Discrete Factorial Representations as an Abstraction for Goal Conditioned RL

Riashat Islam[1,5,7*], **Hongyu Zang**[3], **Anirudh Goyal**[2,7,8], **Alex Lamb**[6,7], **Kenji Kawaguchi**[4], **Xin Li**[3], **Romain Laroche**[5], **Yoshua Bengio**[2,7], **Remi Tachet Des Combes**[5]
[1] McGill University, Montreal, Canada, [2] University of Montreal, Montreal, Canada
[3] Beijing Institute of Technology, Beijing, China, [4] Harvard University, Cambridge, USA
[5] Microsoft Research, Montreal, Canada, [6] Microsoft Research, New York, USA
[7] Mila - Quebec AI Institute, Montreal, Canada [8] DeepMind

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 goal specification, 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. 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, 60] 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, both during training and evaluation [24]. One way to control this behavior is by specifying a reward signal [52, 56]. 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 [62, 63, 36, 37, 57], 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 [20].

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 [23]. 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 it was instructed to reach [61, 48, 44]. 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 its desired behavior [12, 5, 27]. Finally, the diversity of goals seen during training should help improve both credit assignment and generalization [48, 43].

*Corresponding Author. E-mail: riashat.islam@mail.mcgill.ca

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

While this framework is promising, it introduces two new challenges: *goal grounding* [8, 2] and *goal specification* [4]. 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 reaching the reward it was instructed 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 [39, 26]. Goal specification can be done either manually by a developer or by another RL agent, such as a high-level agent which generates goals a lower-level agent then tries to reach [12, 5, 27, 22]. 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 [19, 11, 22, 17, 1, 67]. At the same time, connecting language feedback for an agent is non-trivial, requiring special assumptions or a labeling framework [9].

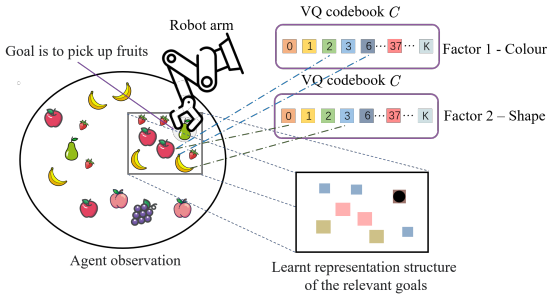


Figure 1: Illustration of learning *discrete* and *factorial* goal representations.

We propose to learn the goal representations with self-supervised learning (either trained on their own, or jointly with the downstream RL objective) while forcing them to be *discrete* and *factorial*. To perform this discretization, we use Vector-Quantization [64, 47, 33] 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 simplifies the grounding of unseen goals, *i.e.*, goals not seen during training, to novel compositions of the trained discrete codes. We show empirically that this improves the generalization performance of goal-reaching policies 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 a lower-level policy, and eventually complete the task (*i.e.*, reach the final 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 $g \in \mathcal{G}$ live in the state space $\mathcal{S}$, *i.e.*, $\mathcal{G} = \mathcal{S}$. 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 $\pi^h_{\theta_h}(g|s)$. 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.

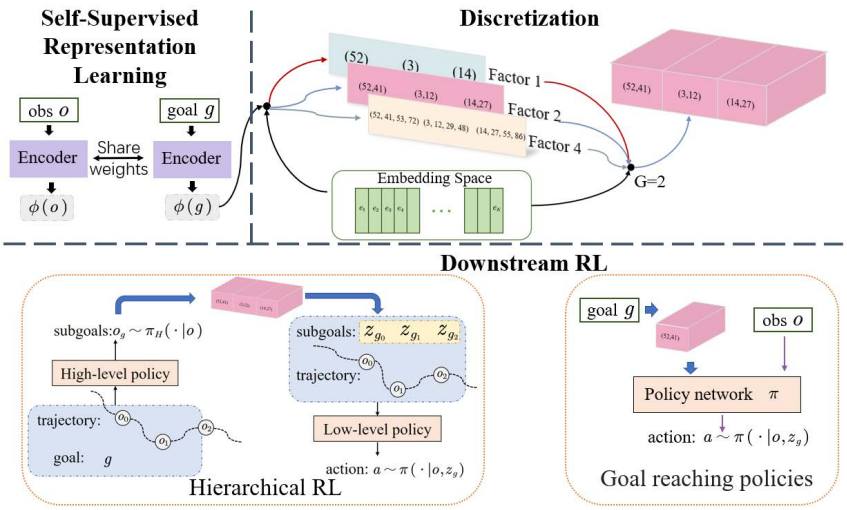


Figure 2: **Summary of Proposed DGRL Model** for improving *goal grounding* and *goal specification* by making goal representations *discrete* and *factorial*. We learn a latent representation for both observations and goals using a self-supervised learning method (sec. 3). We convert the learnt latent representation into discrete latents based on a VQ-VAE quantization bottleneck with multiple factor outputs (sec. 3). 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. 3.1).

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) [36] or policy optimization based algorithms [50, 51, 30]. 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. We consider both the HRL and goal-conditioned setups, and evaluate the significance of learning factorial representation of discrete latent goals in a series of complex goal-conditioned tasks.

Vector Quantized Representations. VQ-VAE [64, 47, 33] 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 [31, 58, 53, 34] have demonstrated the benefits of using a pre-training stage where the representations of raw states 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. We later describe, in Section 6, how discrete goal representations can accelerate learning in complex navigation and manipulation tasks. We emphasize that these representations 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.

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 state and maps it to a continuous embedding: $z_e = \phi(s)$. Here, we explore two different self-supervised techniques for learning representations. For simpler environments, we use a simple autoencoder with its standard reconstruction objective. For more complex environments, we use the Deep InfoMax approach [34] which optimizes for a contrastive objective as a proxy to maximizing the mutual information between representations of nearby states in the same trajectory.

Processing continuous representations via a discrete codebook. We learn discrete representations by using the vector-quantization method from the VQ-VAE paper [64], and follow the multi-factor setup used in Discrete-Value Neural Communication [33]. 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 $c_1, c_2, \ldots, c_G$, where $z_e = \text{CONCATENATE}(c_1, c_2, \ldots, c_G)$, and each segment $c_i \in \mathbb{R}^{m/G}$ (implying that m is divisible by G). Each continuous segment c_i is mapped independently 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:

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

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 [33] and [64]. 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 [33], 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 losses 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 L is the codebook size, G the number of factors per vector, and $q(\cdot)$ the whole discretization process. We train the representations for both the state and goal observations with this discretization bottleneck applied to the continuous representations resulting from the self-supervised training. The number of factors G is a hyper-parameter. In our experiments, we explored different values: $G = 1, 2, 4, 8, 16$, and found that $G = 16$ worked the best. Discretizing with more factors slightly increases computation but reduces the number of model parameters due to the codebook embeddings being reused across the different factors.

3.1 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. This 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 embeddings to the agent, we use their discretized versions, thus grounding goal representations in the input space.

We use the resulting representations to train 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 training episode, a goal is sampled from the goal distribution ρ_g, and the agent gets rewarded for reaching it. This reward can either be *extrinsic*, *i.e.*, part of the environment, or *intrinsic*, *i.e.*, 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 of the state observation. At test time, the agent can either be evaluated on reaching goals within the distribution ρ_g, or for its generalization capability to 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) 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 6, 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, 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 Related Work

Learning with multiple hierarchies has long been proposed in the RL literature, where goal conditioned HRL implements high level planning and low level control using sub-goals. Often in goal conditioned HRL, the higher level policy specifies goals which may not have good *specification* and *grounding*. Several prior works focus on goal-conditioned RL to improve sample efficiency in deep RL tasks [38, 40]. These build on ideas that were proposed years back to solve long horizon tasks by hierarchical RL specifying goals [23, 13, 14, 68]. The goal is to learn to solve sub-goals provided to the policy, by learning to predict a sequence of actions that can reach each of the sub-goals [66, 49, 41]. Additionally, in existing HRL literature, distance measures are often used based on goal-conditioned value functions, allowing to measure distances between states and the sequence of sub-goals to reach [16, 71], for planning [42], or exploration [29]. Since the set of goals specified in the state space can be arbitrary, an additional constraint is often also learnt to tie the distribution of selected goals to those the lower level policy can reach [72]. We tackle this problem by proposing DGRL, for better *grounding* and *specification* of sub-goal representations.

In previous works, mutual information based objectives have been proposed for goal conditioned RL. They perform goal-based representation learning, in order to improve stability of training goal-conditioned value functions [38, 40], or to provide goal representations allowing to identify decision states for better exploration [18]. However, for most of these settings, the sub-goals are based on an external reward and are lacking in terms of specification, which can lead to inefficient training. [32, 72] have proposed approaches that penalize the high level controller for generating sub-goals that are too difficult for the lower level policies, through the use of additional constrained objectives [72]. 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 [3, 54], often as a pre-training phase [55, 69], which can help for both exploration [35] and control [70].

In the context of goal conditioned RL, it can be a challenging problem as it additionally requires learning reliable representations of goals in parallel, purely from high dimensional observations [15]. Previous works [32] have often used the entire observation space as goals, which is not scalable for complex tasks. Other works have used a pre-defined space of sub-goals as domain knowledge [38], or self-play for sub-goal representations [59] to reduce the complexity of goal space design. Most recently, [42, 41] 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 a latent space of discrete goals usable for planning and control. We emphasize that DGRL can be integrated on any existing goal conditioned approach that utilizes learning a sub-goal representation.

6 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 trained to reach. 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 challeng-

ing testbeds that have previously been used in the RL community. DGRL in principle can be applied to any existing downstream goal-conditioned RL tasks. We demonstrate improvements on five such tasks. We consider maze navigation 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 [10] 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.

Demonstrating Factorized Representation Learning We first pick a color-mnist supervised learning example to support the idea that DGRL can learn factorized or compositional representations. Figure 3 displays reconstructed images from a trained decoder operating on a discretized 2-factor

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

Figure 3: Color-MNIST example to demonstrate factorized representations; 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

representation. We find 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 suggests the emergence of "factorization" in the learnt representations. Further experimental details are provided in section C.1.

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 is given in Appendix C.2.

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

Furthermore, for this task, we additionally use an intrinsic reward to promote 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 4 shows that DGRL significantly outperforms 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 5) and demonstrate improved generalization.

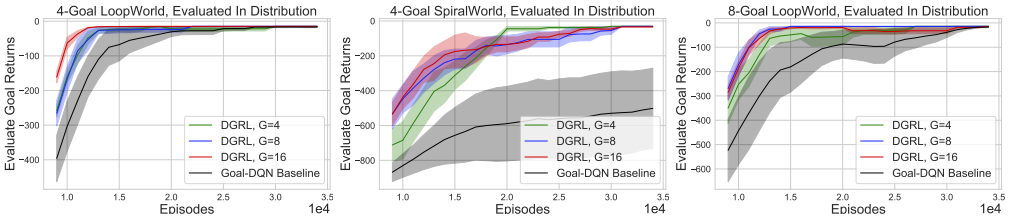


Figure 4: Loopworld maze environment. We show that for $4, 8$ and 16 different discrete factors, DGRL outperforms a goal-DQN baseline agent with continuous goal representations. As we increase the number of factors G to 16, the expressivity of the discrete goal representation increases, lowering the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

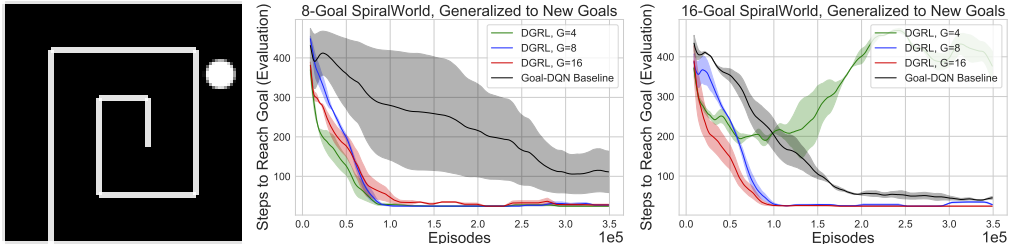


Figure 5: SpiralWorld environment (left). Generalization to a test distribution containing 4-goals in a SpiralWorld environment (left). We show the total number of steps to solve all test set goals, when trained on either an 8-goal or 16-goal training distribution.

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 a 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 [6], or as a higher-level policy in the case of hierarchical RL.

Procedurally Generated MiniGrid Exploration Task. We follow the experimental setup of [6] and [46] and evaluate DGRL on procedurally generated MiniGrid environments [10]. In [6], a goal-generating teacher proposes goals to train a goal-conditioned "student" policy. We integrate DGRL on top of AMIGO [6] 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.4. 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 [72], 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 [72] agent (details in Appendix C.3). Figure 7 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.

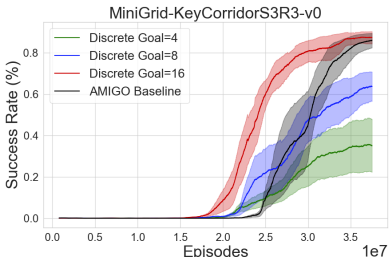


Figure 6: Performance comparison of the Amigo baseline [6, adversarially intrinsic goals] with and without DGRL for goal discretization.

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.

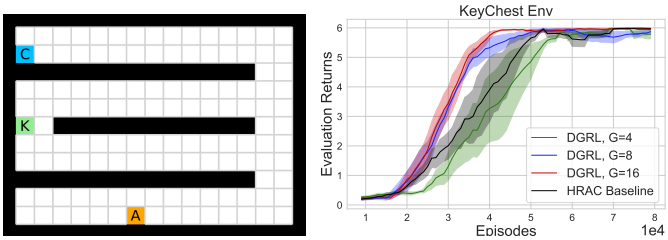


Figure 7: In KeyChest, the agent (A) starts from a random stochastic position, picks up the key (K) and then uses the key to open the chest (C). We find DGRL improves sample efficiency over the HRAC baseline.

Ant Manipulation Control Domains. We employed DGRL on three different continuous control tasks: AntMazeSparse, AntFall and AntPush. We emphasize that these tasks are the more challenging counterparts of AntGather and AntMaze tasks, typically used in the hierarchical RL community [38, 40]. Figure 8 provides an illustration. We evaluate goal discretization by integrating DGRL to the state-of-the-art HRAC baseline. Details of the experimental setup are provided in Appendix 6. Figure 8 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 [7, RIS]: a U-shaped maze, and an S-shaped maze (the S-shaped maze is shown in Figure 9). 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 [20, SAC] and temporal difference models [45, TDM] fail completely. Results in Figure 9 show that DGRL improves the sample efficiency over the RIS baseline. Additional experimental setup and environment configurations are provided in Appendix C.6.

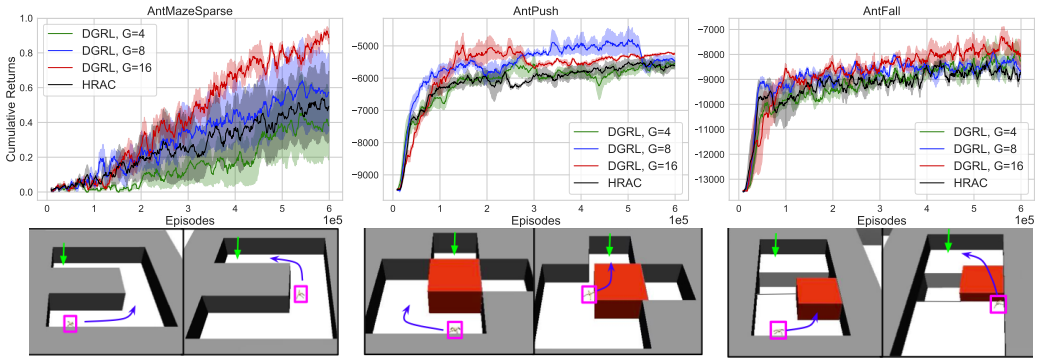


Figure 8: Comparison of DGRL with baseline HRAC [72] on 3 different navigation tasks.

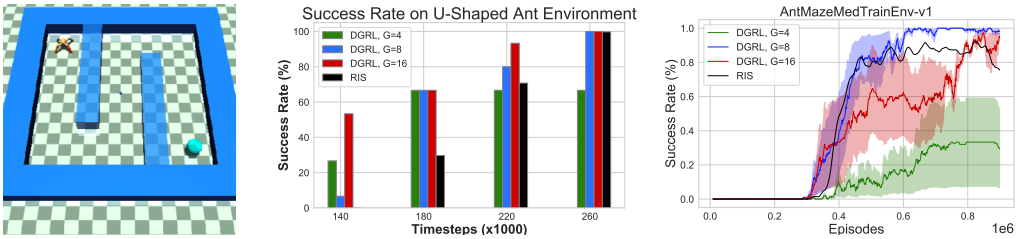


Figure 9: Performance comparison with the success rate of reaching goal positions during evaluation in an extended temporal configuration of the U-shaped and maze-shaped Ant navigation tasks. We find that integrating DGRL with RIS can lead to more sample efficient convergence on these tasks, while baselines such as SAC and TDM (not shown) fail completely on both AntU and AntMaze as reported by [7]. 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 6 to integrate DGRL with the state-of-the-art RIS baseline on the Sawyer task in Figure 10. This manipulation task is adapted from [42], 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 10 show that DGRL improves the sample efficiency over the RIS baseline. Details of the experimental setup are provided in Appendix C.7.

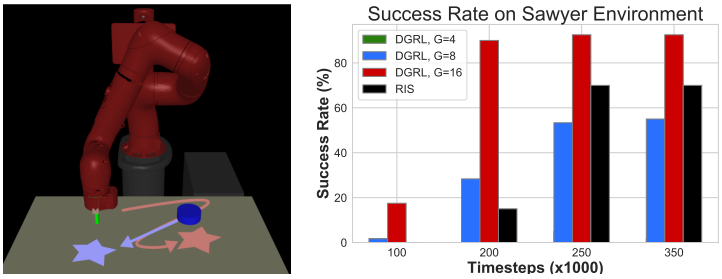


Figure 10: Sawyer Robotic Manipulation Task. Integrating DGRL with RIS (with a larger number of factors, G=8 and G=16, while G=4 fails), improves over the RIS baseline

7 Discussion

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.

Limitations and Future Work 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. Additionally, while we demonstrate that the factorial representations learnt by DGRL can be beneficial, it would be interesting to explore whether we can also enforce *compositionality* in the latent embeddings. Finally, exploring in more details the structure of the discrete factors forming the goals, their coverage, and to which extent they semantically capture the underlying factors of the environment is a promising research avenue.

Acknowledgement

The authors would like to thank Nicolas Heess, John Langford, Yonathan Efroni, Manan Tomar, Dipendra Misra, Akshay Krishnamurthy, Pierre-Yves Oudeyer, Harm Van Seijen and Doina Precup for valuable discussions and insightful comments related to this work. Hongyu Zang and Xin Li were partially supported by NSFC under Grant 62276024.

References

[1] Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. Learning dynamic belief graphs to generalize on text-based games. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3045–3057. Curran Associates, Inc., 2020.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

[22] 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.

[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] Alex Lamb, Riashat Islam, Yonathan Efroni, Aniket Didolkar, Dipendra Misra, Dylan Foster, Lekan Molu, Rajan Chari, Akshay Krishnamurthy, and John Langford. Guaranteed discovery of controllable latent states with multi-step inverse models. *arXiv preprint arXiv:2207.08229*, 2022.

[29] Romain Laroche and Merwan Barlier. Transfer reinforcement learning with shared dynamics. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[30] Romain Laroche and Remi Tachet des Combes. Dr jekyll & mr hyde: the strange case of off-policy policy updates. *Advances in Neural Information Processing Systems*, 34:24442–24454, 2021.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

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

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

[53] 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.

[54] 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.

[55] 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.

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

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

[58] 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.

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

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

[61] 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.

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

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

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

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

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

[67] Nathaniel Weir, Xingdi Yuan, Marc-Alexandre Côté, Matthew Hausknecht, Romain Laroche, Ida Momennejad, Harm Van Seijen, and Benjamin Van Durme. One-shot learning from a demonstration with hierarchical latent language, 2022.

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

[69] 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.

[70] 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.

[71] 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.

[72] 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.

A Appendix

In section 5, 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 Demonstrating Factorized Representations in a Robot Experiment with Visual Background Distractors

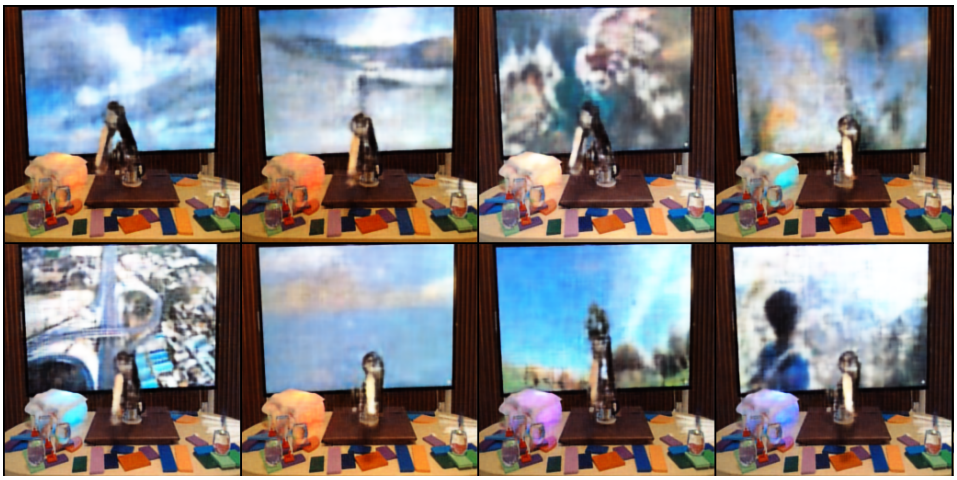
We demonstrate the ability of DGRL to learn factorial representations on real world robot data, where the robot arm moves in presence of background video distractors [28]. Further details on the robot arm data collection are provided below. The data contains rich temporal background noise. We first learn a representation with a simple auto-encoder, following by the discretization bottleneck of DGRL, and then reconstruct the image with different discrete factors. Figure 11 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).

Experiment Details: In this task, the robot arm moves on top of a grid layout, containing 9 different positions. We denote these as the *true states*. 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 a dataset containing pixel based observations only, where the images consist of the robot arm along with the background distractors. Inspired by the exogenous noise information setup [**?**], we setup the robot task while there is a TV playing a video in the background, with other flashing lights nearby. The offline dataset consists of 6 hours of robot data, with 14000 samples from the arm, taking high level actions of move left, right, up and down. A sample point image is collected after each action, and the background distractors changes significantly, due to video and lighting in the background. The goal of the experiment is to predict accurately the ground truth state position by learning latent representations with DGRL.

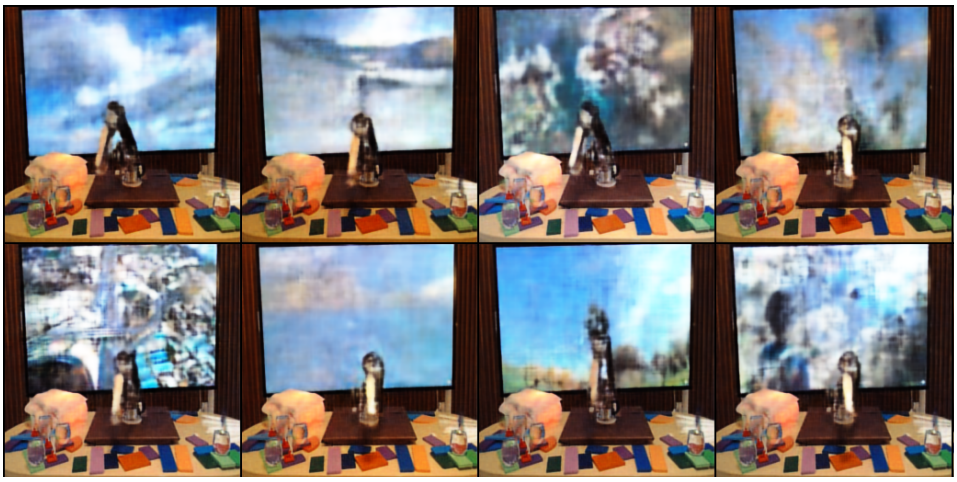
Experiment Results: We evaluate the ability of DGRL to accurately reconstruct the image, by learning the latent state representation while also ignoring the background distractors. This is denoted as the *Image Noise*, where we compare DGRL with and without VIB, alongside a baseline agent which only learns a representation. For learning latent representations, we use a multi-step inverse dynamics model [28]. In addition, we compare the ability of DGRL to accurately predict the ground truth states, denoted by *State Accuracy* solely from the observations, as a classification task. This is challenging since the learnt representation needs to predict ground states while ignoring the irrelevant background information. Furthermore with the learnt model, we predict the time-step for each observation as an additional metric to determine effectiveness of DGRL. The time-step is an indicator of the background noise that appeared in each sample; and with *Temporal Noise*, we evaluate DGRL to predict the time step while ignoring irrelevant information from observations. Experiment results in Figure **??** shows that the use of VIB helps improve the ability of DGRL to remove noise from the representation, while being able to almost perfectly predict the ground truth state of the robot.

C Experiment Details and Additional Results

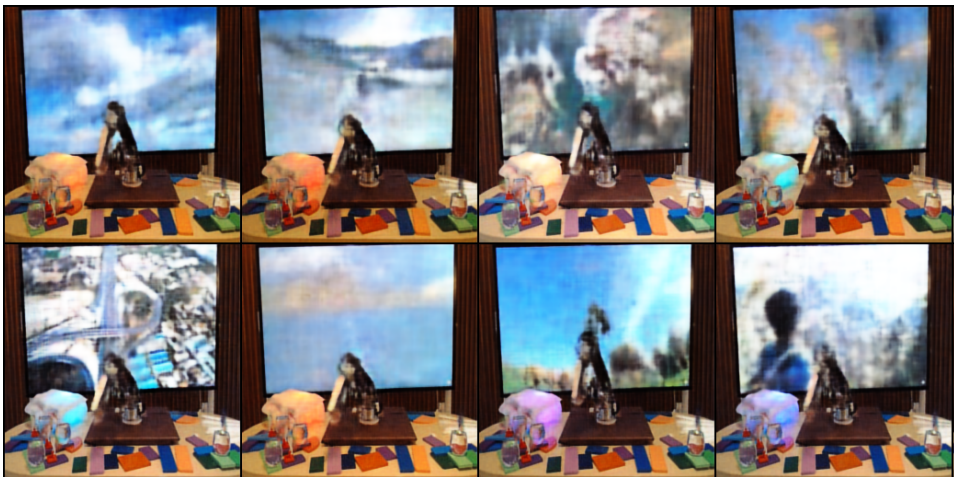
For all our experiments, we use existing open-source implementations of the baseline algorithms. We mostly integrate DGRL with HRAC [72] and RIS [7], which are state of the art goal conditioned algorithms on the complex Ant navigation and robotic manipulation tasks. We use the same hyperparameters and 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 [6, 72]. All our experiment results are based on 3 random seeds, and we provide our implementation for reproducibility.



(a) Normal Reconstructions



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



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

Figure 11: 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

C.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.

C.2 Visual MazeWorlds

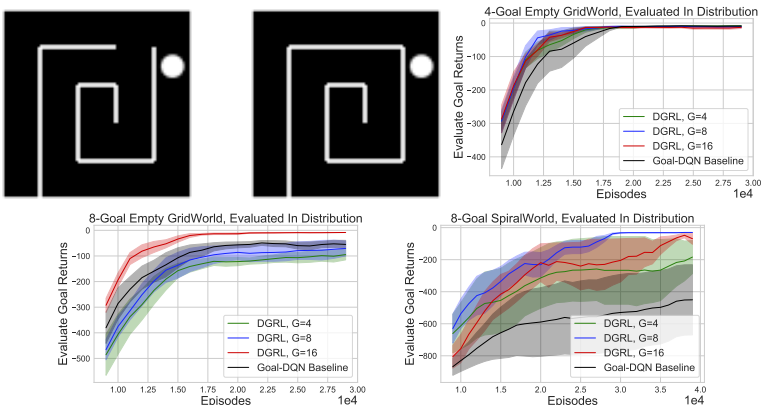


Figure 12: Visualization of the SpiralWorld and LoopWorld environments. Additional experimental results in an empty 4-goal and 8-goal environment across different factors G.

In this section, we provide additional details for the visual maze tasks. We provide environment visualizations, for the spiralworld and loopworld environments in figure 12. 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 [34] 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 significantly outperform a baseline goal DQN agent.

C.3 KeyChest Domain

For prototype and motivation of DGRL, we also used a simple discrete state and action KeyChest environment, as shown in figure 7. This task is inspired from the HRAC algorithm [72], 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 [72], 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.4 MiniGrid Environments

For our experiments, we follow the setup of [6] and [46] and evaluate DGRL on a simple procedurally generated MiniGrid environment [10]. 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 [6], 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 [6]. We use the open-sourced implementation provided by the authors (for more details, see [6]), and simply integrate DGRL on top of the AMIGO baseline, learning a factorial representation for the goal observations.

C.5 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 [72]. 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 13 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.

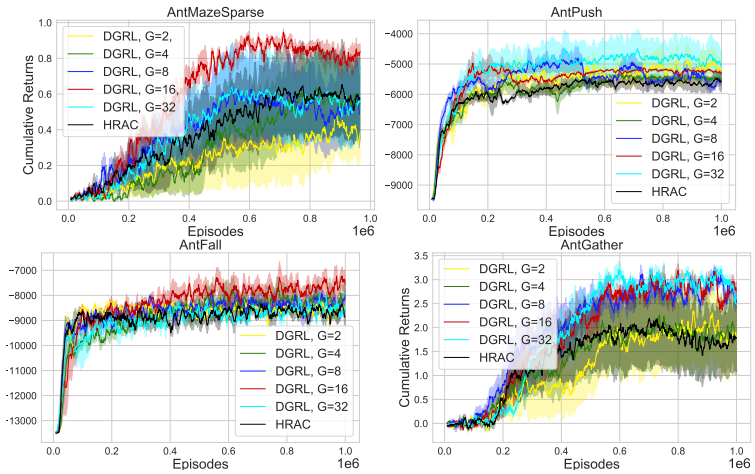


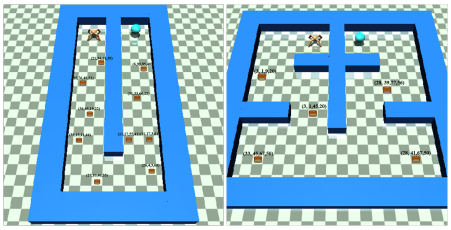
Figure 13: Ablation analysis on the Ant tasks. Standard goal conditioned HRL tasks, AntGather (shown on bottom left) and AntMaze, showing performance comparison of DGRL with baseline HRAC [72] across 3 random environment seed. We include comparisons with all factors G for all the Ant manipulation environments that we expeirmented on, given computation budget.

C.6 Ant Navigation Tasks

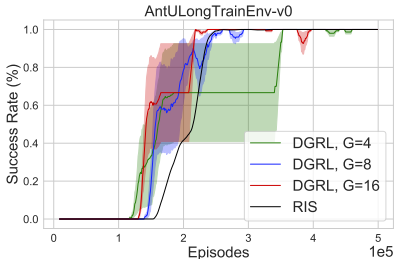
Following the setup in RIS [7], we employ an ant robot with a continuous 8-dimensional action space for all four Ant navigation tasks. reasoning. Figure 14(c) shows an additional result on the U-shaped Ant maze navigation environment. All these environments have a continuous state space including the 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 14, to test the performance of the agent on challenging configurations that require temporally extended

C.7 Robotic Manipulation Task

We consider the visual robot manipulation task Sawyer, from the multiworld environments of [42]. Our setup is based entirely on the experiment details and code provided in the open-source codebase of RIS [7]. 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 [7] 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 [42]. In this setting, we can additionally check for generalization. At test time, the cumulative returns of the agent are provided



(a) U-shaped maze (b) Π-shaped maze



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

Figure 14: 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 [7]. Performance comparison of success rate on AntU-shaped environment (right). We compare DGRL with different factors G and the RIS baseline [7] which has already been shown to outperform other goal conditioned baselines such as LEAP [42] on the Ant navigation tasks.

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

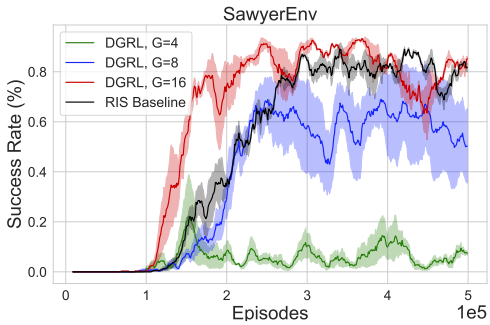


Figure 15: Performance comparison of DGRL with RIS baseline on a complex vision-based robot manipulation task

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

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)**.** *[65, 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

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

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

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

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,

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

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

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

If $\varsigma = q$, then

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

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

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

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 $\sum_{k\in\mathcal{I}_\mathcal{Q}}|\mathcal{I}_k| = n$,

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

For each $k \in [|\mathcal{Q}|]$, if $p(g \in \mathcal{G}_k) = 0$, then the probability of the event of $|\mathcal{I}_k| \geq 1$ 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 [21]) and union bounds over $k \in \mathcal{I}_\mathcal{Q}$ (noticing that $|\mathcal{I}_\mathcal{Q}| \leq |\mathcal{Q}|$ is finite), we have that $\sup_{\theta\in\Theta}|u_n(S,\theta)|\xrightarrow{P}0$ when $n \to \infty$. Thus,

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

□

E Algorithm in Goal Conditioned RL

We present the entire algorithm of DGRL built on top of RIS in algoirithm 1.

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 [7])
  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
```