Discrete Factorial 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 factorial 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 [25, 57] 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 [24]. One way to control this behavior is by specifying a reward signal [49, 53]. 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 [59, 60, 33, 34, 54], 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 [19].

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 [22]. 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 [58, 45, 41]. 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 [11, 4, 27]. Moreover, the diversity of goals seen during training should help improve both credit assignment and generalization [45, 40].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

While this framework is promising, it introduces two new challenges: *goal grounding* [7, 1] and *goal specification* [3]. 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 [36, 26]. 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 [11, 4, 27, 21]. 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 [18, 10, 21, 16]. At the same time, connecting language feedback for an agent is non-trivial (requiring special assumptions or a labeling framework) [8].

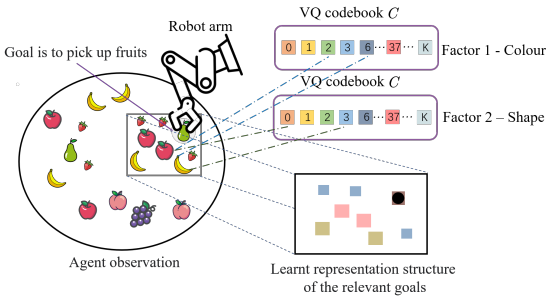


Figure 1: Illustration of learning *discrete* and *factorial* goal representations.

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 *factorial*. To perform this discretization, we use Vector-Quantization [61, 44, 30] 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 factorial 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) [33] or policy optimization based algorithms [47, 48]. 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 factorial representation of discrete latent goals in a series of complex goal-conditioned tasks.

Vector Quantized Representations. VQ-VAE [61, 44, 30] 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 [28, 55, 50, 31] 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 [31] which optimizes for a contrastive objective as a proxy to maximize the mutual information between representations of nearby states in the same trajectory.

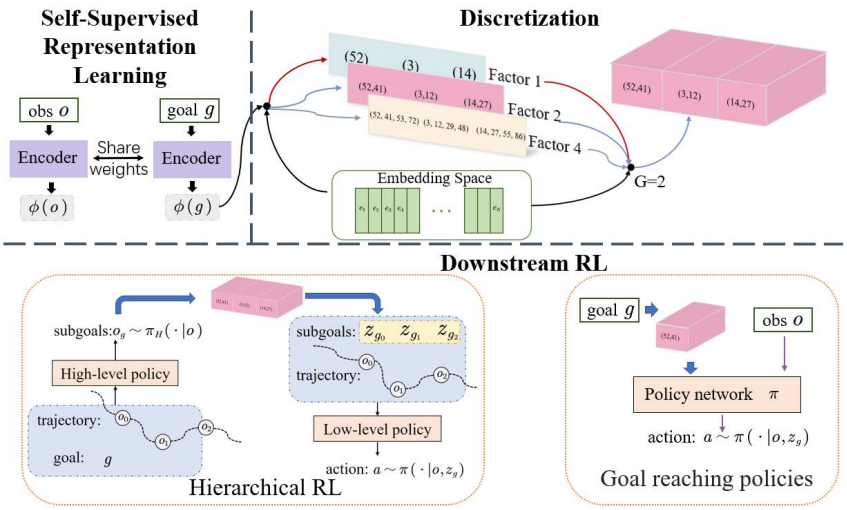


Figure 2: **Summary of Proposed DGRL Model** for improving *goal grounding* and *goal specification* by making goal representations *discrete* and *factorial*. We learn a latent representation for both observations and goals using a self-supervised learning method (sec. 3.1). We convert the learnt latent 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 learn discrete representations by using the vector-quantization method from the VQ-VAE paper [61] and follow the multi-factor setup used in Discrete-Value Neural Communication [30]. 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 m is divisble by G). Each continuous segment c_i is mapped separately to a discretized latent 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:

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

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 [30] and [61]. Here, sg refers to a stop-gradient operation that blocks gradients from flowing into e_{o_i}, and β is a hyperparameter which controls how strongly we move the codes toward the encoded values. Unlike [30], we used a moving average to update the code embeddings rather than learning them directly as parameters. 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 VQ-quantization process 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 n goals observed during training phase, $g_1, \ldots, g_n$. We denote the discretization of g by $q(g)$, and the identity function by id as $\mathrm{id}(g) = g$. 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 D) 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 D □

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 *factorization* 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 consider 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 [9] 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.1.

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 factorial 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.

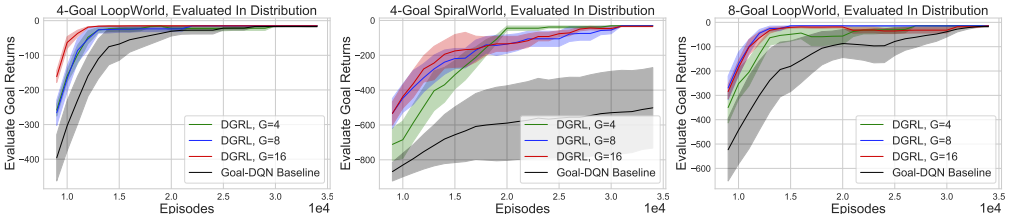


Figure 3: Loopworld maze environment. We show that for different discrete factors of $4, 8, 16$, DGRL outperforms a goal-DQN baseline agent with continuous goal representations. As we increase the number of factors G to 16, the expressivity of the discrete goal representation increases, lowering the 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.

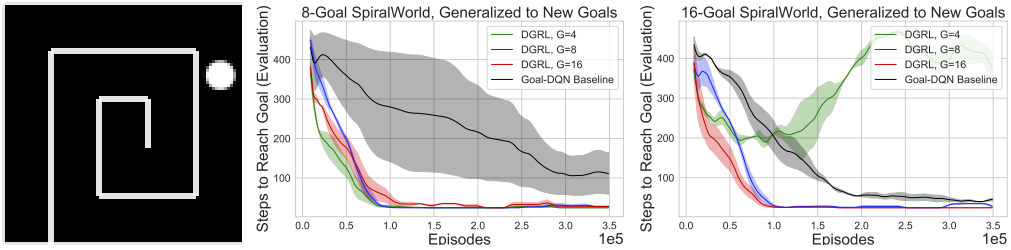


Figure 4: SpiralWorld environment (left). Generalization to test distribution of 4-goals in a SpiralWorld environment (left). We show the total number of steps to solve all test set goals, when trained on either an 8-goal or 16-goal training distribution.

In the previous experiment, we evaluated the generalization ability of DGRL by showing that learning discrete factorial 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 [5], or as a higher-level policy in the case of hierarchical RL.

Procedurally Generated MiniGrid Exploration Task. We follow the experimental setup of [5] and [43] and evaluate DGRL on procedurally generated MiniGrid environments [9]. In [5], a goal-generating teacher proposes goals to train a goal-conditioned "student" policy. We integrate DGRL

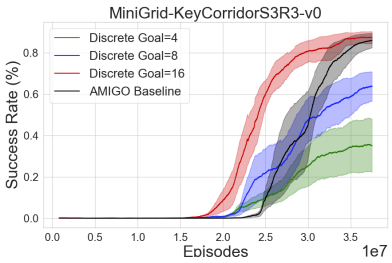


Figure 5: Performance comparison of the Amigo baseline [5] with adversarially intrinsic goals, and adding DGRL for discretization of the goals.

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

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

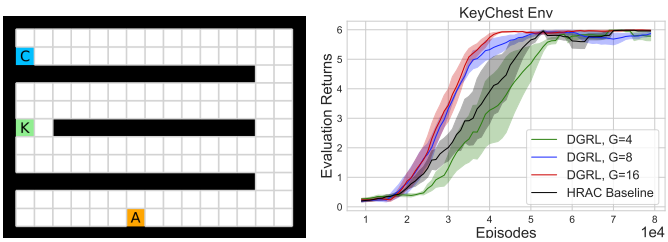


Figure 6: In KeyChest, the agent (A) starts from a random stochastic position, picks up the key (K) and then uses the key to open the chest (C). We find DGRL improved sample efficiency over the HRAC baseline.

on top of AMIGO [5] 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.

Goal Grounding in KeyChest Maze Navigation Domain. We consider a simple discrete state action KeyChest maze navigation task, following [68], 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 [68] agent (details in Appendix C.2). 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 factorization.

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 [35, 37]. 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 5. Figure 7 shows that specifying the goals using the learned codebook helps DGRL achieve a higher success rate compared to the HRAC baseline.

Ant Navigation Maze Tasks. We consider Ant navigation tasks that require extended temporal reasoning, following the setup in Reinforcement learning with Imagined Subgoals [6, 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 [19, SAC] and temporal difference models [42, 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.5.

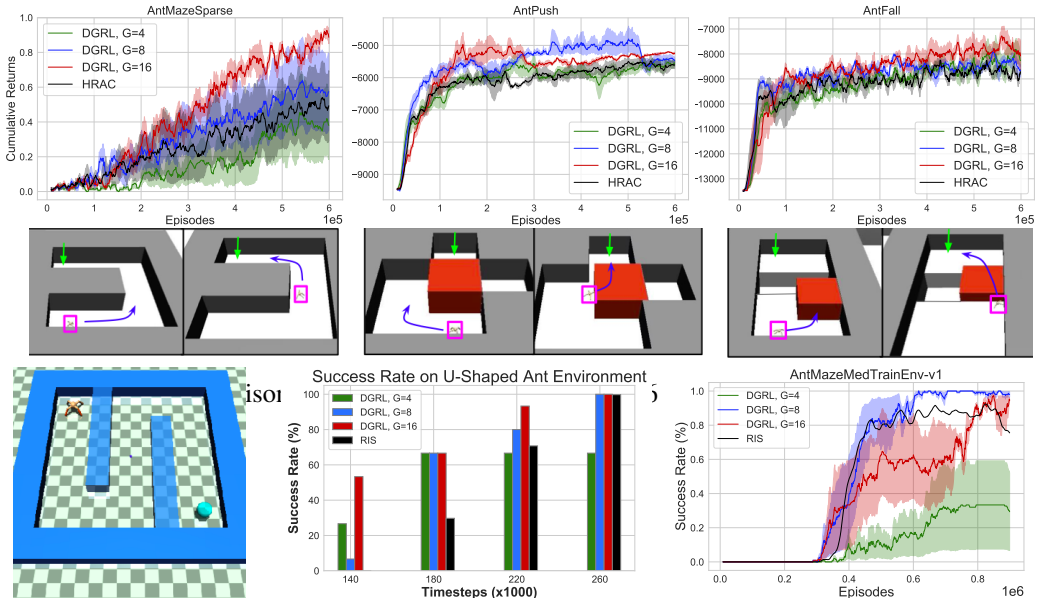


Figure 8: Performance comparison with the success rate of reaching goal positions during evaluation in an extended temporal configuration of the U-shaped and maze-shaped Ant navigation tasks. We find that integrating DGRL with RIS can lead to more sample efficient convergence on these tasks, while baselines such as SAC and TDM (not shown) fail completely on both AntU and AntMaze as reported by [6]. The RIS baseline is based on raw data provided by the authors.

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 [39], 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.6.

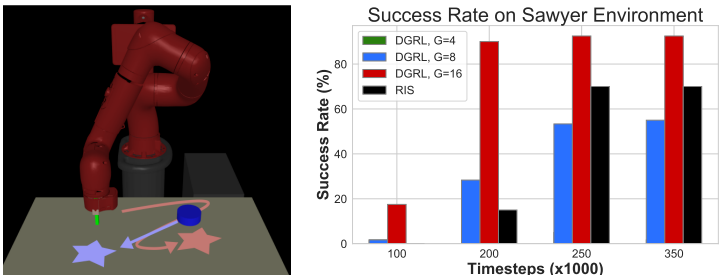


Figure 9: Sawyer Robotic Manipulation Task. Integrating DGRL with RIS (with a larger number of factors, G=8 and G=16, while G=4 fails), improves over the RIS baseline

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 *factorial*. We show that an instantiation of this idea using multi-factor discretization significantly improves performance on a diverse set of benchmarks. 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] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R. Devon Hjelm. Unsupervised state representation learning in atari. 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 8766–8779, 2019.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

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

[12] 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.

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

[14] Zach Dwiel, Madhavun Candadai, Mariano J. Phielipp, and Arjun K. Bansal. Hierarchical policy learning is sensitive to goal space design. *CoRR*, abs/1905.01537, 2019.

[15] 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.

[16] 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.

[17] Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew M. Botvinick, Yoshua Bengio, and Sergey Levine. Infobot: Transfer and exploration via the information bottleneck. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

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

[19] 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.

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

[21] 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.

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

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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.

[29] Andrew Levy, George Dimitri Konidaris, Robert Platt Jr., and Kate Saenko. Learning multi-level hierarchies with hindsight. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[30] 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.

[31] 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.

[32] Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6961–6971. PMLR, 2020.

[33] 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.

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

[35] 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.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

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

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

[50] 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.

[51] Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[52] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R. Devon Hjelm, Philip Bachman, and Aaron C. Courville. Pretraining representations for data-efficient reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12686–12699, 2021.

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

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

[55] 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.

[56] Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. Learning goal embeddings via self-play for hierarchical reinforcement learning. *CoRR*, abs/1811.09083, 2018.

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

[58] 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.

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

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

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

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

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

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

[65] Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. 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 11784–11794. PMLR, 2021.

[66] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. 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 11920–11931. PMLR, 2021.

[67] 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.

[68] 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]

A Appendix

In section B, we discuss extended related work. In section C we include details of our experiment setup and additional ablation analysis evaluating DGRL. Section D contains detailed proof for theorem 1. Finally, we provide our algorithm, DGRL that requires minimal changes to existing goal conditioned RL setup, in section E, with sample code snippet provided in F

B Related Work

Learning with multiple hierarchies has long been proposed in the RL literature, where goal conditioned HRL uses high level planning and low level control using sub-goals. Often in goal conditioned HRL, the higher level policy specifies goals for the lower level control, which may not have good *specification* and *grounding*. Several prior works focus on goal-conditioned RL to improve sample efficiency in deep RL tasks [35], [37], although this has been proposed years back to solve long horizon tasks by hierarchical RL specifying goals [23] [12] [13] [64]. The goal is to learn to solve sub-goals provided to the policy, by learning to predict a sequence of actions that can learn to reach each of the sub-goals [63], [46], [38]. Additionally, in existing HRL literature, often distance measures are used based on goal-conditioned value functions, to measure distances between states and the sequence of sub-goals to reach [15, 67], often for planning [39]. Since the space of specified goals in the state space can be arbitrary, often an additional constraint is also learnt for valid distribution of goals within a constrained space that the lower level policy can reach [68]. We tackle this problem by proposing DGRL, for better *grounding* and *specification* of sub-goal representations.

In previous works, often mutual information based objectives are proposed for goal conditioned RL, which addresses goal based representation learning, such as t improve stability of training goal conditioned value functions [35, 37], or providing goal representations such as to identify decision states for better exploration [17]. However, for most of these settings, the sub-goals are based on an external reward with no specification of sub-goals, which can lead to inefficient training. [29, 68] has proposed approaches for penalizing the high level controller for generating sub-goals that may be too difficult for the lower level policies to reach, through the use of additional constrained objectives [68]. We highlight that our proposed DGRL can be generically applied to any goal conditioned RL literature for better *grounding* and *specification* of the sub-goals. Furthermore, recent work has shown significance of learning representations through self supervised objectives in RL [2, 51], often as a pre-training phase [52, 65], which can help for both exploration [32] and control [66].

In the context of goal conditioned RL, this can be challenging problem as it additionally requires learning reliable representations of goals too, purely from high dimensional observations [14]. Previous works [29] have often used the entire observation space as goals, which is not scalable for complex tasks. Other works have used pre-defined space of sub-goals as domain knowledge [35], or use self-play for sub-goal representations [56] to reduce the complexity of goal space design. Most recently, [39, 38] utilized unsupervised representation learning to learn a goal representation space, which can further be used for planning and control. In this work, we show additionally that using a bottleneck can further lead to factorial representation of goals, while helping with goal specification via learning latent space of discrete goals, that can be used for planning and control. We emphasize that DGRL can be integrated on any existing goal conditioned approach that utilizes learning a sub-goal representation.

C Experiment Details and Additional Results

For all our experiments, we use existing open-source implementations of the baseline algorithms. We mostly integrate DGRL with HRAC [68] and RIS [6], which are state of the art goal conditioned algorithms on the complex Ant navigation and robotic manipulation tasks. We use the same hyperparameters and default configurations as used in the baselines. For the simpler maze tasks such as from MiniGrid and KeyChest, we also use existing setups used by previous algorithms [5, 68].

All our experiment results are based on 3 random seeds, and we provide our implementation for reproducibility.

C.1 Visual MazeWorlds

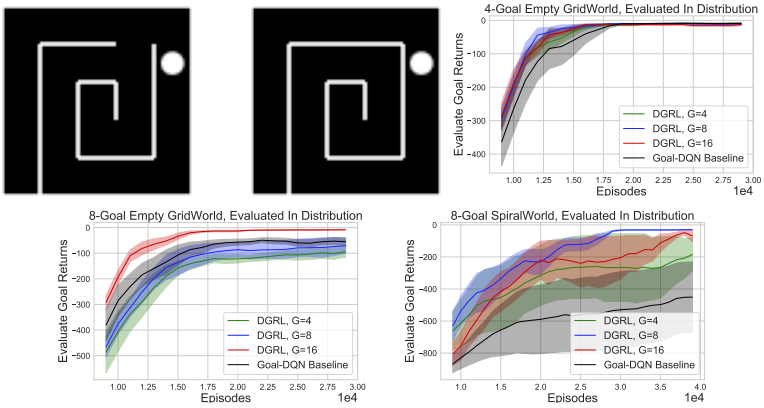


Figure 10: Visualization of the SpiralWorld and LoopWorld environments. Additional experimental results in an empty 4-goal and 8-goal environment across different factors G.

In this section, we provide additional details for the visual maze tasks. We provide environment visualizations, for the spiralworld and loopworld environments in figure 10. For this task, we use 6×6 gridworlds, where the agent receives pixel based full observations, of size $80 \times 80 \times 3$, of the environment. The agent starts deterministically in the environment from one corner of the loop, and the goals are provided in the state space, where agent receives observations of goals. The agent receives a negative reward of -1 at every time step, and a reward of $+5$ when reaching any of the goals.

We train a goal conditioned Q learning agent, with epsilon greedy exploration, where the agent receives goal observations in addition to state observations. We use a simple 2 layer architecture for the value function, which additionally conditions on the goals. In our experiments, we use 4-goals and 8-goals environment, where the agent is trained to reach each of the goals, sampled at every episode. For pixel based observations, we learn a representation ϕ, which can be trained via any self-supervised representation learning objective. For our experiments, we use the Deep InfoMax algorithm [31] for training the representation learner ϕ. Since the environment is quite simple, we found that even pre-training ϕ with random rollout policies is often good enough for training the value functions, as suggested by our experimental results. For DGRL, we additionally use a discrete bottleneck on top of the embedding, and experimentally show that DGRL can significanlty outperform a baseline goal DQN agent.

C.2 KeyChest Domain

For prototype and motivation of DGRL, we also used a simple discrete state and action KeyChest environment, as shown in figure 6. This task is inspired from the HRAC algorithm [68], where the agent starts stochastically in the environment, and the goal is to pick up a key and open a chest. We follow a HRL setup in this task, where the higher level policy provides discrete goals in the state space for the lower level policy to reach. Since the environment has injected stochasticity, such a task requires both low-level control conditioned on the goals, provided by a higher level planner.

The environment has a 3-dimensional state space where the first two represent the position of the agent, while the third dimension represents whether the agent has picked up the key or not. The reward function is sparse, such that the agent only receives a reward $+1$ for picking up the key, and a reward of $+5$ if it can open the chest; otherwise zero rewards elsewhere. This hints to a hard exploration task, where specification of goal plays a key role. The lower level policy receives a

goal reaching reward only, upon reaching the goal state. The higher level policy receives rewards from the environment directly. For the goal reaching reward, we use a standard Euclidean distance between states and goals, and the goal is achieved by the lower level policy if this distance is below a threshold of 0.5.

For DGRL, we apply a discrete bottleneck by first learning a representation of the discrete goal in state space, using an encoder. We then apply a discrete bottleneck on the learnt embedding, followed by a decoder that maps the learnt discrete goal embedding back to the original state space. For baseline [68], the algorithm uses the raw goal states provided by the higher level policy. Both the higher and lower level policies are learnt with actor-critic algorithms. The high level policy is trained based on the task reward, whereas the lower level policy is rewarded for reaching the goals, or nearby regions of the goals, provided by the higher level policy.

C.3 MiniGrid Environments

For our experiments, we follow the setup of [5] and [43] and evaluate DGRL on a simple procedurally generated MiniGrid environment [9]. The minigrid environments are a suite of hard exploration testbeds in RL where the task is designed such that exploration and representation of the visual observations can be disentangled. Following [5], we use the **KCharder**, KeyCorrS4R3 environment, which requires finding a key that can unlock a door which blocks a room. This door needs to be opened by the agent so as to reach the goal. In our experiments, we use the same learning rates, network architecture and other hyperparameters as in the AMIGO paper [5]. We use the open-sourced implementation provided by the authors (for more details, see [5]), and simply integrate DGRL on top of the AMIGO baseline, learning a factorial representation for the goal observations.

C.4 Ant Manipulation tasks

We employ an ant robot with a continuous 8-dimensional action space for all three Ant manipulation tasks. Each episode terminates at 500 time steps in all three tasks.

AntMazeSparse AntMazeSparse is a challenging navigation task with sparse rewards. This environment has a continuous state space including current position, velocity, target location and the current timestep t. The agent is provided with a sparse reward by $+1$ only if the Euclidean distance between the agent and the target location is smaller than 1, where the target location is set at $(2.0, 9.0)$ in the center corridor.

AntPush This environment has the same state space as AntMazeSparse task. The difference is that this environment has a movable block which the agent can interact with. To successfully reach the target position $(0.0, 19.0)$, the agent must push the large block to the side to clear the path to the target location. The success of the agent is defined as having the Euclidean distance of 5 from the target position.

AntFall This environment extends the navigation to three dimensions. Similar to the AntPush task, the environment still has a movable block, while the agent must move the block into a chasm instead of pushing it aside, so that it may walk over it without falling to reach the target position. The target position is fixed to $(0.0, 27.0, 4.5)$ in this environment.

For our experiments, we use the setup provided by the HRAC baseline [68]. In HRAC, an additional adjacency constrained is trained, along with the lower and higher level policies, such that the goals provided by higher controller are within a constrained region of the state space, that can be reachable by the lower level policy. The higher and lower level policies are both trained based on actor-critic algorithms, with separate replay buffers for each policy. The replay buffer for the higher level policy stores one every K transitions.

We integrate DGRL on top of the HRAC baseline setup, where we apply the discrete goal bottleneck based on the output embeddings from higher level policies. Compared to HRAC, DGRL with HRAC would condition the lower level policies on the discrete embeddings of goals.

Figure 11 provides results for all the Ant manipulation tasks. Here we provide the results obtained for all the 5 different Ant environments, and we experiment for a range of discrete factors from 2 to 32. We find that while the best performing factor G is not consistent for all tasks, in general factors of 4, 8 and 16 typically outperform 2 and 32 factors.

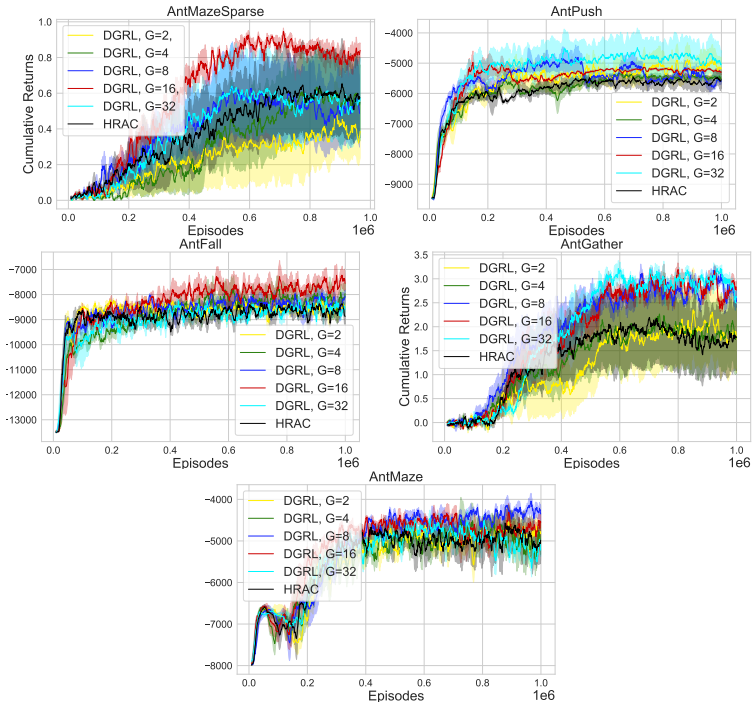


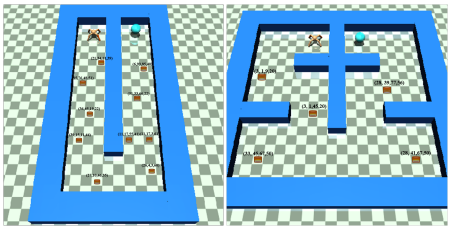
Figure 11: Ablation analysis on the Ant tasks. Standard goal conditioned HRL tasks, AntGather (shown on bottom left) and AntMaze, showing performance comparison of DGRL with baseline HRAC [68] across 3 random environment seed. We include comparisons with all factors G for all the Ant manipulation environments that we expeirmented on, given computation budget.

C.5 Ant Navigation Tasks

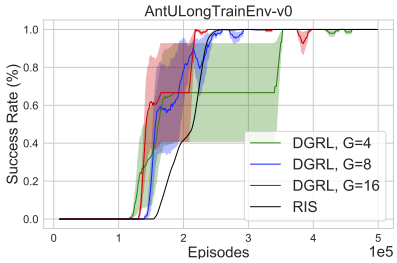
Following the setup in RIS [6], we employ an ant robot with a continuous 8-dimensional action space for all four Ant navigation tasks. reasoning. Figure 12(c) shows an additional result on the U-shaped Ant maze navigation environment. All these environments have a continuous state space including the current position, orientation, the joint angles and the velocities. The goal is considered reached when the Euclidean distance from the target position of the environment is less than 0.5. The agent gets a -1 reward at each time step until the goal is reached. The U-shaped maze has a size of 7.5×18, the S-shaped maze has a size of 12×12, while the π-shaped maze and ω-shaped maze share the same size of 16×16. In the training stage, target and initial position of the agent are sampled randomly at the beginning of each episode. At evaluation, the initial state and the target position are fixed, as illustrated in Figure 12, to test the performance of the agent on challenging configurations that require temporally extended

C.6 Robotic Manipulation Task

We consider the visual robot manipulation task Sawyer, from the multiworld environments of [39]. Our setup is based entirely on the experiment details and code provided in the open-source codebase



(a) U-shaped maze (b) Π-shaped maze



(c) Experiment results comparing different factors G for DGRLintegrated with RIS, and compared with a RIS baseline (baseline data provided by authors)

Figure 12: Visualization of Ant navigation environment, which are standard goal conditioned HRL tasks. We integrate DGRL on top of the existing state of the art RIS baseline [6]. Performance comparison of success rate on AntU-shaped environment (right). We compare DGRL with different factors G and the RIS baseline [6] which has already been shown to outperform other goal conditioned baselines such as LEAP [39] on the Ant navigation tasks.

of RIS [6]. The goal of the agent is to operate a 2D position control and manipulation task. The observations of the environment are based on 84×84 RGB image of the environment. DGRL is trained using a discrete bottleneck based on the learnt representation of the pixel based observations of the environment. We compare the Sawyer environment based on an existing RIS [6] baseline which is already shown to outperform other baselines on this task. For more details on the suite of multiworld environments, including the Sawyer manipulation task, see [39]. In this setting, we can additionally check for generalization. At test time, the cumulative returns of the agent are provided when evaluated on a slightly different task, such as a hard configuration where we evaluate the policy and bottleneck based on learnt representations in the given task.

D Proof of Theorem 1

We consider a goal-conditioned Markov decision process, defined by states $s \in \mathcal{S}$, goals $g \in \mathcal{G}$, actions $a \in \mathcal{A}$, a reward function $r(s, a, g)$, a transition dynamics $p(s'|s, a)$, a maximum horizon T, the initial state distribution ρ_0, and the goal distribution ρ_g. The objective in goal-conditioned RL is to obtain a policy $\pi(a|s, g)$ to maximize the expected sum of rewards $\mathbb{E}_{g\sim\rho_g,((s_t,a_t))_{t=1}^T}[\sum_{t=1}^T r(s_t, a_t, g)]$ where the sequence of state-action pairs $((s_t, a_t))_{t=1}^T$ is sampled according to $s_0 \sim \rho_0$, $a_t \sim \pi(a_t|s_t, g)$, and $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. To study the phenomenon of the goal observations, we define

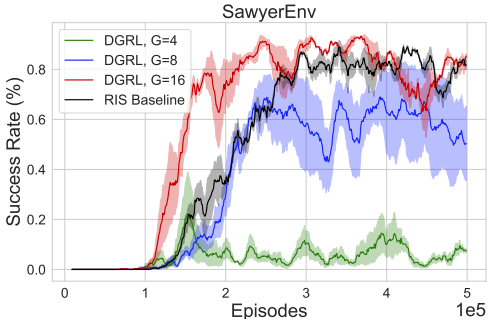


Figure 13: Performance comparison of DGRL with RIS baseline on a complex vision-based robot manipulation task

$$\varphi_\theta(g) = \mathbb{E}_{((s_t,a_t))_{t=1}^T}\left[\sum_{t=1}^T r(s_t, a_t, g)\right],$$

where $\theta \in \mathbb{R}^m$ is the vector containing model parameters learned through goals observed during training phase, $g_1, \ldots, g_n$. Define $\mathbb{1}\{a = b\} = 1$ if $a = b$ and $\mathbb{1}\{a = b\} = 0$ if $a \neq b$. Moreover, $\varphi_\theta \circ \varsigma$ represents the composition of functions φ_θ and ς.

In our proof, we will use the following previous result:

Lemma 1 (Bretagnolle-Huber-Carol inequality)**.** *[62, Proposition A.6.6] If $X_1, \ldots, X_K$ are multinomially distributed with parameters m and $p_1, \ldots, p_K$, then for $\bar{M} > 0$,*

$$\mathbb{P}\left(\sum_{k=1}^{K}\left|\frac{X_k}{m} - p_k\right| \geq \bar{M}\right) \leq 2^K \exp\left(-\frac{m\bar{M}^2}{2}\right).$$

Proof. We decompose 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 $\{\mathcal{G}_k\}_k$ and analyze the concentrations of random variables in terms of $p(g|g \in \mathcal{G}_k)$ and $p(g \in \mathcal{G}_k)$. Let $\varsigma \in \{\mathrm{id}, q\}$. We define $M = \sup_{g \in \mathcal{G}} \varphi_\theta(g)$. Then,

$$\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g))] = \sum_k \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\Pr(g \in \mathcal{G}_k),$$

Using this, we decompose the difference as

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}(\varphi_\theta \circ \varsigma)(g_i) - \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g))] \qquad (2)\\
&= \sum_k \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\left(\frac{|\mathcal{I}_k|}{n} - \Pr(g \in \mathcal{G}_k)\right)\\
&\quad + \left(\frac{1}{n}\sum_{i=1}^{n}(\varphi_\theta \circ \varsigma)(g_i) - \sum_k \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\frac{|\mathcal{I}_k|}{n}\right).
\end{aligned}$$

Since $\frac{1}{n}\sum_{i=1}^{n}(\varphi_\theta \circ \varsigma)(g_i) = \frac{1}{n}\sum_k \sum_{i \in \mathcal{I}_k}(\varphi_\theta \circ \varsigma)(g_i)$,

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}(\varphi_\theta \circ \varsigma)(g_i) - \sum_k \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\frac{|\mathcal{I}_k|}{n}\\
&= \frac{1}{n}\sum_k |\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i \in \mathcal{I}_k}(\varphi_\theta \circ \varsigma)(g_i) - \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\right).
\end{aligned}$$

Substituting these into equation equation 2 yields

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}(\varphi_\theta \circ \varsigma)(g_i) - \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g))] \qquad (3)\\
&= \sum_k \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\left(\frac{|\mathcal{I}_k|}{n} - \Pr(g \in \mathcal{G}_k)\right)\\
&\quad + \frac{1}{n}\sum_k |\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i \in \mathcal{I}_k}(\varphi_\theta \circ \varsigma)(g_i) - \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\right)\\
&\leq M\sum_k \left|\frac{|\mathcal{I}_k|}{n} - \Pr(z \in \mathcal{G}_k)\right| + \frac{1}{n}\sum_k |\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i \in \mathcal{I}_k}(\varphi_\theta \circ \varsigma)(g_i) - \mathbb{E}_{g\sim\rho_g}[(\varphi_\theta \circ \varsigma)(g)|g \in \mathcal{G}_k]\right)
\end{aligned}$$

By using Lemma 1 by setting $\delta = 2^K \exp\left(-\frac{m\bar{M}^2}{2}\right)$ and solving for M, we have that for any $\delta > 0$, with probability at least $1 - \delta$, $\sum_k \left|\frac{|\mathcal{I}_k|}{n} - \Pr(z \in \mathcal{G}_k)\right| \leq \sqrt{\frac{2\ln(2^{|\mathcal{Q}|}/\delta)}{n}} \leq \sqrt{\frac{2|\mathcal{Q}|\ln(2/\delta)}{n}}$, where the last inequality follows from the fact that $1/\delta \leq 1/\delta^{|Q|}$ as $\delta \in (0, 1)$ and $|Q| \geq 1$. Here, notice that the term of $\sum_k \left|\frac{|\mathcal{I}_k|}{n} - \Pr(z \in \mathcal{G}_k^y)\right|$ does not depend on θ. Moreover, note that for any (f, h, M) such that $M > 0$ and $B \geq 0$ for all X, we have that $\mathbb{P}(f(X) \geq M) \geq \mathbb{P}(f(X) > M) \geq \mathbb{P}(Bf(X) + h(X) > BM + h(X))$, where the probability is with respect to the randomness of X. Thus, by combining this and equation equation 3, we have that for any $\delta > 0$, with probability at

least $1-\delta$, the following holds for all θ,

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}(\varphi_\theta\circ\varsigma)(g_i)-\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta\circ\varsigma)(g))] \qquad (4)\\
&\le \frac{1}{n}\sum_{k=1}^{|\mathcal{Q}|}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}(\varphi_\theta\circ\varsigma)(g_i)-\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta\circ\varsigma)(g)|g\in\mathcal{G}_k]\right)+c\sqrt{\frac{2\ln(2/\delta)}{n}}\\
&=\frac{1}{n}\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}(\varphi_\theta\circ\varsigma)(g_i)-\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta\circ\varsigma)(g)|g\in\mathcal{G}_k]\right)+c\sqrt{\frac{2\ln(2/\delta)}{n}}
\end{aligned}$$

If $\varsigma=\mathrm{id}$, then

$$\begin{aligned}
&\frac{1}{n}\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}(\varphi_\theta\circ\varsigma)(g_i)-\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta\circ\varsigma)(g)|g\in\mathcal{G}_k]\right)\\
&=\frac{1}{n}\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\varphi_\theta(g_i)-\mathbb{E}_{g\sim\rho_g}[\varphi_\theta(g)|g\in\mathcal{G}_k]\right)=\omega(\theta).
\end{aligned}$$

If $\varsigma=q$, then

$$\begin{aligned}
&\frac{1}{n}\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}(\varphi_\theta\circ\varsigma)(g_i)-\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta\circ\varsigma)(g)|g\in\mathcal{G}_k]\right)\\
&=\frac{1}{n}\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}(\varphi_\theta\circ q)(g_i)-\mathbb{E}_{g\sim\rho_g}[(\varphi_\theta\circ q)(g)|g\in\mathcal{G}_k]\right)\\
&=\frac{1}{n}\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|\left(\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\varphi_\theta(\mathcal{Q}_k)-\varphi_\theta(\mathcal{Q}_k)\right)=0
\end{aligned}$$

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

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

Define $u_n(S,\theta)=\max_{k\in\mathcal{I}_\mathcal{Q}}\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\varphi_\theta(g_i)-\mathbb{E}_{g\sim\rho_g}[\varphi_\theta(g)|g\in\mathcal{G}_k]$. Then, since $\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|=n$,

$$\omega(\theta)\le u_n(S,\theta)\frac{1}{n}\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k|=u_n(S,\theta).$$

For each $k\in[|\mathcal{Q}|]$, if $p(g\in\mathcal{G}_k)=0$, then the probability of the event of $|\mathcal{I}_k|\ge1$ is zero. Thus, by taking union bounds, with probability one, for all $k\in\mathcal{I}_\mathcal{Q}$, $|\mathcal{I}_k|\to\infty$ as $n\to\infty$. Therefore, by using the uniform law of large numbers (Theorem 2 of [20]) and union bounds over $k\in\mathcal{I}_\mathcal{Q}$ (noticing that $|\mathcal{I}_\mathcal{Q}|\le|\mathcal{Q}|$ is finite), we have that $\sup_{\theta\in\Theta}|u_n(S,\theta)|\xrightarrow{P}0$ when $n\to\infty$. Thus,

$$0\le\sup_{\theta\in\Theta}|\omega(\theta)|\le\sup_{\theta\in\Theta}|u_n(S,\theta)|\xrightarrow{P}0\quad\text{when}\quad n\to\infty.$$

□

E Algorithm

We present the entire algorithm of DGRL built on top of RIS here.

Algorithm 1: RIS with DGRL (changes to RIS in blue)

Initialize replay buffer D
Initialize Q_ϕ, π_θ, π^H_ψ
for k = 1, 2, ... **do**
  Collect experience in D using π_θ in the environment
  Sample batch $(s_t, a_t, r_t, s_{t+1}, g) \sim D$ with HER
  Sample batch of subgoal candidates $z_e \sim D$
  Update Q_ϕ using Policy Evaluation
  Update π^H_ψ using High-Level Policy Improvement
  Output subgoal z_e using $z_e \sim \pi^H_{\psi_{k+1}}(\cdot|s, g)$
  Output discrete goal embedding z_q using Eq. 1 *(Discretization module)*
  Compute prior policy with discrete sub-goal embeddings

$$\pi_k^{\text{prior}}(a \mid s, g) := \mathbb{E}_{z_e \sim \pi^H(.|s,g)}\left[\pi_{\theta'_k}(a \mid s, z_q)\right] \tag{5}$$

  Update π_θ using Policy Improvement with Imagined Subgoals (Eq. 9 in RIS [6])
  Update discretization module using $\mathcal{L}_{\text{discretization}} = \frac{\beta}{G}\sum_i^G ||c_i - \text{sg}(e_{o_i})||_2^2$
end for

F Code Snippet of DGRL

We show a simple code snippet of algorithm 1 below.

```
class RIS(object):
    def __init__(self):
        discrete_cfg = {'groups': self.args.groups, 'n_embed': self.
    args.n_embed}
        self.vq_layer = VectorQuantizerEMA(state_dim, discrete_cfg['
    n_embed'], discrete_cfg['groups']).to(device)
        params = list(self.critic.parameters()) + list(self.vq_layer.
    parameters())
        self.critic_optimizer = torch.optim.Adam(params, lr=q_lr)

    def select_action(self, state, goal):
        with torch.no_grad():
            state = self.encoder(state)
            goal = self.encoder(goal)
            goal_embed, _, goal_ind = self.vq_layer(goal)
            action, _, _ = self.actor.sample(state, goal_embed)
        return action

    def train(self, state, action, reward, next_state, done, goal,
    subgoal):
        if self.image_env: ## FOR SAWYER ENVS
            state_z = self.encoder(state)
            next_state_z = self.encoder(next_state)
            goal_z = self.encoder(goal)
            subgoal_z = self.mlp_encoder(subgoal)
            encoder_loss = self.driml_loss(state, next_state, action)
            if self.args.use_vq == 'true':
                _, vq_loss_state, _ = self.vq_layer(state_z)
                _, vq_loss_next_state, _ = self.vq_layer(next_state_z)
                _, vq_loss_goal, _ = self.vq_layer(goal_z)
                _, vq_loss_subgoal, _ = self.vq_layer(subgoal_z)
                rep_loss = vq_loss_state + vq_loss_next_state +
    vq_loss_goal + vq_loss_subgoal
            rep_loss += encoder_loss

        else: ## for non-image or non-pixel based envs - ANT ENVS
            state_z = self.mlp_encoder(state)
            next_state_z = self.mlp_encoder(next_state)
            goal_z = self.mlp_encoder(goal)
            subgoal_z = self.mlp_encoder(subgoal)
            encoder_loss = self.autoencoder_loss(state)
            if self.args.use_vq == 'true':
                _, vq_loss_state, _ = self.vq_layer(state_z)
                _, vq_loss_next_state, _ = self.vq_layer(next_state_z)
                _, vq_loss_goal, _ = self.vq_layer(goal_z)
                _, vq_loss_subgoal, _ = self.vq_layer(subgoal_z)
                rep_loss = vq_loss_state + vq_loss_next_state +
    vq_loss_goal + vq_loss_subgoal
            rep_loss +=encoder_loss
```

G Response to All Reviewers

G.1 Generic Response to All Reviewers : Compositional / Factorial Representations

Questions and Comments from reviewers regarding compositionality :

- *Question From Reviewer oX1u : The paper claims that the compositional representation helps encode goals into a semantically meaningful latent space (line 77), for example based on the example in Figure 1, I would expect some segments to correspond to goal colors and some segments to correspond to fruit types..*

- *Comment from Reviewer h8LS : I have a comment about the use of the term "compositionality" in the paper used to refer to the discrete representation, which I elaborate more on in the "Questions" section below. I would urge the authors to remove the term compositional from the title as well. In many places in the manuscript, the discrete representations learned by the codebook are characterized as compositional. I would argue that compositionality in the representations learned using neural networks, though desirable, is something we have not yet achieved as a community. In my opinion, the reason for this is that compositionality requires some kind of syntax in the representation, which needs to be interpreted by something else in a very precise and consistent way. To claim that a representation is compositional, one would have to show that consistent replacements of parts of the representations lead to consistent results in behaviour.*

We sincerely thank the reviewers for the insightful comments about the term compositionality used throughout the paper. We are taking account of this, and going forward, would re-phrase this term to mention DGRL learns ***factorized*** or ***factorial*** representations. The use of the term compositionality was based on some related experimental insights we had when working on this paper, which we omitted previously for sake of brevity and conciseness. We note that the use of the discretization bottleneck indeed provides a semantically meaningful and often indistinguishable latent space, depending on the number of factors. For clarification to reviewers, we provide a set of new demonstrations to show how DGRL learns factorized representations. Additional details for the experiments are provided at the end of the manuscript.

G.1.1 Color-MNIST Example for Compositionality or Factorization

We first give a supervised learning example based on the color-MNIST dataset, where factorization in the learnt latent space is achieved. Figure 14 displays reconstructed images from a trained decoder operating on a discretized 2-factor representation. We see that different factors capture information of different semantic nature. More precisely, factor 1 tends to encode the shape of the digit, while factor 2 specialized in its color. This empirically demonstrates the emergence of "factorization" in the learnt representations. Further experimental details are provided in section H.4.1.

G.1.2 Robot Experiment Visual Data with Background Distractors

We then demonstrate the ability of DGRL to learn factorial representations on real world robot data, where the robot arm moves in presence of background video distractors. Further details on the robot arm data collection are provided in the appendix H.4.2. The data contains rich temporal background noise. We first learn a representation with a simple auto-encoder, following by the discretization bottleneck of DGRL, and then reconstruct the image with different discrete factors. Figure 15 demonstrates factorization in the learnt representation tweaking the different factors used in the discretization bottleneck. In particular, we observe some form of "compositionality" emerging as the decoder was never trained on some of the combinations of factors, for instance (person in the background + orange lamp) and (person in the background + arm to the left).

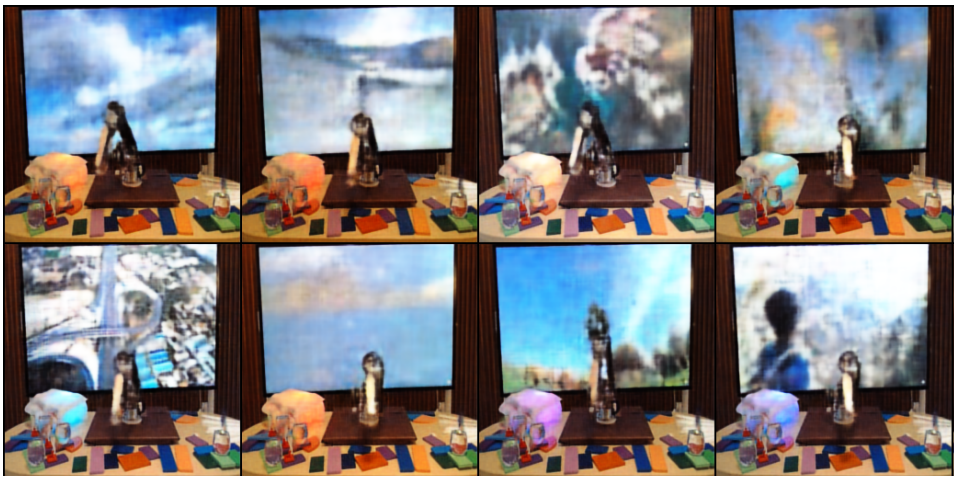
(a) Original Image (b) Both factors recon. (c) Factor 1 recon. (d) Factor 2 recon

Figure 14: Color-MNIST Example to demonstrate compositionality; reconstructing the original images with two factors. Leftmost : Original Image; Left-Middle : Reconstructed Image without substitution; Right-Middle : Reconstructed Image with one groups of discrete codes substituted by zero vectors; Rightmost : Reconstructed Image with the other groups of discrete codes substituted by zero vectors

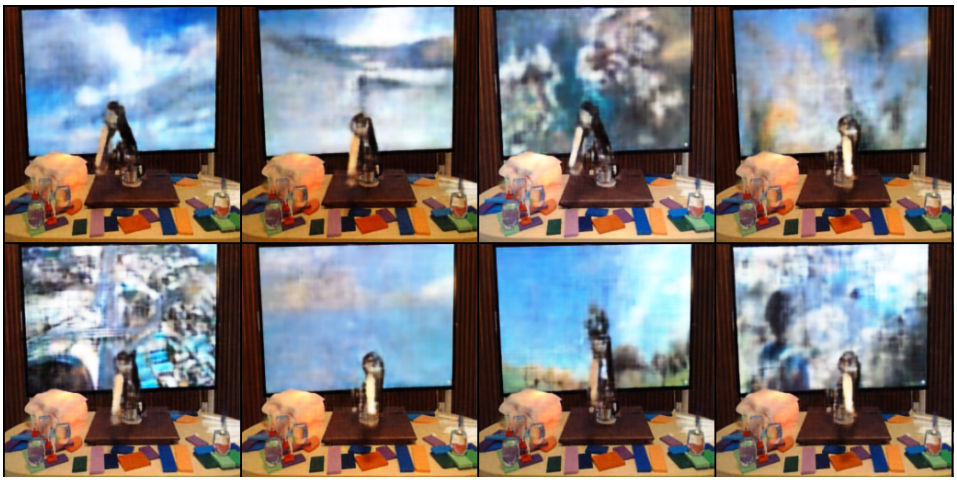
G.1.3 Disentanglement and Semantic Factorization on KeyChest

We then use the KeyChest environment, considered earlier in section 5. In this environment, we show whether adding DGRL leads to any semantic factorization on the learnt discrete codes, given the 2D state positions. We look at the visitations of the codes for each factor and visualize whether some factors correlate with the x or y positions of the state. For this experiment, we change the factors x and y numerically, and observe the learnt discrete codes for each of the factors in the 2D space, as shown in figures 16 and 17.

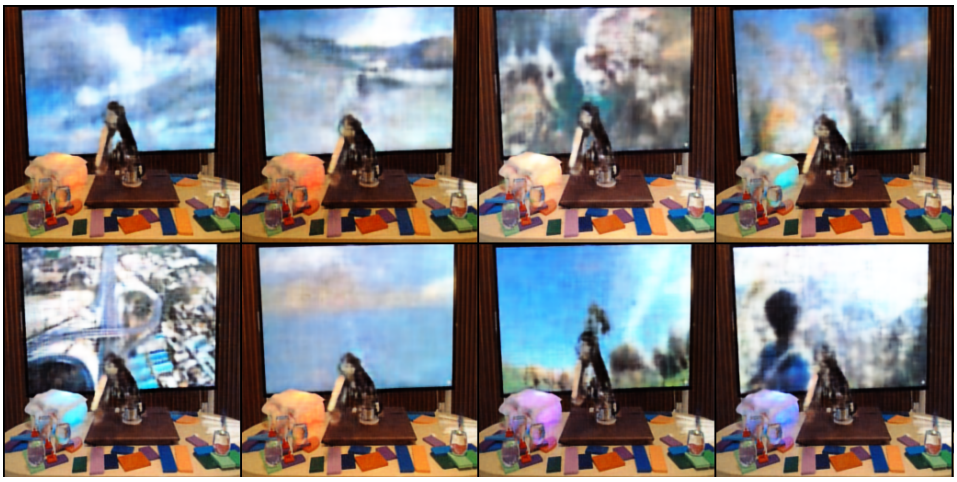
For this experiment, we sampled some 2D state positions with their corresponding discrete codes in several iterations in the middle of the training procedure. Concretely, we sampled the column and the row of state positions that includes the "box" state (the target position in this task) within five consecutive training episodes for visualizing the shift of the codes frequency. Figure 16 and figure 17 shows the different codebooks that are learnt for each state, where we look at 12 different states in total (varying the x and y positions). Experiment results show that differnet codes are being learnt for each state, while there exists some commonalities in the discrete codes as well, since the states are close to each other. For example, using a factor size of 2, figure 16 shows that a discrete code of 200 is learnt for almost all states, while the other factor changes (e.g comparing state $(10, 2)$ and state $(10, 4)$ which suggests that there are some latents which are semantically similar, while for others there exists disentanglement, given the states are nearby each other. Similarly, from figure 17 where we change the x positions of the states, keeping y fixed, for a range of states, we find that one of the factors is dominant for all these states, ie discrete code 80, while the other factor mostly changes (e.g when comparing states $(7, 1)$ and $(10, 1)$. For our experiments, we used a codebook size of 256, since total size of the state space in keychest 2D task is 13×17.



(a) Normal Reconstructions



(b) Change Lamp to Orange (changing a single factor)



(c) Make arm point left (changing a single factor)

Figure 15: An autoencoder trained with 8 discrete codes to reconstruct images of a real-robotic arm with independently changing distractors (TV and lamp). The first two rows show the reconstruction of the initial images. In the middle two rows, we change one of the discrete factors to match its value in the first image, in the resulting reconstruction, the lamp has turned orange. On the bottom two rows, we change another discrete factor, to match its value from the first image, and observe that it made the robotic arm point left

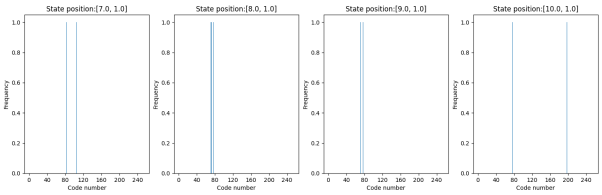


Figure 16: Discrete codes assigned frequency in KeyChest where a goal (x, y) is mapped into discrete codes. The x-axis is the sequence number of the codes, the y-axis is the frequency. We fixed y and changed x to show the corresponding shift of the codes frequency.

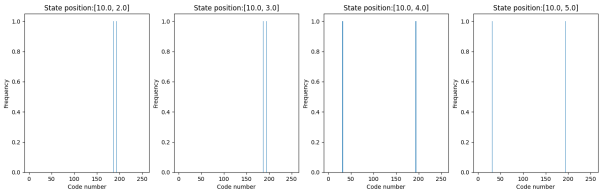


Figure 17: Discrete codes assigned frequency in KeyChest where a state position (x, y) is mapped into discrete codes. The x-axis is the sequence number of the codes, the y-axis is the frequency. We fixed x and changed y to show the corresponding shift of the codes frequency.

H Individual Responses to Each Reviewer Questions

H.1 Response to Reviewer : oX1u

Thank you Reviewer oX1u for your detailed comments and questions. We try to address them accordingly. We answer each of the questions raised in details as below :

Q1 : Why is it necessary to use two different autoencoder architecture for environments with different difficulty : We emphasize here that our approach with DGRL is in fact robust to whether we use the self supervised representation loss with discretization during pre-training phase or fine-tuning phase. The discretization bottleneck can be used only during pre-training, as shown in the maze tasks, or even during fine-tuning (or online phase) as shown in the control tasks. As shown in our experiments, in both cases, DGRL with discretization bottleneck can significantly improve empirical performance, depending on the number of discretizaton factors capturing the representation factorization. Moreover, for our experiments, we built on existing codebase and results provided by the baselines, to ensure reproducibility and fair comparison of baselines; therefore we use different encoder architectures for different tasks, depending on what was used by the baseline method. Our results show that DGRL can be robust to different encoder architectures and tasks, and the approach is a simple addition to existing baselines, which can improve performance by capturing representation factorization.

Q2 : The method assumes that the observations and goals come from the same space, restricting the method from being used when goals are specified in a different modality. We acknowledge that this is a valid consideration raised by the reviewer, since we do not consider settings where goals can come from different input modalities (e.g goals provided by language instructions). However, we want to emphasize that such additional experiments were beyond the scope of our work, since we only considered standard goal based RL settings. However, in some of our experiments, such as the SawyerTask or other control tasks - we do consider goals which are also provided by visual images, which are part of the observation space. To consider goals from other modalities, while considering typical RL tasks with high dimensional observations, is in fact beyond our scope for computational and experimental constraints. In our experiments, we covered a wide range of task setups, starting from toy examples to several standard traditional goal based RL benchmarks, and also ranged from settings where goals are either provided by a higher level controller, or part of

the environment itself. I hope the wide range of experiments we considered are convincing for the reviewer as to why DGRL can be a simple addition, but one that works quite well, across several benchmarks.

Q3 : The paper claims that the compositional representation helps encode goals into a semantically meaningful latent space (line 77), for example based on the example in Figure 1, I would expect some segments to correspond to goal colors and some segments to correspond to fruit types. However, HRL experiments in Section 5 do not provide evidence that the learnt goal representations are semantically interpretable. We emphasize that the traditional goal based RL benchmarks do not also consider such tasks where goals can be semantically interpretable. If such benchmarks were available to us, we could show reasonable results of how DGRL can be useful for semantic separability of goals, consisting of different colors and objects. For our experiments, we mostly considered the standard goal based tasks considered in the literature. However, that being said, in the rebuttal phase, we do add additional results for a wide range of image based data, either in a supervised learning setting or a RL setting, where we show how the multi-factor discretization considered in DGRL can capture different factors of variation in the input image. Please see section G.1 where we provide demonstrations of the compositional representations captured by the model. We want to emphasize that in the updated version of the draft, as suggested by reviewer $h8LS$, we do rephrase the term compostionality to be called *factored* or ***structured*** representations.

Q4 : How are the training goals and new goals for evaluation in the MazeWorld environment being selected? For our experiments, we show both settings where the evaluation goals can be *within distribution* or *out of distribution*. In the MazeWorld environments, given a distribution of training goals, we train an existing representation objective (e.g DRIML in this case) with the additional discretizaton bottlenecks across several training goals. And then during evaluation, consider both settings where evaluation of the Goal-DQN agent is done based on reaching the training goal distribution, or a setting where we place goals in different locations, and show how the method performs for out of distribution settings, as we have shown previously in figure 4.

Q5 : Does the method always require a dataset of goals collected prior to training, in order to train the goal representations? Do the curves reflect the sampling cost of these pre-collected data? Please refer to the answer below for *Q6* as well. For our experiments, we consider both the settings where we train DGRL on pre-collected data (ie, where representations are training during a pre-training phase only), or a setting where goals are collected online by the sampling policy, and we jointly train the encoder and policy/value networks end to end.

Q6 : How does the method compared to end-to-end training, where goals and observations are encoded into a latent space and this encoder is learnt jointly with the policy? We emphasize here that we indeed consider both the settings, where for some simpler tasks we consider only training representations with DGRL during the pre-training phase; whereas in some of the more complex control tasks, we simultaneously or jointly learn the encoder and the policy and value functions (ie, end to end training). Our experimental results Figure 3-4 show that adding DGRL in both the pre-training phase, as considered in task Loopworld and SpiralWorld environments (e.g figure 4) or end to end training (joint training), as considered in the goal image based Sawyer tasks (figure 9) and image based Ant navigaton task (figure 8).

H.2 Response to Reviewer : PnN7

We thank reviewer PnN7 for the detailed feedback. We answer each of the questions raised by the reviewer in details below.

Q1 : Goal space of current experiments and limitations Thank you for bringing this up. As we mentioned in the paper, our experimental results range from 2D goals spaces to even high dimensional image based goal spaces, and we show the effectiveness of DGRL across goal spaces of varying difficulty. As the reviewer pointed out, it would indeed have been interesting to do more experiments with high dimensional goal spaces, where for example the goals naturally have some level of factorization in the images, or goals consist of being able to interac with multiple objects. While

interesting, and as the reviewer pointed out, these are indeed beyond the scope of our current work - not just due to computational or resource limitations, but we would like to point out that even existing goal conditioned RL benchmarks are not really designed (yet) to be adaptable to such settings. It definitely is an interesting future work that we would like to pursue, especially with the idea of learning multiple discrete goal representations which are factorial in nature. However, unfortunately, current approaches are beyond our capabilities, primarily also due to the fact that such baselines are hard to make it to work in practice without reliable existing setups. Reviewer oX1u also pointed out a similar approach - where for example goals can come from different modalities (e.g language based instructions as goals), and we emphasize that these are indeed interesting future works, and we hope DGRL provides a step towards that, due to the ability of capturing factorial representations. We hope that in future, goals that require interacting with objects of different colours and shapes, can perhaps be solved with extensions of DGRL, where truly understanding the causal and factorial representation learning capability is an interesting future work.

Q2 : Continuous vs Discrete Goals and what the goal reaching policy is trained on. For example discretizing the goal when inputting it to the agent, but still rewarding the agnet based on the continuous representation

We ran an additional experiment on the toy task, based on the question above, where we compare the usefulness of using an intrinsic reward based on the discrete codes or using an intrinsic reward which still discretizes the goal, but we use the continuous representation of the goal. We show that due to the benefits of *goal grounding* and *goal specification* that can be achieved due to DGRL, providing an intrinsic reward based on the discrete goal outperforms the counterpart of using the continuous representation of the goal.

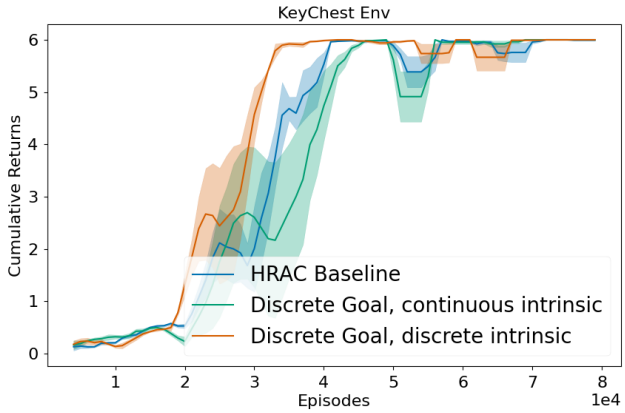


Figure 18: KeyChest result with comparisons between discrete intrinsic reward versus continuous intrinsic reward. In our previous experiment in figure 6 we were not providing an intrinsic bonus to the agent, but only used DGRL and showed benefits of using the discretization bottleneck. The updated result here shows that even when providing an intrinsic reward, DGRL integrated on top of the HRAC baseline can outperform the baseline, especially when using discrete goals for the intrinsic reward, as it allows for coarser *goal specification.*

Q3 : Specifying the codebook size L We regretfully inform that this is indeed true that we have unfortunately not mentioned the codebook size explicitly for the different experiments. For each of our experiments, depending on the complexity of the tasks, we indeed used codebooks of different sizes (while although it is a hyper-parameter, we wanted to ensure for reproducibility that none of our experiments require too high a codebook size, or the results are not too much susceptible to L). As such, for the toy tasks, we typically used codebook sizes of 64 in our experiments, while for the more complex control tasks, we used codebook sizes of 128, with different groups of factors ranging from $2, 4, 8, 16$ and 32 to achieve factorization. In fact, our experiments were more varying the differnet numbers of groups, while keeping the codebook size L fixed in most cases. If enough

compute budgets were available, it would have been interesting to do a wide sweep of experiments varying codebook sizes L, the number of embeddings and the number of groups of factors, to get a clearer picture of the level of factorization that can be achieved for each task.

Q4 : Would this method perform as well if the space of possible goals were itself high-dimensional, rather than effectively 2D? As we mentioned earlier, our experimental results consist of standard goal conditioned RL benchmarks, and we do a wide range of experiments, starting from 2D goal spaces (toy tasks) to high dimensional or visual goal spaces (control manipulation tasks). We emphasize that in either case, given a representation objective, DGRL is a simple integration that can be applied to any existing self supervised or representation learning objective based method that involves learning an encoder (from small state spaces to pixel based input); and we hope that our contribution can be seen as a simple, yet effective methodology, based on discretization bottlenecks, that can be straightforwardly applied to RL tasks to significantly improve performance.

H.3 Response to Reviewer : h8LS

Q1 : The method does not provide a significant improvement in most environments. It is of course interesting to know the benefits of this addition, but as the results show, the results are comparable for the majority of tasks, except in some cases of gridworld or ant maze navigation and the sawyer arm tasks, where the DGRL seems to help with sample efficiency.

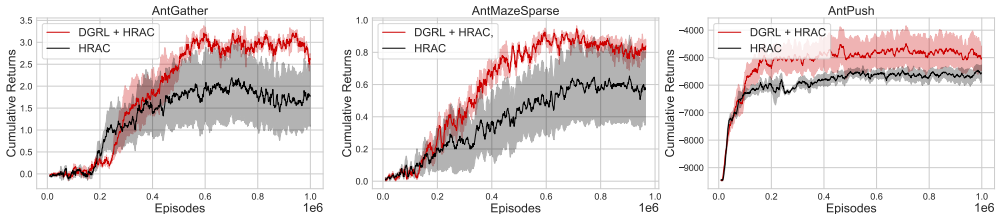


Figure 19: We emphasize the significance of DGRL through this result comparison. The results we show in the paper were being transparent about the effects of the different groups of factors; which is why in our plots, we included comparisons with all groups G. Previously, we reported results without fine-tuning the group factor G for a fair comparison. In this figure, we show the best performing factor G for each environment, that we can specifically fine-tune for the task; and as shown, we see that by simply adding DGRL on top of the existing HRAC baseline, we can achieve sigificant improvements on these hard manipulation tasks. With fine-tuning, the significant benefits of DGRL can become more apparent.

Q2 : Comment about the use of the term "compositionality". I would urge the authors to remove the term compositional from the title as well. Please see the questions section below. In many places in the manuscript, the discrete representations learned by the codebook are characterized as compositional. I would argue that compositionality in the representations learned using neural networks, though desirable, is something we have not yet achieved as a community. In my opinion, the reason for this is that compositionality requires some kind of syntax in the representation, which needs to be interpreted by something else in a very precise and consistent way. To claim that a representation is compositional, one would have to show that consistent replacements of parts of the representations lead to consistent results in behaviour. Due to the lack of such evidence, I would encourage the authors to remove the term "compositional" and instead use, perhaps, "factored" or "structured". I would also be interested to hear the authors' thoughts about my concerns. We hope to have addressed the issues with the term "compositionality" through our range of additional experiments earlier in section G.1. We do note that going further, we would re-phrase this as "factorial" representations, and have demonstrated additional results in section G.1 showing how this arises, with additional details on the experiment setup provided below in section H.4.

H.4 Experiment Details to Demonstrate Compositonality or Factorization

H.4.1 Experiment Details on Color-MNIST Dataset

We include a brief description of the experiment details, used for the color-mnist example to demonstrate factorization. The pixel-based input is first passed through an encoder (a two-layer neural network) to obtain its latent representation with the dimension of 30, we then quantize the continuous representation into two groups of discrete codes, where the codebook size is 256. In the training procedure, two groups of the discrete codes are then concatenated to obtain the discretized representation, and finally passed through a decoder (another two-layer neural network), where we used reconstruction loss (MSE loss) combining with the loss for vector quantization to train the network. While in the testing procedure, we used zero vector to substitute one group of the discrete codes, and then obtain the reconstructed image by concatenating it with the other group and passing through the decoder.

H.4.2 Experiment Setup on Robot Arm with Background Visual Distractor

The robot arm in our experiments moves in a grid with 9 different positions. We use two cameras to take images, for the dataset, one from the front side of the robot and the other with a top down view from above. We collect an image after each action is taken. The robot has 5 actions to take : move forward, backwards, right, left or stay in the current state. We use an episodic length of 500, ie, the robot arm moves for 500 steps after which we re-calibrate. The robot arm dataset is collected with a random uniform policy, for a total of 6 hours collecting 14000 samples. For learning the representation ϕ given the images, we use a small convolutional neural network to get an estimate $\phi(x)$ of the images x. In addition to the CNN network, we further learnt the latent state representation with an autoencoder.