Cell-Free Latent Go-Explore

Quentin Gallouédec [1] **Emmanuel Dellandréa** [1]

Abstract

In this paper, we introduce Latent Go-Explore (LGE), a simple and general approach based on the Go-Explore paradigm for exploration in reinforcement learning (RL). Go-Explore was initially introduced with a strong domain knowledge constraint for partitioning the state space into cells. However, in most real-world scenarios, drawing domain knowledge from raw observations is complex and tedious. If the cell partitioning is not informative enough, Go-Explore can completely fail to explore the environment. We argue that the Go-Explore approach can be generalized to any environment without domain knowledge and without cells by exploiting a learned latent representation. Thus, we show that LGE can be flexibly combined with any strategy for learning a latent representation. Our results indicate that LGE, although simpler than Go-Explore, is more robust and outperforms state-of-the-art algorithms in terms of pure exploration on multiple hard-exploration environments including *Montezuma's Revenge*. The LGE implementation is available as open-source at `https://github.com/qgallouedec/lge`.

1. Introduction

RL algorithms aim to learn a policy by maximizing a reward signal. In some cases, the rewards from the environment are sufficiently informative for the agent to learn a complex policy, and therefore achieve impressive results, including world level in Go (Silver et al., 2016), StarCraft (Vinyals et al., 2019), or learning sophisticated robotic tasks (Lee et al., 2019). However, many real-world environments provide extremely sparse (Bellemare et al., 2016), deceptive (Lehman & Stanley, 2011) rewards, or none at all. In such environments, random exploration, on which many current RL approaches rely, may not be sufficient to collect data that is diverse and informative enough for the agent to learn anything. In these cases, the agent must adopt an efficient exploration strategy to reach high reward areas, which may require a significant amount of interactions.

Recently, Ecoffet et al. (2021) introduced a new paradigm in which a goal-conditioned agent is trained to reach states it has already encountered, and then explore from there. The agent thus iteratively pushes back the frontier of its knowledge of the environment. We call this family of algorithms *return-then-explore*. Ecoffet et al. (2021) provide Go-Explore, an algorithm of this new family, that outperforms by several orders of magnitude the state-of-the-art scores on the game *Montezuma's Revenge*, known as a hard-exploration problem. Go-Explore relies on a grouping of observations into *cells*. These cells are used both to select target observations at the frontier of yet undiscovered states and to build a subgoal trajectory for the agent to follow to reach the final goal cell. As Ecoffet et al. (2021) initially spotted, the cell design is not obvious. It requires detailed knowledge of the observation space, the dynamics of the environment, and the subsequent task. If any important information about the dynamics of the environment is missing from the cell representation, the agent may fail to explore at all. For example, in *Montezuma's Revenge*, possession of a key is a crucial piece of information that when included in the cell representation increases exploration by several orders of magnitude. We also demonstrate in Appendix B that the cell design has a major influence on the results.

In this paper, we present Latent Go-Explore (LGE), a new algorithm derived from Go-Explore which operates without cells. This new algorithm meets the definition of a *return-then-explore* family of algorithms since the agent samples a final goal state at the frontier of the achieved states, returns to it, and then explores further from it. Our main contribution consists of three major improvements.

- A latent representation is learned simultaneously with the exploration of the agent to provide the most up-to-date and informative representation possible.
- Sampling of the final goal is based on a non-parametric density model in latent space. This leverages the learned latent representation for sampling the states

[1]Univ Lyon, Centrale Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, F-69130 Ecully, France. Correspondence to: Quentin Gallouédec <quentin.gallouedec@ec-lyon.fr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

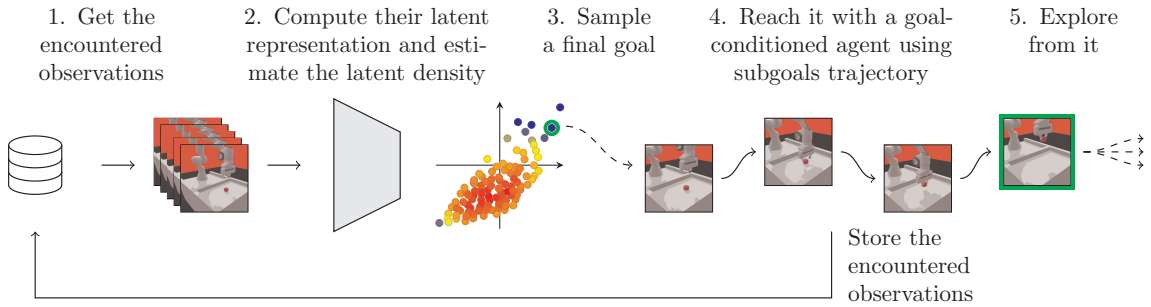


Figure 1. LGE exploration workflow. The encountered observations are encoded in a latent space. A latent density is estimated. A final goal is sampled from the states already reached, by skewing the distribution with the density. A goal-conditioned agent is trained to reach this goal by pursuing a sequence of subgoals, derived from the experiment that led to the final goal. Once the agent has reached the final goal, it explores from it with any exploration strategy.

of interest to be reached.

- The subgoal path pursued by the agent is reduced using a characteristic latent distance.

These three modifications, detailed in Section 3, allow us to generalize the Go-Explore approach to any continuous high-dimensional environment. It also enables the automation of the encoding of observations into an informative latent representation, eliminating the need for manual cell design. The full LGE exploration workflow is presented in Figure 1.

To evaluate LGE, we conducted experiments in the context of reward-free exploration in various hard-exploration environments including a maze, a robotic arm interacting with an object, and two Atari games known for their high exploration difficulty: *Montezuma's Revenge* and *Pitfall*.

LGE can use various types of latent representation learning methods. In this study, we demonstrate the use of three such methods, including inverse dynamics, forward dynamics, and auto-encoding mechanism. We show in Section 4.4 that for the environments studied, LGE outperforms all state-of-the-art algorithms studied in this paper, and in particular Go-Explore for the exploration task.

2. Preliminaries and Related Work

2.1. Preliminaries

Markow Decision Process This paper uses the standard formalism of a discounted Markov Decision Process (MDP) defined as the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho_0)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the (unknown) transition function providing the probability distribution of the next state given a current state and action, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, γ is the discount factor and ρ_0 is the initial distribution of states. A policy, denoted $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ is the probability distribution such that $\pi(a|s)$ is the probability of choosing action a in state s. We denote the previously defined values with discrete time t such that s_t, a_t and r_t denote respectively the state, action, and reward at timestep t. The goal is to learn a policy π that maximizes the long-term expected reward $\mathbb{E}[\sum_{t=0}^{+\infty} \gamma^t r_t]$.

Goal-conditioned MDP We note that every MDP can be augmented into a goal-conditioned MDP with a goal space $\mathcal{G}$ and an initial goal distribution ρ_g. At each timestep, the observation is augmented with a goal and the reward function depends on this goal. A goal-conditioned policy (Kaelbling, 1993), denoted $\pi(\cdot|\cdot, \cdot)$ also depends on the goal.

2.2. Related Work

Exploration in RL can be divided into three types (Ładosz et al., 2022): unstructured exploration[1], intrinsic rewards-based methods, and goal-based methods.

Unstructured exploration In unstructured exploration, the agent does not adhere to a predetermined exploration plan and instead takes actions randomly or according to a simple heuristic. These actions may be sampled uniformly from the action space, or in the continuous case, they may be augmented by exploration noise that is parametrized by the current state (Haarnoja et al., 2018) or not (Lillicrap et al., 2016). Unstructured exploration can be effective in some environments, but it may not be sufficient to explore more complex or sparsely rewarded environments.

Intrinsic rewards-based methods Intrinsic rewards-based methods are inspired by the concept of intrinsic motivation in cognitive science (Oudeyer & Kaplan, 2009). They involve the addition of an additional reward signal, called *intrinsic*, to the reward signal from the environment, called *extrinsic*. This intrinsic reward is designed to encourage

[1]We replace the terminology of Ładosz et al. (2022) *random exploration* by *unstructured exploration* that we think is more accurate.

Algorithm 1 LGE

Input: Number of iterations in the exploration phase T.
Initialize: Replay buffer $D = \emptyset$; Encoding module; Goal-conditioned policy π
while $t < T$ **do**
  Sample a final goal state with Equation (3).
  Build the subgoal trajectory τ^g using Equation (4).
  Initialize the subgoal index: $i \leftarrow 0$.
  while the last goal of τ^g is not reached **do**
    Collect interaction using $\pi(\cdot|\cdot, \tau_i^g)$ and store it into dataset D.
    if subgoal τ_i^g is reached, i.e. $||\phi(s_t) - \phi(\tau_i^g)|| < d$ **then**
      Move to the next subgoal: $i \leftarrow i + 1$.
    end if
  end while
  Explore until the end of the episode with any exploration strategy.
  Update π with any off-policy algorithm and HER.
  Every `update_encoder_freq` timesteps, update encoder ϕ with any representation learning algorithm.
end while
return the goal-conditioned policy π and the dataset D.

exploration. It can be based on the state visitation count (Bellemare et al., 2016; Machado et al., 2020) or on the prediction error of a model learned from the collected data (Houthooft et al., 2016; Pathak et al., 2017; Achiam & Sastry, 2017; Pathak et al., 2019; Burda et al., 2019; Tao et al., 2020).

Goal-based methods Methods that modify the reward of the environment provide no mechanism for distilling the knowledge gained from visiting various states. Agents may visit new states, but they quickly forget about them when other states become newer. To address this issue, recent work has suggested the use of a goal-conditioned autotelic agent specifically trained for the exploration task. This approach allows for the use of the knowledge gained during exploration to realize new user-specified goals (Levine, 2021; Colas et al., 2022). During the exploration phase, the reward signal is ignored, and after the exploration phase, the data collected by the agent is used to learn one or more subsequent tasks (Jin et al., 2020). Goal-based methods condition the agent with a goal that is used to guide exploration towards unknown areas. These methods rely on a goal generator to create goals for the agent. We divide goal-based methods into two categories: *exploratory goal* methods and *goals to explore from* methods (called *post-exploration* in (Yang et al., 2022)).

Exploratory goal methods follow the intuition that the agent discovers new areas of the observation space by pursuing goals that have been little or not achieved before. The challenge of these methods is to choose the goal to be neither too easy nor too hard. The literature contains several ways to approach this trade-off. Some methods sample goals that either maximize Learning Progress (Colas et al., 2019; Portelas et al., 2020) or value disagreement (Zhang et al., 2020). Other methods sample goals from the least visited areas using a parametric density model on the visited states (Pong et al., 2020). It is also possible to imagine goals that have never been reached using a language model (Colas et al., 2020), a generative model (Racanière et al., 2020) or a GAN (Florensa et al., 2018).

In *goals to explore from* methods the agent samples a goal from previously visited states. It returns to it, either by teleportation (Ecoffet et al., 2019; Matheron et al., 2020), or using a goal-conditioned policy (Ecoffet et al., 2021). The challenge of these methods is to choose a goal that is of high exploratory interest. Similarly, some methods estimate the density of the encountered states, using either parametric methods (Pitis et al., 2020) or non-parametric methods (Ecoffet et al., 2021; Matheron et al., 2020), to target the low-density areas.

In summary, methods based on a goal reaching policy should facilitate scalable RL. The stunning results of Go-Explore illustrate this point but remain circumscribed to few environments and require a lot of domain knowledge to work. By bridging with concepts already used in the intrinsic reward literature, we show a way to make this approach more general and simpler.

3. Latent Go-Explore

LGE meets the definition of the *return-then-explore* family of algorithms. First, a final goal state is sampled from the replay buffer, then the agent learns a goal-conditioned policy to reach this goal. When the agent reaches the goal, the agent starts to explore. LGE learns a latent representation of observations and samples the goal pursued by the goal-conditioned agent in priority in low latent density areas. In Section 3.1, we present how the latent representation of observations is learned. In Section 3.2, we show how the latent density is estimated and how the final goal state pursued by the agent is sampled. Finally, in Section 3.3, we show how to build a subgoal trajectory from the final goal to increase the agent's performance, in particular in far-away goal situations. The pseudo-code of the resulting algorithm is presented in Algorithm 1.

3.1. Learning a Latent Representation

The literature contains several latent representation learning methods for RL. Learning such a representation is orthogonal to our approach. Hence, LGE can be combined with any

learning method without the need for further modifications. Choosing the best representation learning method given the environment is out of the scope of this paper. In this paper, we present three methods of representation learning that have been found to work well with our test environments. Two of these methods are inspired by the literature on intrinsic reward-based methods,

Inverse dynamic representation learning Pathak et al. (2017) proposed an intrinsic reward calculated based on the agent's prediction error of the consequence of its own actions. The representation is learned using two submodules. The first encodes the observation into a latent representation $\phi(s_t)$. The second takes as input $\phi(s_t)$ and $\phi(s_t + 1)$ and outputs the prediction of the action taken by the agent at time step t. The parameters θ of the inverse model $\mathcal{P}_\theta^{\text{inv}}$ are optimized by minimizing the loss function:

$$L = \frac{1}{|N|} \sum_{(s_t, a_t, s_{t+1}) \sim D} \frac{1}{2} \| a_t - \mathcal{P}_\theta^{\text{inv}}(s_t, s_{t+1}) \|_2^2 \quad (1)$$

The inverse dynamics representation learning allows getting a latent representation of the states containing only the aspects of the state on which the agent can have an influence.

Forward dynamic representation learning In Achiam & Sastry (2017), the intrinsic reward is calculated based on the prediction error of a model approximating the transition probability function of the MDP. Two submodules are used. The first one encodes the observation to a latent representation $\phi(s_t)$. The second takes as input $\phi(s_t)$ and a_t and outputs the prediction of the next state $\hat{s}_{t+1}$. The model parameters θ are optimized by minimizing the loss function:

$$L = -\frac{1}{|N|} \sum_{(s_t, a_t, s_{t+1}) \sim D} \log \mathcal{P}_\theta(s_{t+1} \mid s_t, a_t) \quad (2)$$

Vector Quantized Variational Autoencoder (VQ-VAE) Autoencoding (Hinton & Salakhutdinov, 2006) aims to train a neural network to reconstruct its input by learning a compressed representation of the data. This approach is known to be effective in extracting useful features from the input, especially images. For Atari environments, we use a VQ-VAE (van den Oord et al., 2017), a technique that combines autoencoding with vector quantization, and has shown good results, while being simple to train. We use the coordinates of the embeddings in the embeddings table as the latent representation.

3.2. Density Estimation for Intrinsic Goal Sampling

The success of the proposed method relies on the agent's ability to generate for itself goals that it will be able to reach and then explore from there. For the agent to progress in the exploration of the environment, these goals must be at the edge of the yet unexplored areas. To identify these areas, we use an estimator of the density of latent representations of the encountered states. Moreover, we require the goal to be reachable. The set of reachable states is a subset of the state space that we assume to be unknown. The easiest way to satisfy the previous requirement is therefore to sample among the states that have already been reached.

We estimate the density of latent representations of the encountered states (called latent density) using the particle-based entropy estimator originally proposed by (Kung et al., 2012) and used in the literature on intrinsically motivated RL (Liu & Abbeel, 2021b;a). This estimator has the advantage of being nonparametric and thus does not hinge on the learning capabilities of a learned model. Appendix C describes the details of the implementation of this estimator, denoted $\hat{f}$.

The sampling of the final goal state follows a geometric law on the rank in the latent density sort R_i. The probability to draw s_i as the final goal state is

$$\mathbb{P}(G = s_i) = (1 - p)^{R_i - 1} p \quad (3)$$

where G is the random variable corresponding to the final goal state, and $0 \leqslant p \leqslant 1$ is a hyperparameter controlling the bias towards states with a low latent density.

This method has the advantage of being robust to approximation errors in the density evaluation, which can be particularly important in low density areas. In doing so, we only focus on the ability of the model to correctly order the observations according to their latent density.

The representation is jointly learned with the exploration of the agent. Therefore, the latent density must be regularly recomputed to take into account the most recent representation on the one hand, and the recently visited states on the other hand. However, considering the slow evolution of this value, we choose to recompute the latent density only once every 5k timesteps for maze and robotic environments, and every 500k timesteps for Atari environments. This allows us to significantly reduce the computation needs while having a low empirical impact on the results.

3.3. Subgoal Trajectory

As learning progresses, the sampled final goal states are increasingly distant. However, reaching a distant goal is challenging because it implies a sparse reward problem.

To overcome this problem, we condition the agent to successive intermediate goals $\tau^g = (g_0, g_1 ..., g_L)$ that should guide it to the final goal state g_L. These intermediate goals

are chosen from the trajectory that led the agent to the final goal state $(s_0, s_1, \ldots, s_T)$.

The trajectory that led the agent to the final goal state is unlikely to be optimal. Plus, if the agent is conditioned by the whole trajectory, it may fail to reach all of them, even though some of them may not be necessary to reach the final goal state. To allow the agent to find a better path to the final goal state, we remove some subgoals from this trajectory. To decide whether a subgoal should be removed from the trajectory, we evaluate the latent distance to the previous subgoal. If the distance is less than the threshold, then the goal is removed.

$$\forall i \leqslant L-1, \quad ||\phi(g_i) - \phi(g_{i+1})|| > d \tag{4}$$

Unlike Go-Explore, LGE don't use the best known trajectory that leads to the sampled goal area (cell). The main reason is that the best known trajectory may be particularly difficult to reproduce, due to the dynamics and the stochasticity of the environment, or cause the early termination of the episode. For example, in *Montezuma's Revenge*, there are two ways to reach the bottom of the left ladder (a necessary step to get the first key). The first one consists in jumping right from the promontory, but causes the death of Panama Joe (the character) and as a result ends of the episode. The second one, longer, consists in going around by the right ladder. Therefore, if the agent always chooses the shortest path (like Go-Explore), it will most likely fail to reach the first key and to further explore the environment.

Once the goal is reached, the agent explores using any exploration strategy. For the sake of simplicity, we choose a random exploration strategy for our experiments. We also impose that the agent repeats the previous action with a probability of 90%. This technique has been shown to increase the results significantly (Ecoffet et al., 2021).

4. Experiments

To demonstrate the effectiveness of our method, we apply it to a range of pure exploration tasks. We focus on environments for which naive random exploration is not sufficient to explore the rich variety of reachable states. We compare the results obtained with LGE with the results obtained using several algorithms based on intrinsic curiosity and others based on goal-directed strategies. For each environment, LGE uses the representation method that empirically gives the best results. Consequently, we use the forward dynamics for the maze environment, the inverse dynamics for the robotic environment, and the VQ-VAE for Atari.

In terms of infrastructure, each run was performed on a single worker machine equipped with one CPU and one NVIDIA® V100 GPU + 120 Gb of RAM.

4.1. Environments

Continuous maze We train an agent to navigate in a continuous 2D maze. The corresponding configuration is shown in Figure 2. The agent starts every episode in the center of the maze. At each timestep, the agent receives the current coordinates as an observation and chooses an action that controls its location change. If the agent collides with a wall, it returns to its previous position. The reachable space is a square of 12×12 and the agent's action is limited to $[-1, 1]$ horizontally and vertically. The agent can interact 100 times with the environment (which is just enough to explore all the maze), after which the episode ends.

Robotic environment Robotic environments are interesting and challenging application cases of RL, especially since the reward is often sparse. We simulate a Franka robot under the PyBullet physics engine using panda-gym (Gallouédec et al., 2021). The robot can move and interact with an object. The agent has access to the position of the end-effector and the position of the object, as well as to the opening of the gripper. The agent interacts 50 times with the environment and then the object and the robot arm are reset to their initial position.

Atari We train LGE on two high-dimensional Atari 2600 environments simulated through the Arcade Learning Environment (ALE, Bellemare et al. (2013)) that are known to be particularly challenging for exploration: *Montezuma's Revenge* and *Pitfall*. Details of the settings used are presented in Appendix A.

4.2. Baselines

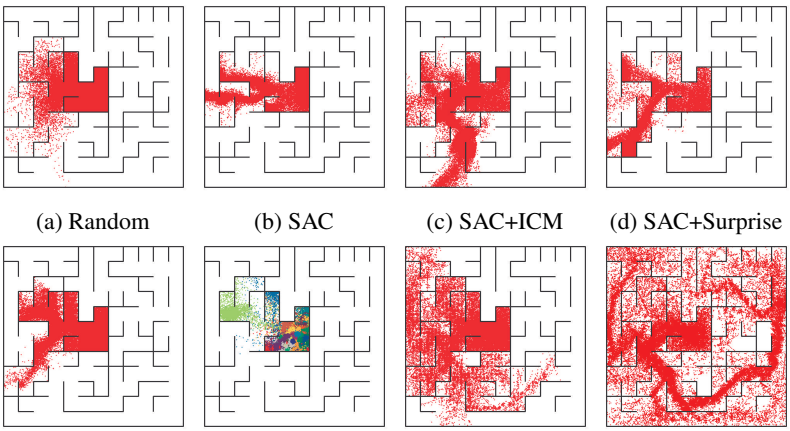
4.2.1. Random Exploration

Most RL methods from the literature do not follow any structured exploration strategy. In a reward-free context, the performance of the latter is often equivalent to a random walk. We take as a reference a random agent, whose actions are uniformly sampled over the action space at each time step, Soft Actor-Critic (SAC, Haarnoja et al. (2018)) and Deep Deterministic Policy Gradient (DDPG, Lillicrap et al. (2016)) for continuous action space environments.

4.2.2. Intrinsic Reward-Based Exploration

In this paper, we take as reference two widely used intrinsic reward-based methods combined with either SAC or DDPG. These methods stand out from the others because, despite their simplicity, they have demonstrated good performance on a wide variety of tasks.

Intrinsic Curiosity Module (ICM, Pathak et al. (2017)) The intrinsic reward is computed as the mean square error between the true latent representation and the one predicted



(a) Random (b) SAC (c) SAC+ICM (d) SAC+Surprise

(e) Skew-Fit (f) DIAYN (g) Go-Explore (h) LGE (ours)

Figure 2. Space coverage of the maze environment after 100k timesteps. In (f), the different colors are the different skills.

by a learned dynamic model given the action taken. The encoder is trained jointly with an inverse dynamics model.

Surprise (Achiam & Sastry, 2017) The intrinsic reward is the approximation of the KL divergence between the actual transition probabilities and a learned transition model.

4.2.3. Goal-Directed Exploration

Go-Explore The agent divides the observation space into cells, prioritizes the cells that have been visited the least, returns to them using a goal-conditioned policy, and then continues exploring from that point. This is the policy-based and without domain knowledge variant of Go-Explore, but we refer to it simply as *Go-Explore*. The observations in continuous environments are converted into cell representations by discretizing them. In the maze environment, we use a 24×24 grid, and in the robotic environment, we use a grid with a 0.1m resolution for the position of the gripper and object. For Atari, we use the same fixed cell representation as proposed by (Ecoffet et al., 2021) in the policy-based case: the observation is converted to grayscale and reduced to the size of 8×11 pixels. The depth is then reduced from 256 to 8 values according to $\lfloor \frac{8p}{255} \rfloor$ where p is the pixel value. The resulting image is the representation of the cell. Go-Explore is the closest baseline to our algorithm. Appendix D details the differences between LGE and Go-Explore.

Diversity Is All You Need (DIAYN, Eysenbach et al. (2019a)) The agent is conditioned by a *skill* and a discriminator predicts the skill pursued by the agent. The more the discriminator predicts with certainty the skill pursued, the bigger the reward. Conjointly, the discriminator is trained to maximize the distinguishability of skills.

Skew-Fit (Pong et al., 2020) The agent's goal sampling is skewed to maximize the entropy of a density model learned on the achieved states.

For the goal-directed methods, we use Hindsight Experience Replay (HER, Andrychowicz et al. (2017)) relabeling which has shown to significantly increase learning.

To nullify the variation in results due to different implementations, we implement all algorithms in the same framework: Stable-Baselines3 (Raffin et al., 2021). The set of intrinsic reward-based methods and goal-directed methods are underpinned by the same off-policy algorithm. The hyperparameters for this algorithm are identical. For the maze environment, we use SAC, while for the robotic environment, we use DDPG as it gives better results for all methods. For Atari environments, we use QR-DQN (Dabney et al., 2018), as it commonly considered to be a strong baseline on it. For Atari, we only compare LGE to Go-Explore as it far outperforms the others. To negate the influence of a bad choice of hyperparameter on the results, the method-specific hyperparameters are optimized. Appendix A details the optimization process and the resulting hyperparameters.

4.3. Measuring the Exploration

In this paper, we focus on the agent's ability to explore its environment in a pure exploration context, i.e. in the absence of extrinsic reward. This step is particularly important because, in the case of an environment with very sparse rewards, the agent can interact a large number of times with the environment without getting any reward. It is therefore necessary to follow an efficient exploration strategy to discover the few areas of the state space where the agent can get a reward. To be able to compare the results obtained by different methods in this context, it is necessary to use a common metric for the quality of exploration.

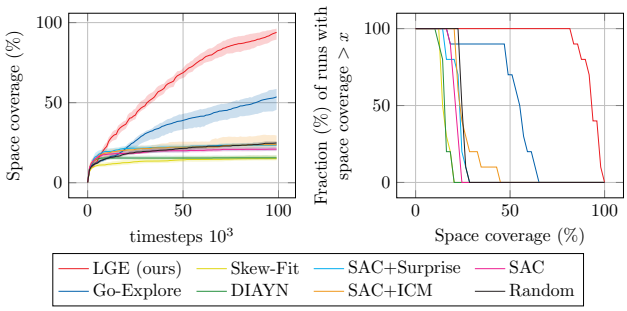


Figure 3. Comparison of the space coverage of the maze environment. Each experiment is run 10 times. The left plot represents the space coverage (number of cell divided by the total number of reachable cells) across timesteps. The solid lines are the IQMs and the shaded areas are the 95% confidence intervals. The right plot is the final performance profile (higher is better).

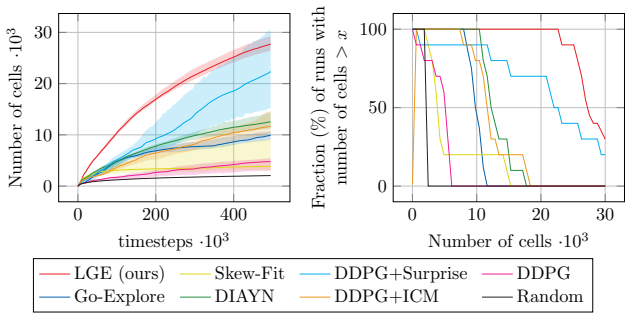


Figure 4. Comparison of exploration with the robotic environment. Each experiment is run 10 times. The left plot represents the number of explored bins across timesteps. The solid lines are the IQMs and the shaded areas are the 95% confidence intervals. The right plot is the final performance profile (higher is better).

The literature uses various metrics. Some papers use the average reward on a hard-exploration task (Ali Taïga et al., 2020), the zero-shot performance on a predefined task (Sekar et al., 2020) or monitor specific identifiable events in the environment that indirectly informs the degree of exploration (Gülçehre et al., 2020). We argue that these indirect measures are unsatisfactory as they rely on the subsequent learning ability of an online and offline agent respectively. For simplicity, we use the number of visited cells as the metric, whose construction strategy is explained in Section 4.2.3. Therefore, the figures represent the number of cells explored, although Go-Explore is the only algorithm to explicitly maximize this metric. Following the guidelines of (Agarwal et al., 2021), we use for all plots in this paper the interquartile mean (IQM) with the 95% confidence interval.

4.4. Main Results

The exploration results for the maze environment are presented in Figure 3. A rendering of the positions explored by the agent is presented in Figure 2. We note that only LGE and Go-Explore significantly outperform the results obtained with random exploration. This demonstrates the effectiveness of the *return-then-explore* paradigm in this environment. We note that exploration based on intrinsic curiosity does not yield significantly better results than those obtained by random exploration. We hypothesize that the simple dynamics of the environment makes the intrinsic reward to quickly converge to 0.0. Surprisingly, neither Skew-Fit nor DIAYN performs significantly better than random exploration. For DIAYN, we find that most of the skills were concentrated in the initial position area of the agent. We hypothesize that this is the consequence of the lack of *post-exploration* described by (Yang et al., 2022). Finally, we note that, although the cell size has been optimized, LGE significantly outperforms Go-Explore. LGE manages to cover almost the entire reachable space at the end of the runs while exhibiting low variability in the results.

The exploration results for the robotic environment are presented in Figure 4. We notice that LGE significantly outperforms all other methods. Notably, Go-Explore performs only slightly better than random exploration. We note that Go-Explore does not learn to grasp the object throughout the learning process. The results presented on a robotic environment by Ecoffet et al. (2021) are much better. We presume this is mainly due to the meticulous work done on the state space examination and the induced cell design. Here, we use a naive grid-like cell design. Although the grid parameter is optimized, it does not yield good exploration results with this environment. We thus demonstrate the benefit of using a learned representation to automatically capture important features of the environment's dynamics.

The exploration results for Atari are presented Figure 5. We see that both LGE and Go-Explore quickly discover a large number of cells, then continue their exploration by regularly discovering new cells. LGE slightly outperforms

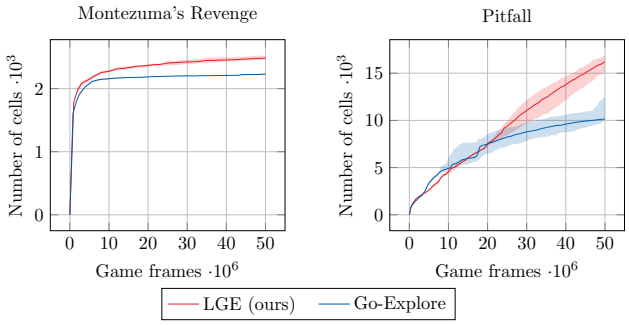


Figure 5. Comparison of exploration on the Atari environments. Each experiment is run 3 times. The solid lines are the IQMs and the shaded areas are the 95% confidence intervals.

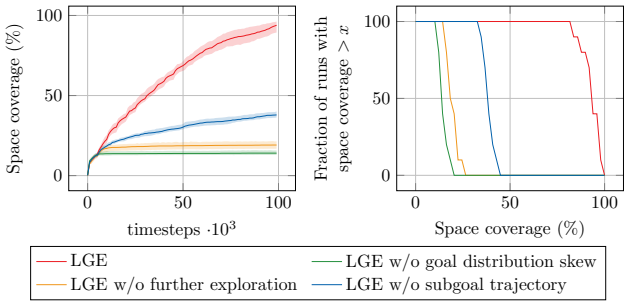


Figure 6. Result of ablation study on the maze environment. Each experiment is run 10 times. The left plot represents the space coverage across timesteps. The solid lines are the IQMs and the shaded areas are the 95% confidence intervals. The right plot is the final performance profile (higher is better).

Go-Explore on both *Pitfall* and *Montezuma's Revenge*. Nevertheless, we note that the number of discovered cells is much lower than that of Go-Explore in its full configuration (around 5k for *Montezuma's Revenge*), including domain knowledge and the ability to reset the environment in any state. This shows the criticality of these settings for exploring these particular environments.

4.5. Ablation Study

We study the impact of the ablation of three key elements of LGE. (1) LGE without further exploration (as exploratory goal methods, see Section 2.2): the environment is reset once the agent reaches the final goal instead of performing the exploration random interactions. (2) LGE without skewing the final goal distribution in favor of low latent density areas, the final goals are sampled uniformly among the reached goals. (3) LGE without subgoals trajectory reduction: the agent is just conditioned by the final goal state. We perform these ablation studies on the maze environment and use the same hyperparameters as in Section 4.4. The results are shown in Figure 6.

The impact of the three ablations on the outcome is significant. We find that exploration after reaching the final goal is crucial, confirming the results of (Yang et al., 2022); without it, the agent reaches the limits of its knowledge but has little chance to explore further. Additionally, sampling goals with low latent density can significantly improve results by directing exploration to states with high exploratory value. Furthermore, we observe that conditioning the agent with successive subgoals greatly improves its exploration.

5. Discussion

5.1. Limitation and Future Work

Goal-achievement functions In LGE, an agent is considered to have reached a goal (whether final or intermediate) when the latent distance between its state and the goal is below a threshold. This is a naive way of defining a goal achievement function (Colas et al., 2022) that depends crucially on the latent representation. We believe that the results could be improved by envisioning a more informative and suitable goal achievement function for our method.

The initial state must remain the same across episodes The approach we propose is based on the assumption that the agent is always initialized in the same state. This assumption guarantees that at the beginning of each episode, all the states previously reached are reachable and that the subgoal trajectory starts with the initial state of the agent. However, in some environments, especially in procedurally generated environments, this assumption is not fulfilled (Küttler et al., 2020). In this situation, trying to follow the subgoal trajectory may be counterproductive in reaching the final goal. It is also possible that the final goal is not even reachable. However, note that even the pursuit of an unreachable goal can foster exploration. We believe that an approach inspired by generative networks such as (Racanière et al., 2020) may be appropriate to overcome this problem.

High-dimension environments and representation learning Our main contribution consists in the generalization of the Go-Explore approach by using a latent representation. LGE is notably effective in high-dimensional environments, specifically those with image observations. Representation learning is the keystone of the method. We provide a proof of concept for a forward model, an inverse model, and a VQ-VAE. We believe that the results can be greatly improved by choosing more finely the representation learning method for each environment by taking advantage of the many works dealing with this subject (Lesort et al., 2018). Representations are expected to encapsulate transitional proximity between observations, a feature not guaranteed by most learning methodologies. Nonetheless, in practice, such transitional proximity is often exhibited in learned representations.

The representation used by Search on Replay Buffer (SoRB) (Eysenbach et al., 2019b) is directly that of the critic. Using the same reward structure as LGE, the critic thus has the nice property of basically learning the negative distance of the shortest directed path between two states. Overall, we believe that the use of SoRB in the "Go" phase can be a substantial improvement of LGE and is a promising way to solve the three limitations mentioned above.

Finally, we believe that the community should endeavour to find a relevant metric for exploration, especially for image-based environments. We expect that such a metric would allow a more accurate comparison of different methods.

5.2. Conclusion

We introduce LGE, a new exploration method for RL. In this method, our agent explores the environment by selecting its own goals based on a jointly learned latent representation. LGE can be used as pre-training in environments where rewards are sparse or deceptive. Our main contribution is to generalize the Go-Explore algorithm, allowing us to benefit from representation learning algorithms for exploration. We present statistically robust empirical results conducted on diverse environments, including robotic systems and Atari games, that demonstrate our approach's significant improvement in exploration performance.

Acknowledgements

We would like to thank Adrien Ecoffet, Joost Huizinga, and Jeff Clune for their encouraging feedback and for the time they spent discussing this exciting topic. We would also like to thank the reviewers of our article for their efforts in reviewing and providing valuable comments. Their comments and suggestions were very helpful in converging on the final version of this work. This work was granted access to the HPC resources of IDRIS under the allocation 2022-[AD011012172R1] and 2022-[AD011013894] made by GENCI.

References

Achiam, J. and Sastry, S. Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning, 2017.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. G. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, pp. 29304–29320, 2021.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631. Association for Computing Machinery, 2019.

Ali Taïga, A., Fedus, W., Machado, M. C., Courville, A. C., and Bellemare, M. G. On Bonus-Based Exploration Methods in the Arcade Learning Environment. In *8th International Conference on Learning Representations*, 2020.

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight Experience Replay. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying Count-Based Exploration and Intrinsic Motivation. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by Random Network Distillation. In *7th International Conference on Learning Representations*, 2019.

Colas, C., Fournier, P., Chetouani, M., Sigaud, O., and Oudeyer, P.-Y. CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1331–1340. PMLR, 2019.

Colas, C., Karch, T., Lair, N., Dussoux, J.-M., Moulin-Frier, C., Dominey, P., and Oudeyer, P.-Y. Language as a Cognitive Tool to Imagine Goals in Curiosity Driven Exploration. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3761–3774. Curran Associates, Inc., 2020.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. Autotelic Agents with Intrinsically Motivated Goal-Conditioned Reinforcement Learning: a Short Survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional Reinforcement Learning With Quantile Regression. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *The Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2892–2901. AAAI Press, 2018.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. Go-Explore: a New Approach for Hard-Exploration Problems, 2019.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First return, then explore. *Nature*, 590(7847): 580–586, 2021.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is All You Need: Learning Skills without a Reward Function. In *7th International Conference on Learning Representations*, 2019a.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 15220–15231. Curran Associates, Inc., 2019b.

Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic Goal Generation for Reinforcement Learning Agents. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1514–1523. PMLR, 2018.

Gallouédec, Q., Cazin, N., Dellandréa, E., and Chen, L. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

Gülçehre, Ç., Le Paine, T., Shahriari, B., Denil, M., Hoffman, M., Soyer, H., Tanburn, R., Kapturowski, S., Rabinowitz, N. C., Williams, D., Barth-Maron, G., Wang, Z., de Freitas, N., and Team, W. Making Efficient Use of Demonstrations to Solve Hard Exploration Problems. In *8th International Conference on Learning Representations*. OpenReview.net, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313 (5786):504–507, 2006.

Houthooft, R., Chen, X., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. VIME: Variational Information Maximizing Exploration. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Jin, C., Krishnamurthy, A., Simchowitz, M., and Yu, T. Reward-Free Exploration for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 4870–4879. PMLR, 2020.

Kaelbling, L. P. Learning to Achieve Goals. In Bajcsy, R. (ed.), *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1094–1099. Morgan Kaufmann, 1993.

Kung, Y.-H., Lin, P.-S., and Kao, C.-H. An optimal k-nearest neighbor for density estimation. *Statistics & Probability Letters*, 82(10):1786–1791, 2012.

Küttler, H., Nardelli, N., Miller, A., Raileanu, R., Selvatici, M., Grefenstette, E., and Rocktäschel, T. The NetHack Learning Environment. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7671–7684. Curran Associates, Inc., 2020.

Ładosz, P., Weng, L., Kim, M., and Oh, H. Exploration in Deep Reinforcement Learning: A Survey. *Information Fusion*, 85:1–22, 2022. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2022.03.003.

Lee, J., Hwangbo, J., and Hutter, M. Robust Recovery Controller for a Quadrupedal Robot using Deep Reinforcement Learning, 2019.

Lehman, J. and Stanley, K. O. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223, 2011.

Lesort, T., Díaz-Rodríguez, N., Goudou, J.-F., and Filliat, D. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2018.07.006. URL `https://www.sciencedirect.com/science/article/pii/S0893608018302053`.

Levine, S. Understanding the World Through Action. In Faust, A., Hsu, D., and Neumann, G. (eds.), *Proceedings of the Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1752–1757. PMLR, 2021.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous Control with Deep Reinforcement Learning. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations*, 2016.

Liu, H. and Abbeel, P. APS: Active Pretraining with Successor Features. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6736–6747. PMLR, 2021a.

Liu, H. and Abbeel, P. Behavior From the Void: Unsupervised Active Pre-Training. In Ranzato, M., Beygelzimer, A., Nguyen, K., Liang, P., Vaughan, J., and Dauphin, Y. (eds.), *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021b.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M. J., and Bowling, M. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Machado, M. C., Bellemare, M. G., and Bowling, M. Count-Based Exploration with the Successor Representation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 5125–5133. AAAI Press, 2020.

Matheron, G., Perrin, N., and Sigaud, O. PBCS: Efficient Exploration and Exploitation Using a Synergy Between Reinforcement Learning and Motion Planning. In Farkas, I., Masulli, P., and Wermter, S. (eds.), *Artificial Neural Networks and Machine Learning*, volume 12397 of *Lecture Notes in Computer Science*, pp. 295–307. Springer, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Oudeyer, P.-Y. and Kaplan, F. What is Intrinsic Motivation? A Typology of Computational Approaches. *Frontiers in Neurorobotics*, 1:6, 2009.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-Driven Exploration by Self-Supervised Prediction. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, PMLR, 2017.

Pathak, D., Gandhi, D., and Gupta, A. Self-Supervised Exploration via Disagreement. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071. PMLR, 2019.

Pitis, S., Chan, H., Zhao, S., Stadie, B. C., and Ba, J. Maximum Entropy Gain Exploration for Long Horizon Multi-goal Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7750–7761. PMLR, 2020.

Pong, V., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S. Skew-Fit: State-Covering Self-Supervised Reinforcement Learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7783–7792. PMLR, 2020.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y. Teacher Algorithms for Curriculum Learning of Deep RL in Continuously Parameterized Environments. In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 835–853. PMLR, 2020.

Racanière, S., Lampinen, A. K., Santoro, A., Reichert, D. P., Firoiu, V., and Lillicrap, T. P. Automated Curriculum Generation through Setter-Solver Interactions. In *8th International Conference on Learning Representations*, 2020.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms, 2017.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to Explore via Self-Supervised World Models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8583–8592. PMLR, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016.

Tang, Y. and Kucukelbir, A. Hindsight Expectation Maximization for Goal-conditioned Reinforcement Learning. In *24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 2863–2871. PMLR, 2021.

Tao, R. Y., Francois-Lavet, V., and Pineau, J. Novelty Search in Representational Space for Sample Efficient Exploration. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 8114–8126. Curran Associates, Inc., 2020.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural Discrete Representation Learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T.,

Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019. doi: 10.1038/s41586-019-1724-z.

Yang, Z., Moerland, T. M., Preuss, M., and Plaat, A. First Go, then Post-Explore: the Benefits of Post-Exploration in Intrinsic Motivation, 2022.

Zhang, Y., Abbeel, P., and Pinto, L. Automatic Curriculum Learning through Value Disagreement. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7648–7659. Curran Associates, Inc., 2020.

A. Hyperparameters and Environments Settings

To limit the impact of the large variability of results depending on the hyperparameters, we chose to optimize the hyparameters for each experiment. For maze and robotic environments, we selected 100 unique sets of hyperparameters from a search space presented in Table 1 using Optuna (Akiba et al., 2019). For each hyperparameter set, we train the model with 3 different seeds and keep the median score. For Atari, we selected 10 unique sets of hyperparameters and train the agent just once.

The method-specific parameters that have not been optimized are presented in Table 2.

The hyperparameters used for the off-policy agent are identical for all algorithms. They are presented in Table 3.

For Atari, we mainly use the setting recommended by Machado et al. (2018). Like (Ecoffet et al., 2021), we use both sticky actions and start no-ops.

B. On the Criticality of Cell Representation in Go-Explore

In Go-Explore, similar observations are grouped into cells and each cell encountered is stored in an archive. The cell representation is a critical aspect of Go-Explore. In the *Montezuma's Revenge* environment, a slight variation in cell representation results in an order of magnitude difference in the results. The cells are used to (1) estimate the density of states encountered in the observation space and sample a target cell against it; (2) divide this goal reaching task into a sequence of subgoals.

We argue that building a cell representation to capture the relevant components of an environment to perform the desired task requires a significant amount of domain knowledge. In general, this cell representation cannot be generalized to other tasks or to other environments.

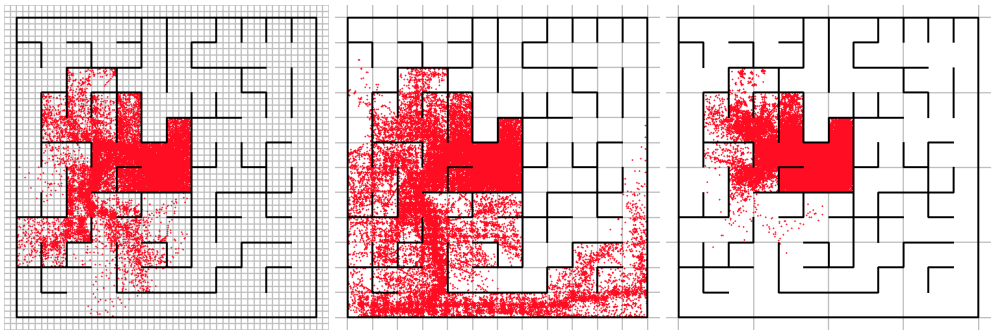
To support our claim, we present in Figure 7 the space coverage in a continuous maze for different cell design. We show that even in this simple environment, a small variation in cell design has a significant impact on the result.

On the left, the cells are small, and the agent must visit each of them. If the agent interacts long enough with the environment, it should eventually explore the whole space. On the right, the cells are large. We can observe some detached areas, because the agent has not visited the cell enough to discover the next one, but enough so that this cell is no longer listed as a target cell.

Table 1. Search space and resulting hyparparameters after optimization.

METHOD	HYPERPARAMETER	SEARCH SPACE	MAZE	ROBOTIC	ATARI
LGE	Latent distance threshold	$[0.1, 0.2, 0.5, 1.0, 2.0]$	1.0	1.0	2.0
	Latent dimension	$[4, 8, 16, 32, 64]$	16	8	$8 \times 8 \times 8$ [c]
	Geometric parameter	$[0.001, 0.002, 0.005, 0.01, 0.02, 0.05]$	0.05	0.01	0.001
GO-EXPLORE	Cell size	$[0.2, 0.5, 1.0, 2.0, 5.0]$	2.0	0.2	$11 \times 8 \times 8$ [b]
ICM	Scaling factor	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{-1}	10^{-2}	N/A
	Actor loss coefficient	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{2}	10^{-3}	N/A
	Inverse loss coefficient[a]	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{1}	10^{-3}	N/A
	Forward loss coefficient[a]	$[10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$	10^{1}	10^{2}	N/A
SURPRISE	Feature dimension	$[2, 4, 8, 16, 32]$	2	16	N/A
	Desired average bonus	$[10^{-2}, 10^{-1}, 10^{0}, 10^{1}]$	10^{-2}	10^{-2}	N/A
	Model train frequency	$[2, 4, 8, 16, 32, 64, 128]$	64	8	N/A
	Model learning rate	$[10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$	10^{-5}	10^{-5}	N/A
DIAYN	Number of skills	$[4, 8, 16, 32, 64, 128]$	32	32	N/A
SKEW-FIT	Number of models	$[5, 10, 20, 50, 100, 200]$	50	50	N/A
	Density power	$[-5.0, -2.0, -1.0, -0.5, -0.2, -0.1]$	-1.0	-0.2	N/A
	Number of pre-sampled goals	$[64, 128, 256, 512, 1024, 2048]$	64	128	N/A
	Success distance threshold	$[0.05, 0.1, 0.2, 0.5, 1.0]$	0.5	0.2	N/A

[a] In the original paper, the sum of the forward and inverse loss coefficients is 1. We get better results without this constraint.
[b] Width $\times$ Height $\times$ Number of grayscale values. Not optimized, taken from the original paper.
[c] Width $\times$ Height $\times$ Number of embedding vectors.



(a) Cell size $= 0.5 \times 0.5$ (b) Cell size $= 2.0 \times 2.0$ (c) Cell size $= 6.0 \times 6.0$

Figure 7. Go-Explore scene coverage after 100k timesteps. The cell design is represented by the gray grid. We show the results for 3 different cell widths and shift. The red dots represent the visited states.

Table 2. Hyperparameters specific to each method. Their value is identical for all experiments and have not been optimized.

METHOD	HYPERPARAMETER	VALUE
LGE	Encoder module trained every N timesteps	5k (Maze and Robotic), 500k (Atari)
	Learning	0.001
	Batch size	32
	Gradient steps	500 (Maze and Robotic), 5k (Atari)
	Exploration strategy	Random
	Repeat action probability	0.9
GO-EXPLORE	Exploration strategy	Random
	Repeat action probability	0.9
ICM	Feature size	16
	Networks	$[64, 64]$
	Activation function	ReLU
SURPRISE	Networks	$[64, 64]$
	Activation function	ReLU
DIAYN	Discriminator networks	$[256, 256]$
	Activation function	ReLU
SKEW-Fit	Gradient steps	100
	Batch size	2048
	Learning rate	0.01

C. On the Density Estimation

Let $s_1, \ldots, s_n$ be a sample of event locations in a $\mathbb{R}^d$-space. Assume that the event location s follows a common distribution with density function $f(s)$. For any sample s_i and s_j in this sample, assume that $D_i(s_j) = ||s_i - s_j||$, denotes the euclidian distance between s_i and s_j. For any $k \leqslant n$, let $D_{(k)}(s_i)$ be the distance with k-th nearest neighbors of s_i with respect to the euclidian distance.

(Kung et al., 2012) propose an optimal unbiased estimator $\hat{f}$ for the density:

$$\hat{f} = \frac{kU_k^*}{k-1} \tag{5}$$

where

$$U_k^* = \frac{(k-1)}{nC_d D_{(k)}^d} \tag{6}$$

and

$$C_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)} \tag{7}$$

Hence we have

$$\hat{f} = \frac{k}{nC_d} D_{(k)}^{-d} \tag{8}$$

We follow the recommendation of (Kung et al., 2012) to take

$$k = 2n^{1/d} \tag{9}$$

D. Comparing Go-Explore and LGE

The Go-Explore algorithm as presented by Ecoffet et al. (2021) has many components. All these components allow to obtain good results on test environments. In this article, we implement our own version of Go-Explore. We have tried to stick as much as possible to the initial implementation and to improve some aspects. We keep the essence of Go-Explore, but our implementation is not intended to be equivalent to the initial implementation. The main goal here is to compare LGE and Go-Explore. Thus, the two implementations differ only in the elements that make them unique. To the best of our knowledge, all the composnts that we did not implement are compatible with LGE. It is likely that they improve LGE and Go-Explore in a similar way. In this section, we describe the implementation of LGE and Go-Explore. We explain their differences if any.

Policy-based Go-Explore The initial implementation of Go-Explore distinguishes between the case where the environment can be reset to any desired state and the case where this is not possible. In this paper, we choose the general setting where the environment can't be reset to any desired state, and we therefore work with the so-called policy-based implementation of Go-Explore.

Exploration after returning In the original implementation of Go-Explore, once a cell is returned, exploration proceeds with random actions for a certain number of timesteps. For both LGE and Go-Explore, we set this number of timesteps to 50 for all environments. Note that the agent can interrupt this exploration beforehand if the maximum num-

Table 3. Hyperparameters of the off-policy agent. These hyperparameters are identical for all methods and for all experiments. The hyperparameters related to HER relabeling only apply to the methods for which the agent is goal-conditioned (DIAYN, Go-Explore, Skew-Fit and LGE).

HYPERPARAMETER	SAC	DDPG	QR-DQN
NETWORKS	$[300, 400]$	$[300, 400]$	CNN from (Mnih et al., 2015)
LEARNING RATE	3×10^{-4}	10^{-3}	5×10^{-5}
LEARNING STARTS AFTER N TIMESTEPS	100	100	1M
BATCH SIZE	256	100	32
DISCOUNT FACTOR (γ)	0.99	0.99	0.99
POLYAK UPDATE COEFFICIENT (τ)	0.005	0.005	1.0
TARGET ENTROPY	2.0	N/A	N/A
TARGET UPDATE EVERY N TIMESTEPS	N/A	N/A	10k
ϵ DECREASES DURING N TIMESTEPS	N/A	N/A	4M
INITIAL ϵ	N/A	N/A	1.0
FINAL ϵ	N/A	N/A	0.05
TRAIN EVERY N TIMESTEPS	1	1	10
GRADIENT STEPS	1	1	1
HER SAMPLING PROBABILITY	0.8	0.8	0.8
HER RELABELING STRATEGY	Future	Future	Future

Table 4. Atari setting.

PARAMETER	Value
RESET ON LIFE LOSS	Yes
START NO-OPS	From 1 to 30
ACTION REPETITIONS	4
STICKY ACTION PROBABILITY σ	0.25
OBSERVATION PREPROCESSING	84×84, grayscale
ACTION SET	Full (18 actions)
MAX EPISODE LENGTH	100k
MAX-POOL OVER LAST N ACTION REPEAT FRAMES	2

ber of interactions with the environment is reached. (Ecoffet et al., 2021) shows that action consistency generally allows for more effective exploration, especially in the robotic environment. For LGE and Go-Explore, we use the same trick: the agent chooses the previous action with a probability of 90%, and uniformly samples an action with a probability of 10%.

Cell design The original implementation of Go-Explore provides two methods for generating the cell representation.

1. When the observation is an image, the observation is grayscaled and downscaled. The image produced is the cell representation. The parameters to get this representation (downscaling width and height and number of shades of gray) are optimized during training to maximize an objective function that depends on a target split factor.

2. When the observation is a vector, each component of the vector is discretized separately by hand before learning.

For all the environments presented in this paper, we use a naive method of cell generation corresponding to a discretization of the observation. The granularity of the discretization is a hyperparameter. The choice of this hyperparameter is crucial, we develop it in more detail in the Appendix B.

Goal-conditioning In the original implementation of Go-Explore, the agent is conditioned by the cell representation of the goal. We note that this representation can vary during learning, and even in size (see previous paragraph). It is not clear how to structure the agent's network when the size of the input varies during the learning process.

In our implementation, we choose to condition the agent by the goal observation rather than by the representation of its cell. We also condition the agent by the goal observation in LGE.

RL agent In the original implementation of Go-Explore, the goal-conditioned agent is based on the on-policy PPO algorithm (Schulman et al., 2017). For both LGE and Go-

Explore, we rather chose to use an off-policy algorithm (SAC or DDPG) to use a Hindsight Experience Replay (HER, Andrychowicz et al. (2017)) relabelling, which has shown to perform better in a sparse reward environment.

Reward In the original implementation of Go-Explore, the agent gets $+1$ reward for reaching intermediate cells and $+5$ reward for reaching the final cell of a path. The rest of the time, it receives a 0 reward. For both LGE and Go-Explore, following the suggestions of (Tang & Kucukelbir, 2021), we rather choose the following structure for the reward: the agent gets a reward of 0 for a success (target cell reached for Go-Explore and latent distance with the goal state below the distance threshold for LGE) and -1 the rest of the time. As noted by (Eysenbach et al., 2019b), in this setting, an optimal agent tries to terminate the episode as quickly as possible. We therefore set `done = True` only at the end of the post-exploration, and not when the agent reaches a final state.