shown to be superior to previous goal conditioning methods. The task of the agent is to control a 2-DoF robotic arm from image input and move a puck positioned on the table. The Sawyer task is designed for training and generalization. At test time, it evaluates the agent's success at placing the puck in desired positions in a temporally extended configuration. This is a challenging vision-based complex motor task, since test time generalization requires temporally extended reasoning. Results in Figure 9 show that DGRL improves the sample efficiency over the RIS baseline. Details of the experimental setup is provided in Appendix C.5.

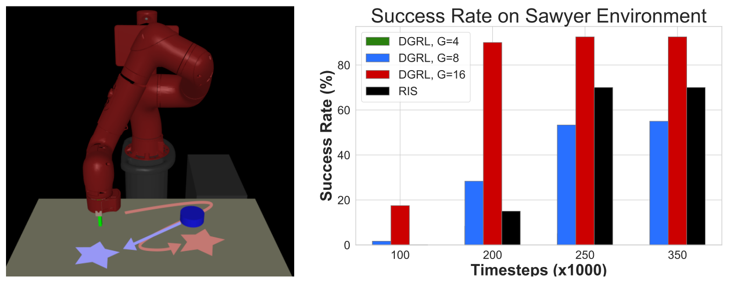

Figure 9: Sawyer Robotic Manipulation Task. Integrating DGRL with RIS (with a larger number of factors, G=8 and G=16, while G=4 fails), improves over the RIS baseline

6 Conclusion

Our work provides direct evidence that performance of goal-conditioned RL can be improved when the representations of the goals are both *discrete* and *compositional*. We show that an instantiation of this idea using multi-factor compositional discretization significantly improves performance on a diverse set of benchmarks. Moreover, DGRL provides especially large improvements in generalization when an agent is tasked with seeking out goals not seen during training. An interesting question that arises from our work is how to theoretically *ground* and *specify* goals, which might be helpful for efficient structured exploration in tasks where goal seeking is crucial.

References

[1] Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding language to autonomously-acquired skills via goal generation. *arXiv preprint arXiv:2006.07185*, 2020.

[2] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

[3] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

[4] Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B. Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[5] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1430–1440. PMLR, 2021.

[6] Crystal Chao, Maya Cakmak, and Andrea L Thomaz. Towards grounding concepts for transfer in goal learning from demonstration. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–6. IEEE, 2011.

[7] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

[8] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

[9] Ugo Dal Lago, Marco Pistore, and Paolo Traverso. Planning with a language for extended goals. In *AAAI/IAAI*, pages 447–454, 2002.

[10] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

[11] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pages 271–278. Morgan Kaufmann, 1992.

[12] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res.*, 13:227–303, 2000.

[13] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[14] Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

[15] Herbert P Grice. Logic and conversation. In *Speech acts*, pages 41–58. Brill, 1975.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[17] Robert I Jennrich. Asymptotic properties of non-linear least squares estimators. *The Annals of Mathematical Statistics*, 40(2):633–643, 1969.

[18] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[19] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer, 1993.

[20] Leslie Pack Kaelbling. Learning to achieve goals. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1094–1099. Morgan Kaufmann, 1993.

[21] Leslie Pack Kaelbling et al. An architecture for intelligent reactive systems. *Reasoning about actions and plans*, pages 395–410, 1987.

[22] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[23] Khimya Khetarpal, Zafarali Ahmed, Gheorghe Comanici, David Abel, and Doina Precup. What can i do here? a theory of affordances in reinforcement learning. In *International Conference on Machine Learning*, pages 5243–5253. PMLR, 2020.

[24] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[25] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[26] Dianbo Liu, Alex M Lamb, Kenji Kawaguchi, Anirudh Goyal, Chen Sun, Michael C Mozer, and Yoshua Bengio. Discrete-valued neural communication. *Advances in Neural Information Processing Systems*, 34, 2021.

[27] Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *Advances in Neural Information Processing Systems*, 33:3686–3698, 2020.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[29] Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130, 1993.

[30] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3307–3317, 2018.

[31] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018.

[32] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *CoRR*, abs/1909.10618, 2019.

[33] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[34] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14814–14825, 2019.

[35] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[36] 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.

[37] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep RL for model-based control. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[38] Roberta Raileanu and Tim Rocktäschel. RIDE: rewarding impact-driven exploration for procedurally-generated environments. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[39] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

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

[41] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1312–1320. JMLR.org, 2015.

[42] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[44] Wolfram Schultz. Predictive reward signal of dopamine neurons. *Journal of neurophysiology*, 80(1):1–27, 1998.

[45] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[46] David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

[47] Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1):123–158, 1996.

[48] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

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

[50] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768, 2011.

[51] Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

[52] Gerald Tesauro. Practical issues in temporal difference learning. *Advances in neural information processing systems*, 4, 1991.

[53] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[54] Aad W. van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes*. Springer New York, 1996.

[55] Vivek Veeriah, Junhyuk Oh, and Satinder Singh. Many-goals reinforcement learning. *CoRR*, abs/1806.09605, 2018.

[56] Marco A. Wiering and Jürgen Schmidhuber. Hq-learning. *Adapt. Behav.*, 6(2):219–246, 1997.

[57] Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12611–12620. PMLR, 2021.

[58] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...

(a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] . Main claims are justified with a set of illustrative examples and experiments; both standard and toy examples included for experiments to justify the key claims.
(b) Did you describe the limitations of your work? [Yes]
(c) Did you discuss any potential negative societal impacts of your work? [N/A] Besides the general impact of machine learning methods we do not foresee and identify any particular and immediate negative societal impact of our work. We do not conduct any experiments that could be in itself harmful for society but understand that the applicability of this method is very broad.
(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] . Code will be provided along with the supplementary materials for reproducibility.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . More details to be included in supplemetary section
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] . All experiments are run across 3 to 5 random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . This will be mentioned in supplementary and acknowledgements
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]