Journal of Artificial Intelligence Research 76 (2023) 1181-1218 Submitted 11/2022; published 04/2023

An Overview of Environmental Features that Impact Deep Reinforcement Learning in Sparse-Reward Domains

Jim Martin Catacora Ocana CATACORA@DIAG.UNIROMA1.IT
Department of Computer, Control and Management Engineering
Sapienza University of Rome, Italy
Roberto Capobianco ROBERTO.CAPOBIANCO@SONY.COM
Sony AI, Zürich, Switzerland
Daniele Nardi NARDI@DIAG.UNIROMA1.IT
Department of Computer, Control and Management Engineering
Sapienza University of Rome, Italy

Abstract

Deep reinforcement learning has achieved impressive results in recent years; yet, it is still severely troubled by environments showcasing sparse rewards. On top of that, not all sparse-reward environments are created equal; in other words, they can differ in the presence or absence of various features, with many of them having a great impact on learning. In light of this, the present work puts together a literature compilation of such environmental features, covering particularly those that have been taken advantage of and those that continue to pose a challenge. We expect this effort to provide guidance to researchers for assessing the generality of their new proposals and to call their attention to issues that remain unresolved when dealing with sparse rewards.

1. Introduction

In recent years, deep reinforcement learning (DRL) has achieved remarkable results in a number of domains, such as: learning to play Go at a superhuman level only from self-play (Silver et al., 2017), defeating professional players in heads-up Poker (Heinrich & Silver, 2016), reaching Grandmaster level in Starcraft II (Vinyals et al., 2019), surpassing the scores of non-expert human players in all 57 Atari games belonging to a famous benchmark suite (Puigdomenech Badia et al., 2020a), as well as performing complex robotic tasks (Levine et al., 2015; Gu et al., 2017; Rajeswaran et al., 2018; Haarnoja et al., 2019).

Despite this success, there are still various settings in which DRL struggles to perform competently. One of these corresponds to sparse-reward environments; that is, domains where rewards are zero everywhere except in few special states. The main problem with such a lack of reinforcement is that it gives DRL agents no reason to prefer one action over another. And this in turn causes them to generally adopt the sensible strategy of trying every action evenly at random, which sadly leads nowhere in large and complex environments. A famous example of an environment with very sparse rewards is the Atari game Montezuma's revenge. In the seminal work of Mnih et al. (2015), this was one of the few games for which DQN showed no competence at all; indeed, this learner never scored above zero throughout training (see Figure 1). For many years, otherwise highly successful

©2023 AI Access Foundation. All rights reserved.

general-purpose end-to-end DRL algorithms (e.g., Hessel et al., 2018; Espeholt et al., 2018; Horgan et al., 2018) were unable to even beat inexperienced human players in this game. It is only fairly recently that generic approaches (e.g., Puigdomenech Badia et al., 2020b, 2020a) have hit that milestone. These state-of-the-art methods report a score of about 11k points, which is more than double that of an amateur player and roughly sufficient to complete the first level of the game. Nevertheless, these results are still far from the best machine learning solution (Ecoffet et al., 2021), which obtains around 40k points without prior domain knowledge; and they are not even close to the human world record of 1.3M points (Ata, 2021).

Over the years, resounding progress was also made by numerous DRL algorithms designed specifically for handling sparse rewards (Bellemare et al., 2016; Shelhamer et al., 2016; Osband et al., 2016; Kulkarni et al., 2016; Pathak et al., 2017; Florensa et al., 2017; Vezhnevets et al., 2017; Racaniere et al., 2017; Machado et al., 2017; O'Donoghue et al., 2018; Stanton & Clune, 2018; Sukhbaatar et al., 2018; Veeriah et al., 2018; Florensa et al., 2018; Nachum et al., 2018; Burda et al., 2019b; Eysenbach et al., 2019b; Choi et al., 2019; Savinov et al., 2019). For instance, RND (Burda et al., 2019b) was the first end-to-end solution to reach the current state-of-the-art score of around 11k points in Montezuma's revenge (a couple of years earlier than general-purpose methods). However, despite their meritorious achievements, one lingering issue that affects most of these algorithms is their lack of generality. In other words, their applicability and effectiveness depend heavily on the specific features of the environment being tackled. This fact has already been pointed out by Taiga et al. (2020), though phrased slightly differently: good performance in a given hard exploration environment is not indicative of good performance in other such environments. To give an example, RND, although outstanding in Montezuma's revenge, performs poorly in worlds that contain distractors or that demand it to retrace its steps. Meanwhile, other algorithms have specialized in exploiting particular environmental features. For instance, reverse curriculum generation (Florensa et al., 2017) is a rather ingenious solution applicable only in domains that are resettable and reversible.

To date, the knowledge of these environmental features that have a manifest impact on the performance of deep reinforcement learning algorithms, especially in sparse-reward environments, is found dispersed among independent studies. Thus, this work brings a contribution by composing an overview of such features, with an emphasis on those whose presence facilitates learning as well as those that make it more challenging. Importantly, this study focuses exclusively on end-to-end single-agent reinforcement learning methods. Therefore, other promising techniques (e.g., imitation learning, advice based learning, multi-agent reinforcement learning) and algorithms (e.g., Baker et al., 2020; Ecoffet et al., 2021) will mostly not be reviewed. Apart from providing a deeper understanding of the intricacies of applying reinforcement learning in domains with sparse rewards, with this work we further expect to offer guidance to researchers for assessing the generality of their newly proposed solutions. To this end, our overview additionally provides detailed descriptions of how (i.e., what mechanisms) and why (i.e., which components of an algorithm confer a weakness) these features affect learning. Then, equipped with such information, researchers could validate, from a theoretical standpoint, if analogous mechanisms unfold within their approaches or if they exhibit the same weaknesses. One last virtue of this study is that it pinpoints several

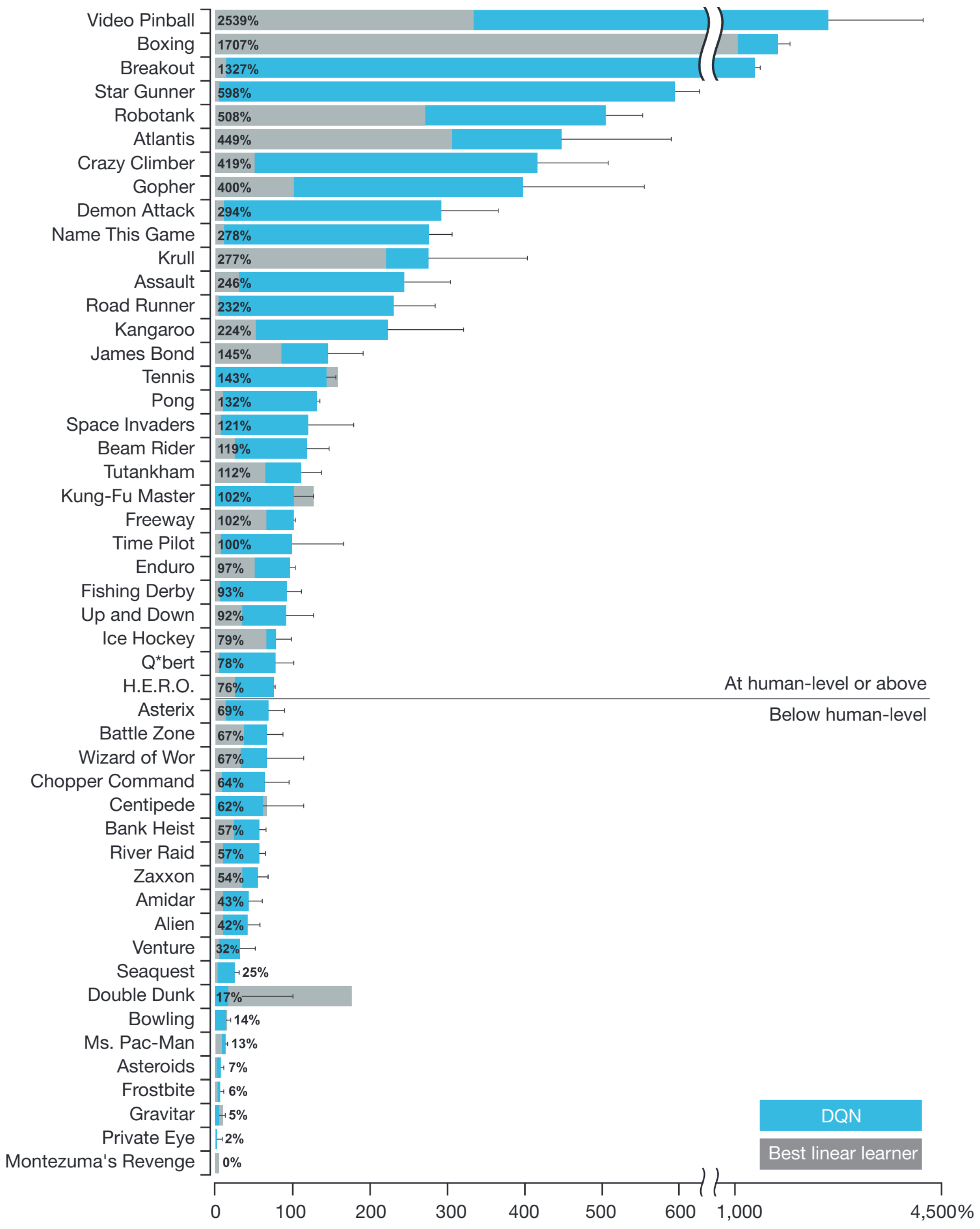


Figure 1: Performances achieved by DQN after 50 million frames of experience across various Atari games (disclosed by Mnih et al., 2015). It can be noticed that while this method is highly successful in several tasks, it makes not the slightest progress in the sparse-reward world of Montezuma's revenge.

open areas of research, which correspond to features whose handling is still out of grasp for existing algorithms.

2. Background

The next paragraphs provide important definitions concerning reinforcement learning.

2.1 Markov Decision Processes (MDPs)

MDPs (Howard, 1960) are a formalism for modeling sequential decision problems that concisely encode the interactions between agents and their environments. An MDP is a tuple $\langle S, A, \mathcal{T}, \mathcal{R}, \gamma\rangle$, comprising:

- The set S of all states an environment can assume, which includes the set $S_0 \subset S$ of initial states and the set $S_G \subset S$ of terminal states,
- The set A that contains every action an agent can execute,
- A probability distribution over transitions $\mathcal{T} : S \times A \times S \to [0, 1]$ that determines the next state given the current state and action,
- A reward function $\mathcal{R} : S \times A \times S \to R \subset \mathbb{R}$ that quantifies with a real scalar value the immediate goodness of a transition,
- And a discount factor $\gamma \in [0, 1]$ that weighs down the value of future rewards.

An MDP additionally satisfies the Markov property (Sutton & Barto, 1998), such that the next state and reward depend solely on the state and action at the current time step t, meaning that the following equality holds true for all feasible pairings of $r \in R$ and $s' \in S$:

$$\Pr\{R_{t+1}{=}r, S_{t+1}{=}s'|S_t, A_t\} = \Pr\{R_{t+1}{=}r, S_{t+1}{=}s'|S_0, A_0, R_0, ..., S_t, A_t\} \tag{1}$$

Unfortunately, it is often the case that an agent perceives only a small region of its world at a time (i.e., it has partial observability). In those scenarios, rather than an MDP, it is more convenient for analysis purposes to phrase the corresponding decision process as a partially observable MDP (POMDP), which is expressed as the tuple $\langle S, A, O, \mathcal{T}, \mathcal{R}, \gamma\rangle$. Where: O is the set of all observations, $\mathcal{T}$ is a probability distribution over transitions $\mathcal{T} : S \times O \times A \times S \times O \to [0, 1]$ that determines the next state and next observation given the current state and action, while $S, A, \mathcal{R}, \gamma$ maintain the definitions offered for an MDP.

Relative to an MDP, another important concept is that of a policy, which is a generally stochastic mapping from states to actions $\pi(a \in A|s \in S)$ that fully encodes the decision-making behavior of an autonomous agent. A similar notion also exists in a POMDP; however, here it is best to be more general and instead represent a policy as a mapping from past observations, actions and rewards to the present action.

2.2 Reinforcement Learning

Reinforcement learning (RL) is concerned with learning in MDPs without having an a priori model of the domain (transition and reward functions). The main characteristics of RL are that learning is accomplished through raw experience by executing actions and receiving rewards, and that learning is synonymous with the maximization of rewards collected over time (Sutton & Barto, 1998). The most common objective of RL is to find at least one of possibly many optimal policies π^* for the underlying MDP. In this context, an optimal policy is one that in every situation yields the highest sum of rewards, typically defined in terms of the expected cumulative discounted reward (also known as expected return).

Formally, the return G, associated with a single rollout in which an agent is in state s after t time steps and thenceforth enacts policy π to receive a sequence of future rewards $R_{t+1}, R_{t+2}, ...$, is expressed as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0} \gamma^k R_{t+k+1} \tag{2}$$

Naturally, the corresponding expected return is computed as the expectation $E[\cdot]$ over all possible rollouts involving policy π and state s, as it is seen next:

$$E_\pi[G_t|S_t = s] = E_\pi\Big[\sum_{k=0} \gamma^k R_{t+k+1}|S_t = s\Big] \tag{3}$$

The term $E_\pi[G_t|S_t = s]$ is also called the value of s under π. A function $v_\pi(s)$ that encodes this information for every state in S is known as a value function. By virtue of the previous definitions, an optimal policy π^* is then any one whose values are greater than or equal to those of every other policy for every state, that is: $v_{\pi^*}(s) \geq v_\pi(s), \forall s \in S \wedge \forall \pi \in \Pi$; where Π denotes the space of all plausible policies.

The former notion of value can also be extended to every state-action pair in $S \times A$. This produces an extremely helpful function called an action-value function or q-function, which is formalized as:

$$q_\pi(s, a) = E_\pi\Big[\sum_{k=0} \gamma^k R_{t+k+1}|S_t = s, A_t = a\Big] \tag{4}$$

Given the previous framework, different approaches for solving RL problems then diverge mainly on how they address the optimization task of finding a policy that maximizes the expected return. A few broad methodologies include: temporal difference (Sutton, 1988), policy gradient (Williams, 1992; Sutton et al., 1999a) and actor-critic (Konda & Tsitsiklis, 2000).

When the state or action space of a problem is too large or continuous, it is no longer feasible to properly perform such an optimization in a tabular manner. In those cases, many researchers have turned to deep reinforcement learning, which, in a nutshell, leverages deep neural networks to better approximate the value functions or policies we wish to learn.

2.3 The Difficulty of Learning with Sparse Rewards

Whether working with classical or deep reinforcement learning, we will run into serious issues when the chances of seeing a reward are next to nothing. To be more concrete, here we refer strictly to the luck enjoyed by a uniform policy; that is, one in which every action has an equal probability of being selected. Such low odds can emerge in a number of domains, the most common situation being when every non-zero incentive is extremely far from any initial state (reward sparsity caused by distance). But other scenarios are possible too; for instance, when a nearby reward has a fleeting existence spanning merely a few time steps (reward sparsity caused by time).

In this work, we call any domain where the probability of reinforcement is minimal a sparse-reward environment; and we can easily see why these are problematic. If every incentive received by a learner is zero; then, from its perspective, the expected return has to be flat. And here we have already hit a roadblock, because there is no meaningful way to maximize a flat function; mathematically, it is an ill-defined problem. Or from the point of view of a gradient descent optimizer, there are no gradients that could sensibly guide the maximization of such a function. Arguably, the most rational thing to do at this point is to consider that every state is equally rewarding, and ergo, that every action is equally good. In other words, to develop an optimal policy that behaves uniformly. Ultimately, such a policy traps us in an eternal loop of disappointment, as by our initial definition it has virtually no chance of discovering an incentive; and hence, of wrinkling the landscape of the expected return.

Notably, the previous recount perfectly illustrates how most reinforcement learning algorithms respond to the absence of rewards, owing to two reasons. One, they are designed with the sole objective of maximizing the expected cumulative discounted reward. That is, the very deed we just said is impossible to do in a sparse-reward environment. And two, they are endowed with far too basic exploration strategies (e.g., ϵ-greedy, Boltzmann exploration, entropy regularization or action space perturbations). These, in the presence of a flat expected return, likewise devolve into selecting actions evenly; therefore, they are of no help in getting learners out of their entrapment.

3. Approaches to Reward Sparsity

DRL researchers have followed a wide range of avenues to find satisfactory solutions to sparse-reward tasks. Some of these miscellaneous ideas include. The use of **reward bonuses** (oftentimes called intrinsic rewards) that are added to, and thus reformulate, the reward function of the original task (denoted as task or extrinsic reward). These bonuses need to be dense (i.e., non-zero in almost every state) in order to overcome the sparsity of rewards and hence promote an extensive exploration of the environment. Several generally applicable interestingness measures have been proposed as bonuses, such as: learning progress (Schmidhuber, 1991), novelty (Bellemare et al., 2016; Burda et al., 2019b), curiosity (Stadie et al., 2015; Pathak et al., 2017; Houthooft et al., 2016) or empowerment (Mohamed & Rezende, 2015). In most cases, these measures are computed from experiences encountered across training episodes, but some methods also compute them from intra-life experiences seen only within each individual episode (Stanton & Clune, 2018; Savinov et al., 2019). The generation of an **automatic curriculum** that presents a sequence of tasks of increasing reward sparsity to the learning agent, corresponding to simplifications of the original problem. In this way, the learner can reuse the expertise acquired in simpler tasks to solve more difficult ones. Such a curriculum has been implemented in a variety of forms. For example, by requesting the agent to revisit a previously seen state or observation selected at random from the entire history of past trajectories (Veeriah et al., 2018). Similarly, by soliciting the agent to reach states or observations (i.e., goals) specified by another model. The latter can be a second interacting policy simultaneously trained to set challenging goals (Sukhbaatar et al., 2018) or a generative adversarial network trained to produce goals of intermediate difficulty (Florensa et al., 2018). Another idea is to modify each episode's initial task configuration. Specifically, to start the agent close to a known solution state in the first episode of training and to start it in some past state (sampled randomly from a memory) in each subsequent episode (Florensa et al., 2017). The adoption of **hierarchical learning** over skills, options or goals. A skill is a mapping from subsets of states to subsets of actions or other skills that produces a distinct behavior. An option is a special case of a skill, formally defined by three components: a policy, an initiation set of states and a termination condition (Sutton et al., 1999b). Learning a high-level policy directly over these abstractions instead of primitive actions can improve exploration in sparse-reward domains as the interacting agent can reach further with fewer decisions. Numerous methods acquire such hierarchies within two stages; first they discover skills and then they learn a policy over those skills. Plenty of techniques for skill discovery exist. And these encompass ideas as disparate as rewarding a set of policies for being distinguishable from each other (Gregor et al., 2016; Eysenbach et al., 2019b) or extracting skills from the graph Laplacian associated with the task's transition matrix (Machado et al., 2017). Studies focused on options usually learn a predefined finite number of them concurrently with a policy over options that orchestrates which option is active and when (Bacon et al., 2017). Everything is then trained from the combined experiences of all options and thanks to the policy gradient theorem (Sutton et al., 1999a). Analogously, other hierarchical algorithms simultaneously learn a goal-conditioned policy together with a policy that controls the previous one via goals, which are commonly expressed as states to reach, (Vezhnevets et al., 2017; Nachum et al., 2018; Levy et al., 2019). Such a policy over goals is responsible for maximizing extrinsic rewards while the controlled low-level policy is intrinsically rewarded for fulfilling its commanded goals. Other original solutions, briefly described next, comprise. The construction of a set of policies optimized for **state covering** (Lee et al., 2019). The use of approximations to **Bayesian uncertainty** for guiding deep exploration (Osband et al., 2016; O'Donoghue et al., 2018; Sekar et al., 2020). The **self-imitation** of an agent's own past trajectories (Oh et al., 2018), especially those identified as profitable or unfamiliar. The self-supervised assimilation of **actionable representation or distance models** that internalize the number of actions needed to move between any two states of the world, and whose distance information is eventually transformed into shaped rewards (Ghosh et al., 2019; Florensa et al., 2019).

4. Impactful Environmental Features

The following subsections elaborate on a number of environmental features that according to our review of the literature have noticeable effects on the performance of reinforcement learning algorithms developed for dealing with sparse rewards. In each case, we provide a definition of the given feature, useful examples and a summary of previous works that have already looked into such features. Particularly, every study included in the next subsections makes one of the following contributions. Exploits the occurrence of a feature to facilitate exploration in sparse-reward problems. Exposes how an algorithm that is otherwise successful at handling a lack of external reinforcement fails when a certain feature is present. Or, delivers a description, expressed as an hypothesis or supported by empirical evidence, of the

Feature	Definition	Examples
Resettability	Designers have control over each episode's initial state	Simulated robotic environments are usually resettable
Reversibility	Being able to return to a previous state during an episode	An unclimbable cliff dividing the world into two strata is an irreversibility
Retracing steps	Having to revisit past states during an episode	Searching for a key and then returning to a locked door under partial observability
Partially ordered subtasks	Having to perform diverse skills with any ordering being feasible	Multiple radially-expanding culs-de-sac, each posing a distinct challenge
Distractors	Dynamic elements that are observable but irrelevant to the main task	Leaves moving in a breeze
Stochastic dynamics	Performing the same action in the same state does not always lead to the same outcome	Sticky actions
Action-prediction degeneracy	Diverse actions lead to the exact same outcome	Pushing a rigid block into the ground
Deadly states	States that produce the early termination of an episode	Enemies, lava floors, falling from high elevations, etc.
Procedural generation	Each episode has a slightly different layout but the abstract objective remains the same	Varying geometries, textures, lighting, etc.

Table 1: Summary of environmental features contained in this study.

mechanisms by which a feature impacts the machinery devised to conquer reward sparsity. Table 1 gives a preview of all the features we have compiled in this study, indicating also several prototypical domains that internalize them.

4.1 Resettability

A resettable environment lets researchers configure any valid state as the initial state of a given episode. For example, robotic simulators readily position a robot according to any desired joint angle and velocity configuration (see Figure 2). As this property explicitly allows to reformulate the original task, it has been exploited by numerous automatic curriculum approaches. And thanks to that, they all turned out to be capable of exploring whole environments while receiving at most just one extrinsic reward at the end of a successful episode. For instance, Florensa et al. (2017) enforce a reverse curriculum by altering the initial state distribution of a problem. In particular, this distribution at first is set to occupy a small volume around a terminal goal, which must be known beforehand. But afterward, it is consented to gradually grow with every new state encountered, as this algorithm learns to reach said terminal state from further and further away. Ultimately, the ambition of this method is to engulf inside such a distribution the entire state space associated with a given task, while simultaneously developing a policy capable of solving the latter starting from anywhere in the world. In practice, this method maintains in memory a set of previously seen states, which when training starts is manually populated with a few states located nearby the known goal. This technique then leverages resettability by forcing each episode to initiate in a state sampled uniformly at random from the previous set. Similarly, Eysenbach et al. (2019a) establish a curriculum that pushes the learning agent to master its environment locally. Effectively, this algorithm trains a goal-conditioned policy to connect any two states that are close to each other. And to this end, in each episode it samples an initial state (from anywhere in the world) plus a goal state (forced to be within 4 interaction steps of the start 80% of episodes), and then resets the environment to the former. In this way, during testing, if the given goal is determined to be near the start, the learned policy is directly asked to reach such a target. Else, planning is applied to decompose the task by defining an ordered set of waypoints (i.e., states outlining the shortest path to the target). Of which, the first one is issued to the policy at the beginning of the episode and the rest are dispensed one by one each time the immediate sub-goal is achieved. Figure 3 demonstrates how this method (called SoRB) takes advantage of resettability to outclass a powerful general-purpose approach such as C51 (Bellemare et al., 2017). Sukhbaatar et al. (2018) also rely on the aforementioned feature to generate a self-play curriculum (in one of their proposed variants at least). Such a curriculum emerges from the interaction between two learning agents: a setter and an imitator. The setter proposes a task by executing a trajectory and it is rewarded when the task is too complicated for the imitator, which in turn is rewarded for getting to the same final state as the setter. It is in virtue of this competition that these agents increasingly explore more of their environment. Critically, for this to work best, the imitator must be reset to the same or a very similar initial state as the setter, which ensures that the difficulty of self-play tasks is always the right amount.

4.2 Reversibility

In a reversible environment, if going from state A to state B is feasible for an agent, then moving from state B to state A is also feasible for any two states (see Figure 4). Moreover, in such domains, just like in any standard RL problem, the agent has no a priori knowledge of the transition function, and ergo of the action or the sequence of actions that would allow it to return to a previous state. Irreversibility is not an uncommon feature; it appears in many benchmark environments used to test reinforcement learning. For instance, it happens in videogames (e.g., when an avatar falls off a cliff that it cannot climb back up or when a car agent rolls over because of an accident) and in robotic tasks (e.g., when a manipulator throws an object outside its workspace). Reversibility is a necessary condition for the applicability of the proposal made by Florensa et al. (2017), which when met along with resettability produces outstanding results (see Figure 6.) As reported in the previous subsection, this algorithm performs exploration backward, starting from a known goal. However, if all trajectories connecting the unadulterated initial state distribution to that target are irreversible (e.g., the goal is at the bottom of an unclimbable cliff, as depicted in Figure 5), then such a reverse curriculum will never solve the problem in hand. Sukhbaatar et al. (2018) too exploit reversibility in another of their self-play curriculum variants. As already described, this method creates a competition between a setter and an imitator. And in this further variant, after the setter suggests a task, policy control is switched to

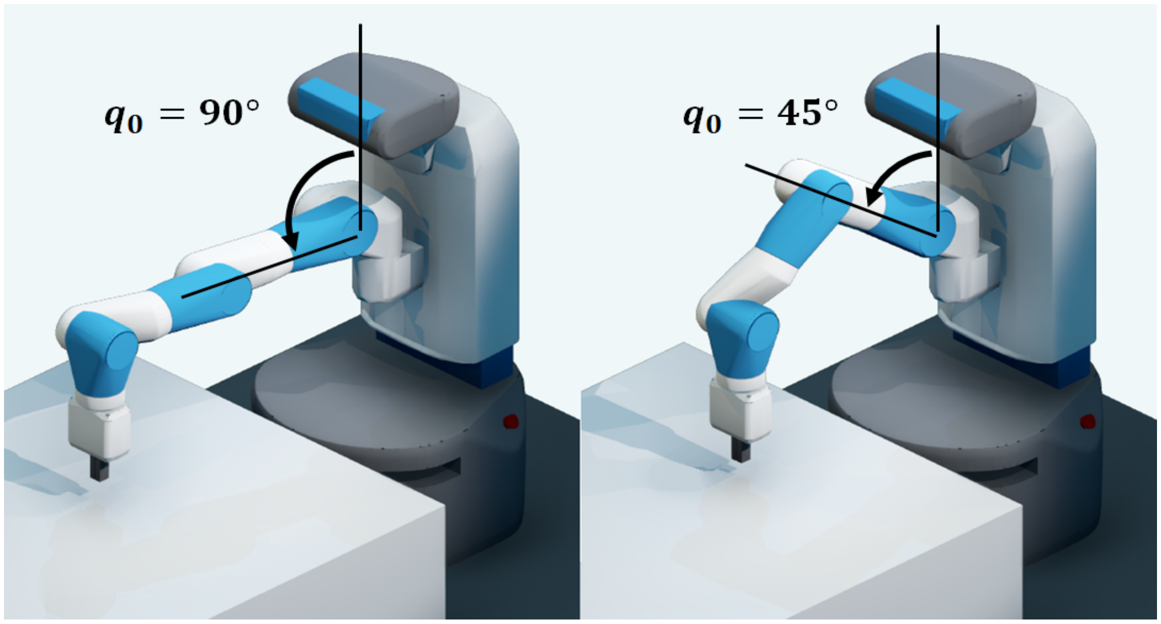


Figure 2: Simulated manipulator environments usually allow the resetting of their initial state to any desired joint configuration.

the imitator, which is rewarded for moving from the current state to the setter's initial state (note that resettability is not required here). Hence, this approach is sensible only in reversible domains, since in the opposite case the setter would be compelled to always propose impossible tasks. Another work that puts reversibility in the spotlight is the one formulated by Grinsztajn et al. (2021). To be clear, this algorithm operates properly independently of whether or not an environment has irreversibilities. Notwithstanding, concurrently to a policy, it trains a neural model to predict if a transition between two states is reversible or not. For that, it uses the temporal ordering found in past trajectories as ground truth. Said model is then used in one of two ways. To constrain exploration by forbidding irreversible actions. Or, to guide exploration even in the presence of reward sparsity by allocating large internally-generated positive bonuses to transitions currently predicted to be easy to reverse.

4.3 Retracing Steps

A learning agent is forced to retrace its steps in any environment that demands revisiting some states before reaching its goal. For an illustrative example, consider an agent that although being near a locked door, to open it, it must first search elsewhere for the corresponding key. Crucially, the agent has to navigate extensively through the world before finding said key, and then, to return to the locked door, it has no choice but to revisit the same locations seen along the path leading to the key. In a successful episode, the agent gets one positive reward for opening the door and possibly another for collecting the key; no other state or transition gives an incentive. Scenarios like the previous one are problematic for a wide range of algorithms designed to thoroughly explore sparse-reward domains. Specifically, the fact that

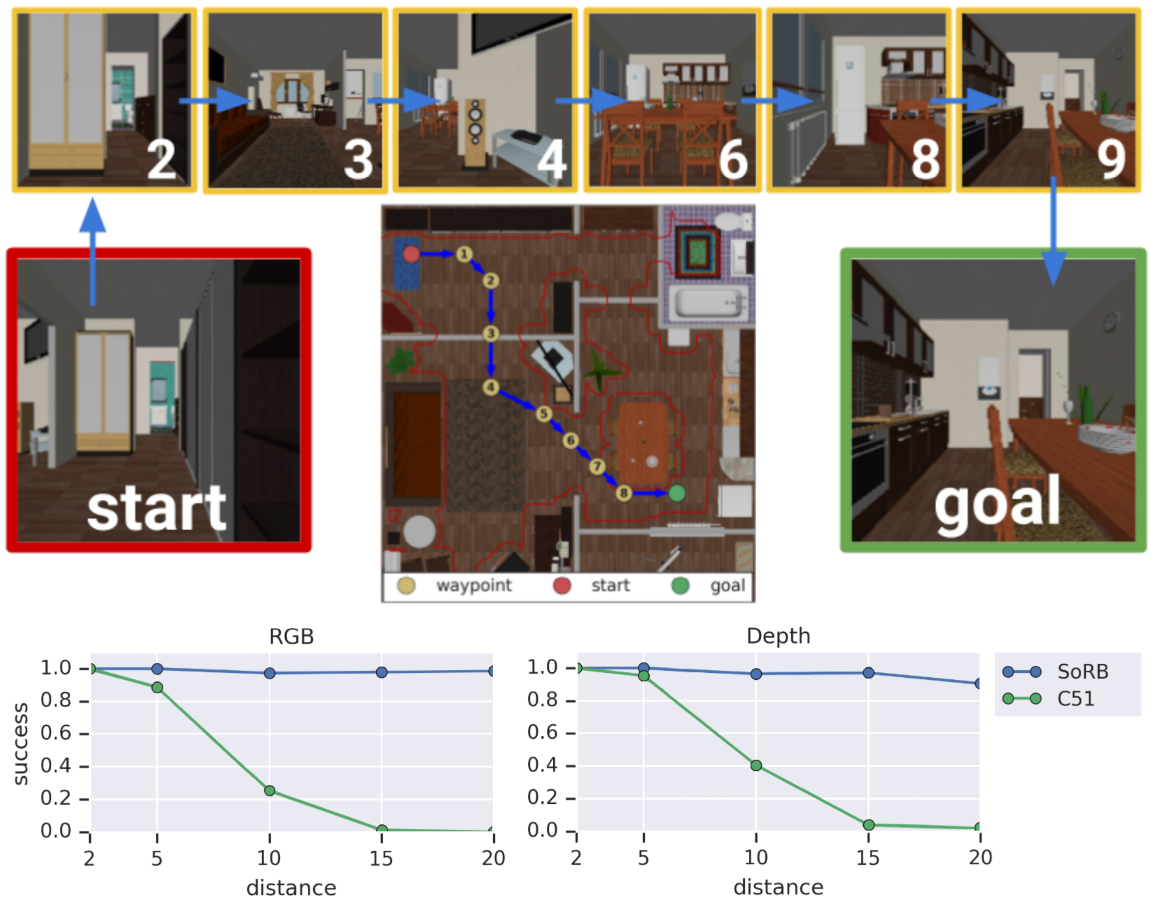


Figure 3: Comparison between SoRB and C51 in a sparse-reward visual navigation domain (found in Eysenbach et al., 2019a). The task (top) requires an agent to traverse a virtual 3D house until arriving at a goal observation presented to it before the simulation initiates. A positive reward is only handed when the target is reached. The graphs at the bottom report the success ratio attained when observations are RGB images (left) and depth images (right). Their x-axes indicate the distance between start and goal, measured in terms of interaction steps. These results clearly expose the superiority of SoRB, which as the distance to the target increases performs progressively better than C51.

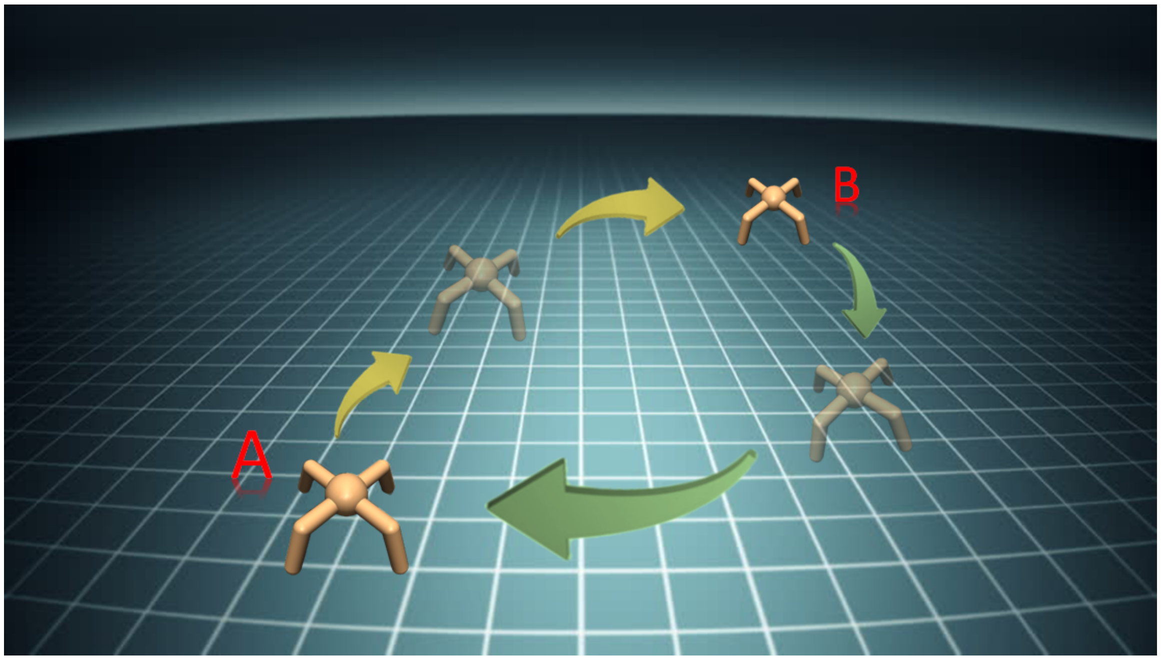


Figure 4: A flat and infinite arena is reversible since an agent that navigates from A to B sees no obstacle that forbids it to move in the opposite direction, from B to A.

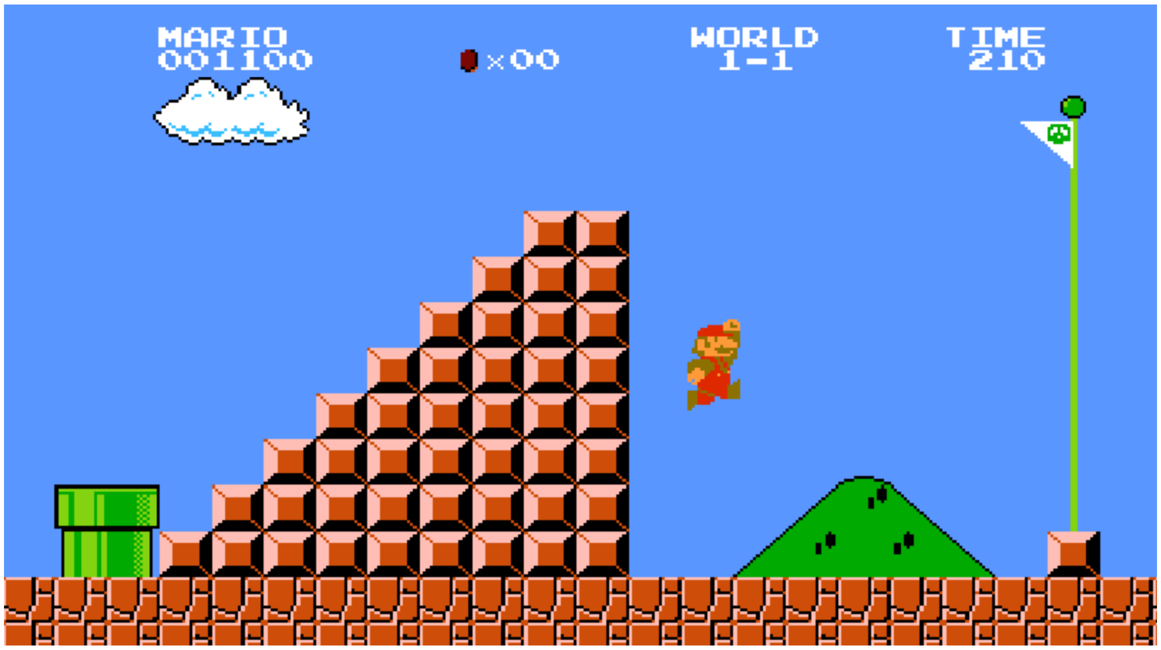


Figure 5: A irreversibility is present in the first level of the original Super Mario Bros, as Mario cannot jump back up the triangular structure located next to the goal pole.

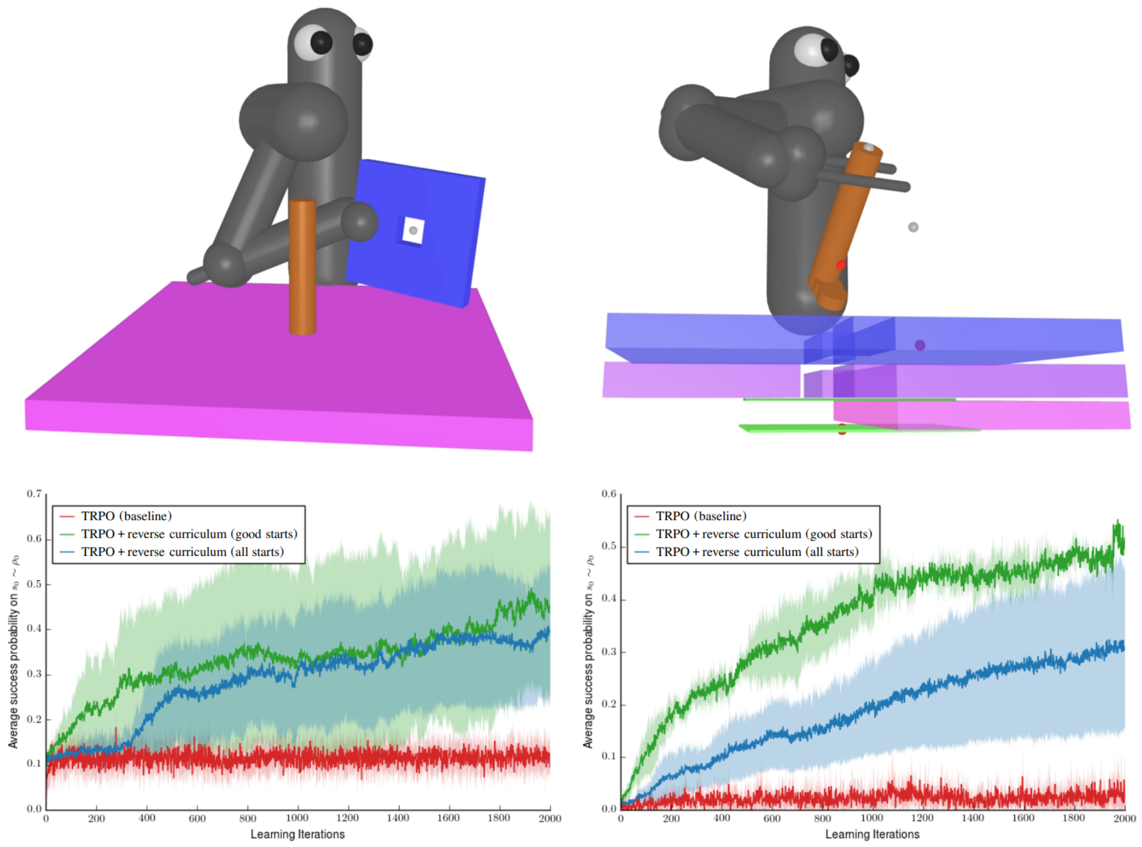
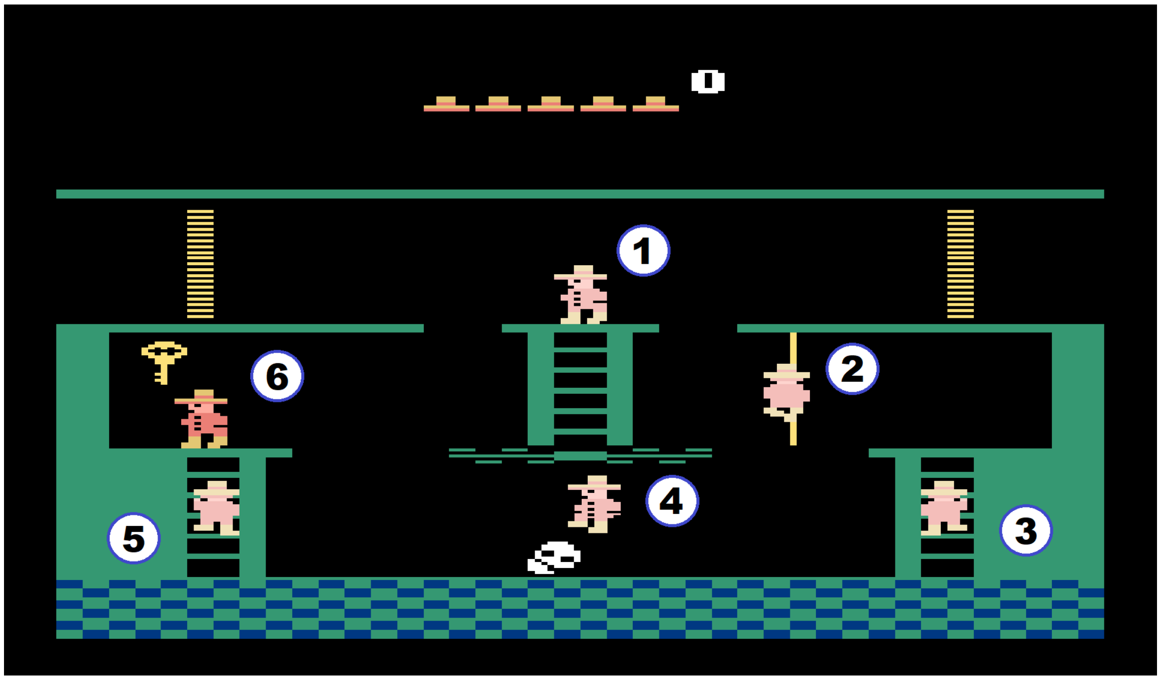


Figure 6: Results published by Florensa et al. (2017), comparing their method against TRPO (Schulman et al., 2015). The tasks are: ring on peg (left) and key insertion (right). Their objectives are self-explanatory from their names; the former task requires a 7 DOF robot to place a square disk on a round peg (the task is complete when the disk is within 3 cm of the bottom of the 15 cm tall peg). While the latter asks the same robot to execute a complex maneuver before properly inserting a key into a keyhole (i.e., it must first insert the key at a specific orientation, then rotate 90 degrees clockwise, push forward, then rotate 90 degrees counterclockwise). In both cases, agents receive joint angles as inputs (the disk and the key are welded to the hand of the robot; hence, extra information about their poses is not necessary), and are positively rewarded only when finishing a task. Additionally, the original initial state distribution of both environments is uniform over their entire state spaces (this is what TRPO faces). The graphs at the bottom display the evolution of the success ratio as training progresses. They tell us that while an algorithm like TRPO, which lacks any mechanism for dealing with sparse rewards, is hopeless in these robotic tasks. When the same technique is coerced to follow a reverse curriculum that exploits resettability and reversibility, its competence then skyrockets.

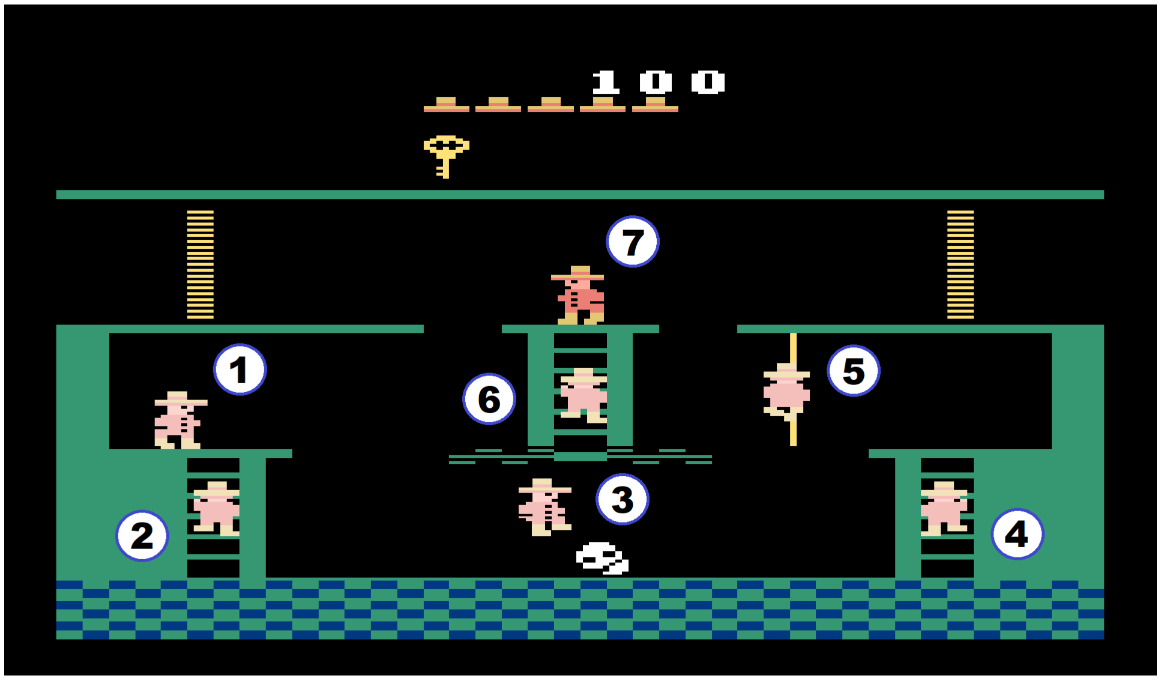
two very similar states (in fully observable environments) or even two identical observations (in partially observable environments) correspond to two fundamentally different states (e.g., before and after collecting the key) can very well confuse the exploration machinery of said methods. They could wrongly assume that states belonging to the return trip have already been visited and that would result in them halting exploration prematurely. For instance, Roderick et al. (2018) showed that the issue of retracing steps severely impacts algorithms that employ across-training novelty-based bonus rewards. This work evaluated one of these approaches, namely DQN-CTS (Bellemare et al., 2016), in a version of Montezuma's revenge modified such that the interacting agent is conceded just one life instead of six per episode. Note that the initial room of this game compels the agent to first find a key, then backtrack to its initial position and finally leave the room by unlocking one of two doors (as shown in Figure 7). This experiment reveals that DQN-CTS cannot escape the first room of the game in the modified setup. The authors argue that this occurs because such a learner is unable to disassociate a state seen before collecting the key from another seen afterward if the avatar occupies the same spatial location in both. They are just too alike for this learner to tell them apart, and as a result, it assigns an equal bonus to both of them; which is true for every location along the path between the agent's initial position and the key. A direct consequence of this bonus conflation is that no round-trip policy can ever procure a higher return than the most optimal outbound policy. And this is ultimately the most likely cause of this learner never leaving the initial room; it has little to none motivation to retrace its steps. Conversely, see that when the avatar is given six lives instead of one as in the original setup, a second strategy for escaping the initial room becomes viable, which involves killing itself right after getting the key. An action that instantly places the avatar back in its initial position while still holding the key. Thus, as backtracking is no longer mandatory here, the same DQN-CTS learner is now quick to learn to commit tactical suicide and eventually ends up discovering around 15 rooms per run (see Figure 9). According to Dann et al. (2019), the inability of the previous algorithm to differentiate between pre- and post-key states even in the fully observable first room of Montezuma's revenge is due to the small size of the key. Therefore, we should expect the problem of retracing steps to be more prominent in partially observable environments, since it is no longer conditioned by the size of things in them (see Figure 8). On top of that, methods driven by across-training novelty generally treat observations the same way they treat states and allocate bonuses directly to them (i.e., no internal states, no memory). That is, they lack any mechanism that would allow them to temporally discriminate between two indistinguishable observations within an episode.

4.4 Partially Ordered Subtasks

This feature is visible in domains that request the learning agent to carry out multiple distinct subtasks during a single episode, with many of them starting from the same state. An archetypal environment is one where the agent is initially located in a central spot from which a number of culs-de-sac expand radially outward (see Figure 10). Each cul-de-sac imposes a different subtask as it urges the mastering of a specific ability (e.g., swimming, climbing) and the full task involves visiting every one of them all the way to the end. An important detail is that the environment provides no indication of the completion of any subtask. And only when the agent fulfills the whole task, it receives a reward. We say that

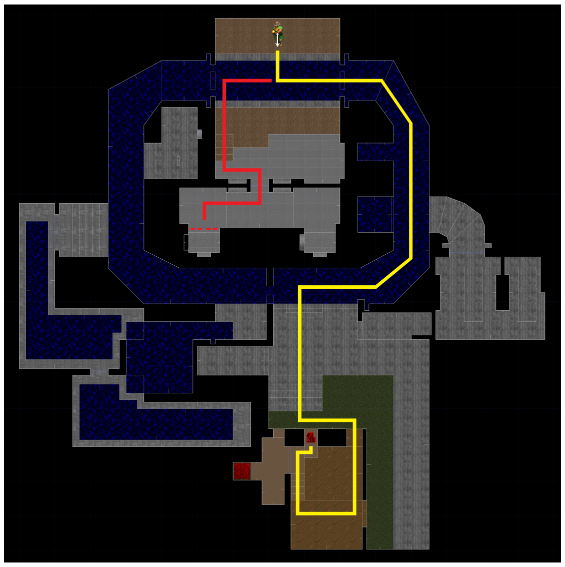


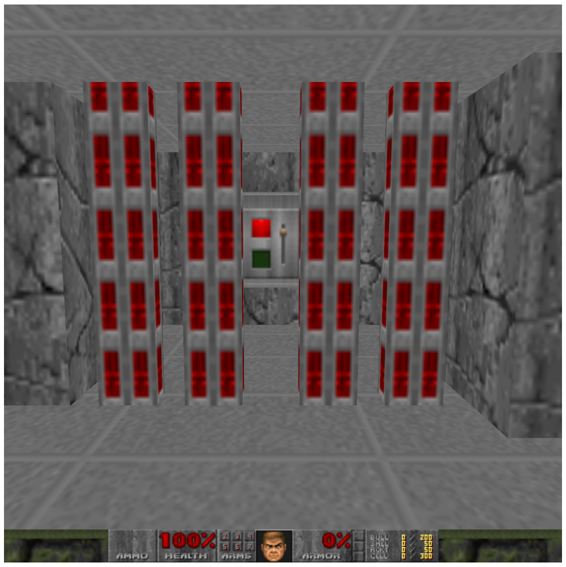
(a) Panama Joe's journey to the key on the left.

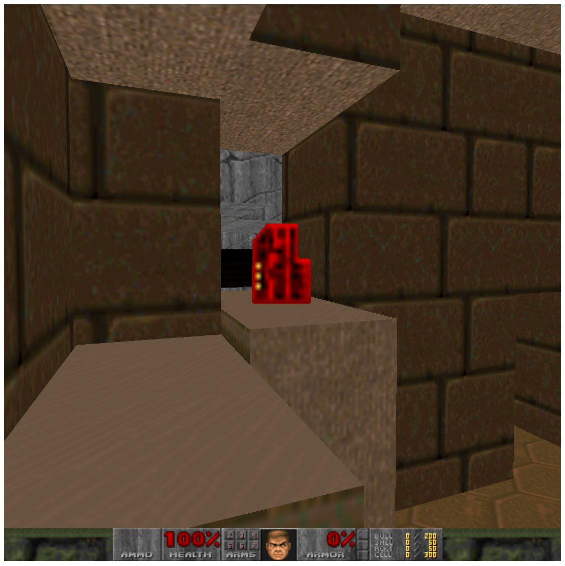


(b) Panama Joe's journey back to his starting position.

Figure 7: First room of Montezuma's revenge. To leave it, and if life loss must be avoided, Panama Joe has no choice but to retrace his steps after collecting the key that opens any one of the two yellow doors at the top.


(a) Underhalls map.


(b) Red gate is initially locked.


(c) Red card key is found.

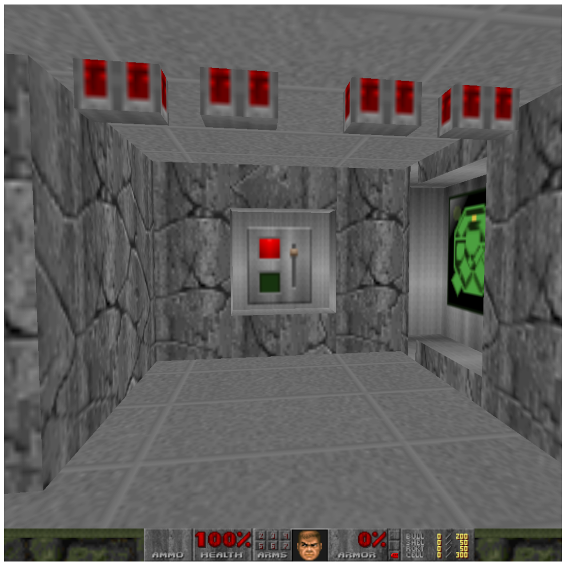

(d) Red gate is opened with the red key.

Figure 8: The second level of the classic first-person (i.e., partially observable) game Doom 2, named Underhalls, requires the doomguy to unlock a red gate. As seen in the level's map, this gate is close to the doomguy's initial position (red path). But to open it, he is forced to first travel forth and back through several other sectors of the world (yellow path) in order to retrieve a red card key.

the previous set of subtasks is partially ordered because the agent is free to attempt them in any order at any point in time. Nevertheless, the solution to the problem may or may not depend on the order in which these subtasks are performed. Stanton and Clune (2018) reason

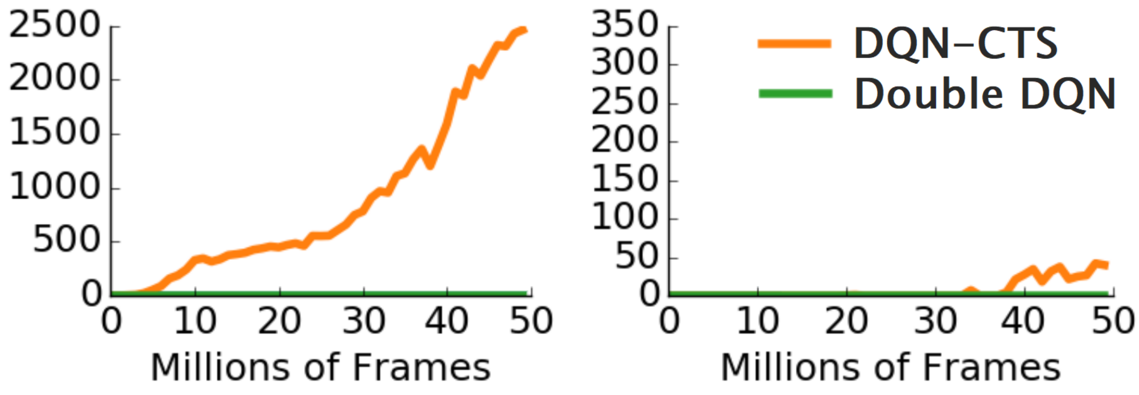


Figure 9: Scores achieved by a couple of algorithms in Montezuma's revenge when the avatar is given 6 lives (left) and one life (right) per episode (reported by Roderick et al., 2018). We see that in a difficult sparse-reward environment such as this one a technique like Double DQN (Hasselt et al., 2016), which has not been specifically designed to handle an overwhelming lack of reinforcement, proves to be utterly incompetent. In turn, a DQN-CTS agent works fantastically when backtracking is not compulsory (in the standard setup with 6 lives), but it barely gets any points when it must retrace its steps (in the single-life setup).

about various mechanisms by which domains comparable to the former example can pose a complication for across-training bonus-based algorithms (e.g., Bellemare et al., 2016; Pathak et al., 2017; Burda et al., 2019b). First of all, for most such methods, if not all of them, their bonus function remains essentially unchanged in the short term (during a single episode). In other words, if the learning agent stands still, it will receive the same bonus at each time step. This has a dramatic consequence for said algorithms in the presence of partially ordered subtasks that do not signal subtask completion. Namely, their optimal policy is always one that visits the most rewarding cul-de-sac and stays there until the episode terminates. Under such circumstances, it is unclear whether or not these techniques are at all capable of learning a policy that concatenates isolated abilities in a controlled manner within an episode. A second difficulty is catastrophic forgetting (French, 1999), which is when a neural model loses an acquired ability in response to not performing it recently. Across-training bonuses are usually designed to decrease over time as the learner keeps reaching the same states or repeating the same transitions (this happens in the three methods previously given as an example). In this context, environments with partially ordered subtasks are especially interesting since they open up the possibility for such approaches (and in fact any approach) to focus on mastering a single subtask at a time. Assuming that happens, these algorithms may for instance choose to dedicate themselves solely to learning how to swim at the beginning of training, becoming good at it after some time. At that point, the bonuses accredited to swimming would have decreased so much that it would be more rewarding to stop doing it entirely and start focusing on another activity, like climbing. And here is where catastrophic forgetting becomes a problem, it would obviously take these

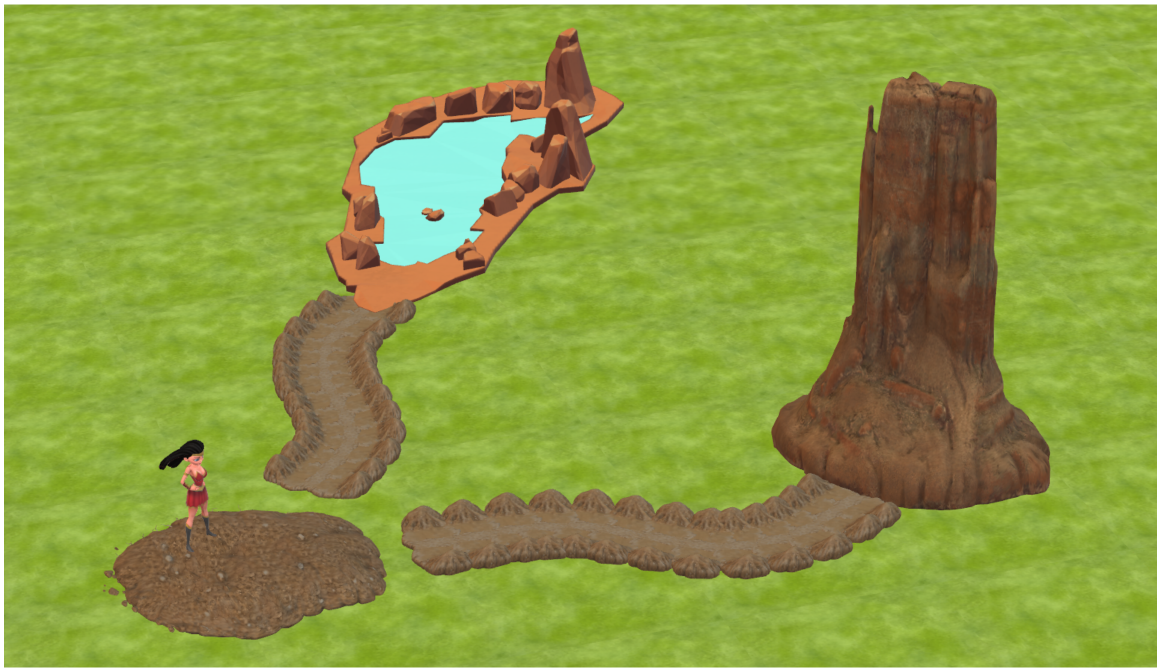
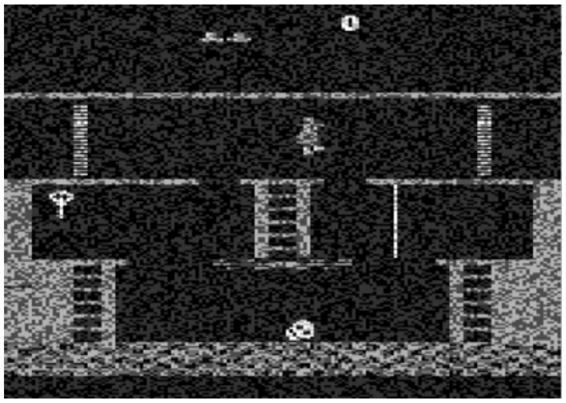


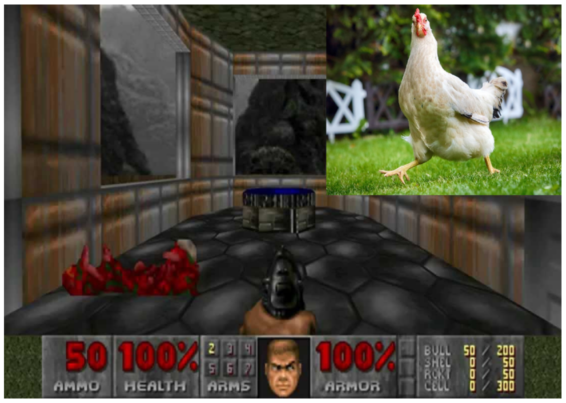
Figure 10: Idealized scenario that consents an agent to freely explore different subtasks, such as swimming or climbing.

learners another significant amount of time to pick up climbing, and by the time they do so they would have forgotten all about swimming. Ultimately, catastrophic forgetting can be expected to push these algorithms to become specialists, which perform one subtask with competence and are clueless about all the others. In addition, catastrophic forgetting would also affect the exploration models of such methods, making them lose memory of having visited any other cul-de-sac but the one currently being paid attention to. One last challenge put by environments displaying partially ordered subtasks arises when they further demand that the latter be performed in a specific order to get a reward. For this analysis, let's also consider that the prior issues of learning a composite policy and catastrophic forgetting have been resolved. In such a scenario, if we again assume that methods guided by across-training bonuses can fixate on one subtask at a time, then it would be possible though undesirable for them to focus on the wrong subtask first. For example, while a problem requires to first go swimming and then go climbing, these learners may unfortunately decide to master climbing first. After becoming proficient climbers, they will start getting interested in swimming, which will concentrate higher bonuses by now. Given our initial premises, these algorithms will then be driven to concatenate climbing and swimming, in that order, within the same policy and episode. Thus, they will learn to swim while still going climbing at the start of every episode. And that means that from this point onward the visitations and bonuses associated with climbing will never become less and higher, respectively, than those associated with swimming. In the end, these methods will not have enough incentive to go climbing a second time after going swimming and the full task will not be solved.

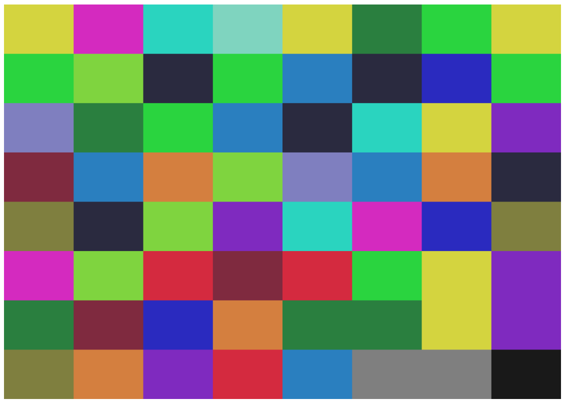
4.5 Distractors

A distractor is a dynamic element that can be observed by the learning agent but is irrelevant to the resolution of the task in hand. Distractors come in different flavors: stochastic or deterministic (usually with extremely complex or even chaotic dynamics), controllable or uncontrollable. Pathak et al. (2017) offer the scenario of an agent observing the movement of tree leaves in a breeze as a very compelling mental image of a distractor. The motion of millions of leaves is undoubtedly a complex dynamical system that has an enormous number of feasible configurations and whose dynamics is impossible to predict purely from visual information. On top of that, the agent cannot manipulate the wind or the leaves directly; hence, such observed dynamicity transpires entirely beyond its control. Figure 11 portrays other examples of distractors. Generally speaking, distractors introduce two important complications into a reinforcement learning problem: the addition of plenty of purposeless states as well as transitions and the unpredictability of their dynamics. Given the generality of these two concerns, distractors have the potential to affect virtually any method known to tackle sparse rewards. Notwithstanding, all experimental evidence found in the literature regarding the detrimental presence of distractors is related to bonus-based approaches. It is clear why these methods experience such hardship when facing distractors. To circumvent reward sparsity, they generate dense bonuses grounded on the idea of compensating agents for finding or learning something new in the environment (e.g., novelty, curiosity). Therefore, since distractors can be designed to produce endless new observations and to embody unlearnable-to-perfection dynamics, they can easily attract the complete attention of such algorithms and make them ignore the actual task in full. For example, Kim et al. (2019) and Song et al. (2020) added pixel-level white noise to each image observation received by an agent playing Montezuma's revenge (see Figure 11(a)). In addition, the dynamics of this noise was kept independent of the agent's actions (i.e., it could not turn it off or tamper with it). Results from both these works show that the performance of RND (Burda et al., 2019b), an across-training novelty-driven approach, falls sharply to nearly zero when such an uncontrollable stochastic distractor is in place (see Figure 12(a)). And these two studies agree that such degradation is likely a consequence of the continuous stream of new noise configurations, which artificially maintains novelty high in every state, and thus, motivates said learner to stay still. Another distractor with similar characteristics was proposed by Pathak et al. (2017). In this case, an agent was asked to solve a sparse-reward navigation task in a visually rich 3D environment. And the distractor was a large region of white noise stitched next to each image observation sent to the agent, which again had no control over this distractor. The cited study experimented with an across-training curiosity bonus, computed directly as the prediction error of a learned one-step forward dynamics model (i.e., a model that given the latest observation and action predicts the next observation). And it reports that such a bonus engenders a policy that reaches the navigation goal merely 60% of the time. The authors argue that the impossibility of predicting white noise with zero error keeps this curiosity bonus slightly up everywhere and that discourages the behavior policy from traveling far. Burda et al. (2019a) and Savinov et al. (2019) investigated the effects caused by controllable stochastic distractors. Both works adopted a visual navigation domain, which was augmented with a noisy TV that played the role of a distractor. This TV was implemented as a region of the environment that always displays some picture retrieved from a large dataset. In the first work, this TV was present only in one location of the environment (a wall whose view could be blocked by others); while in the second, it was placed on top of every image observation, covering the same area each time (as shown in Figure 11(b)). Importantly, in either case, the learning agent was endowed with one extra action that allowed it to change the station of this noisy TV at will. In fact, whenever the agent chose that action, the current TV picture was replaced with another sampled uniformly at random from the corresponding dataset. The two studies evaluated the response of ICM (Pathak et al., 2017) to this noisy TV. To incite exploration, ICM relies on an across-training curiosity bonus that, exactly as in a method mentioned before, is equal to the prediction error of a forward dynamics model. But contrastingly, the observation embeddings going into that model are shared with an inverse dynamics model trained in parallel (i.e., a model that given the current and next observations infers the action taken). Because of that, those embeddings will only encode information about elements the agent can control, which makes ICM immune to uncontrollable distractors (as proven by its authors). In the end, the findings of both works are consistent with each other, and demonstrate that ICM is not suitable against a noisy TV (see Figure 12(b)). This is explained by noticing that information about the noisy TV necessarily leaks into the shared embeddings since it improves the accuracy of the inverse dynamics model, and that negates all the benefits of incorporating this extra model. In other words, ICM becomes aware of this TV and, consequently, it gets pathologically attracted to its constantly surprising transitions. Catacora Ocana et al. (2022) note that previous works have heavily banked on the difference in dynamics between distractors and their encompassing domains to design algorithms tailored to ignore dynamic elements that are uncontrollable or have highly stochastic dynamics. To blur the line between distractor and environment in terms of dynamics, this study proposed two new benchmark tasks containing controllable deterministic distractors. In both cases, once again, the main task involved visual navigation and the distractor was a TV that covered the whole field of view of the interacting agent, which was given one additional action to turn the TV off or on. One TV was rendered as an 8 by 8 grid where each cell could take one of 16 colors (see Figure 11(c)). Seeing that the total number of TV stations is immense, the agent was supplementary provided with 4 special actions that allowed it to navigate to any station within an episode. Furthermore, the dynamics of this distractor was made chaotic in the sense that starting from the same TV station two sequences of actions disagreeing only on the first one will generate two completely different sequences of stations. Tests carried out by the authors of this benchmark with RND and ICM reveal that both algorithms invariably neglect the navigation task and fixate purely on this TV. These two approaches exclusively learn simple policies that move straight forward in some general direction of the TV space while almost never observing the same station or transition twice during training. And that happens thanks to the stochasticity of the policies themselves and the chaotic regime of this distractor. The second TV was rendered as a 4 by 4 grid where each cell could be one of two colors. Here, the agent was granted just one extra action to interact with the TV; hence, it could ever experience only one sequence of stations. The online version of EC (Savinov et al., 2019) was evaluated in this second scenario. EC incentivizes exploration by means of an intra-life curiosity bonus. In a nutshell, EC learns a reachability model that predicts if two states are less or more than a predefined number of actions apart from each other. With the help of that model, in each episode it keeps a memory of states that when first encountered


(a) White noise applied to Montezuma's revenge.



(b) Noisy TV displaying animal pictures.


(c) One station of a controllable deterministic TV.

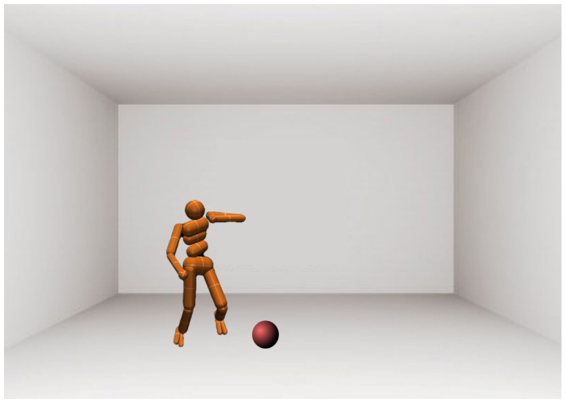
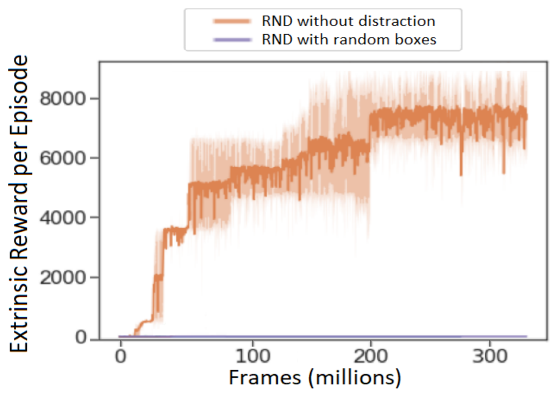

(d) An ordinary ball can be a powerful distractor.

Figure 11: Miscellaneous distractors.

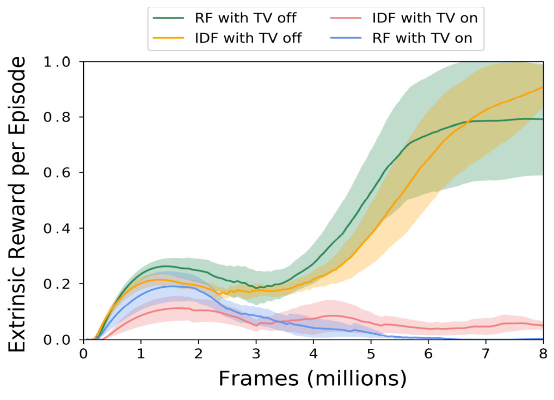
are far from every state already in memory and assigns a large bonus to them. Ultimately, such a distance constraint enables EC to successfully handle a noisy TV. Unfortunately, when the authors confronted EC with their second deterministic TV, this method became utterly mesmerized by it. The reason is clear, EC quickly learns the trivial policy of always changing the station, which collects many bonuses, but it is just a local maximum.

4.6 Stochastic Dynamics

As seen in the previous subsection, the existence of elements with stochastic dynamics can be a serious inconvenience to some exploration strategies. Sadly, such elements are not limited to act as distractors, and indeed, they can be an inextricable component of the task we wish to solve. A good example is the introduction of sticky actions into games of the Arcade Learning Environment (Machado et al., 2018). Sticky actions are implemented simply by defining that the action ultimately executed by an avatar is chosen probabilistically between the one commanded by the learning agent and the action performed in the preceding interaction step (see Figure 13). This means that, from the learner's perspective (which is only ever aware of the action it instructed), facing sticky actions is equivalent to facing



(a) Montezuma's revenge + random boxes.

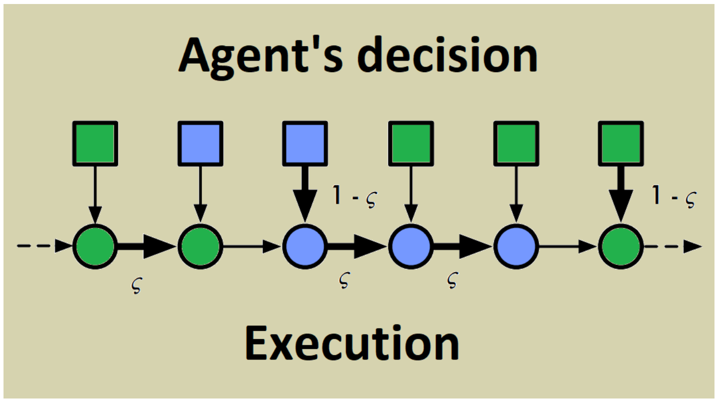


(b) 3D Maze + noisy TV.

Figure 12: Two experiments demonstrating that algorithms developed to work under sparse rewards are sometimes troubled by distractors. The one on the left, run by Kim et al. (2019), evaluates RND in Montezuma's revenge without and with a distraction. The latter is framed as multiple 7x7 boxes filled with pixel-wise noise that can randomly pop up anywhere within each image observation. The presented graph convincingly tells us that, even though this algorithm does fairly well in the standard setup of the game, it just cannot handle an uncontrollable stochastic distractor. Meanwhile, the experiment seen on the right, described in Burda et al. (2019a), examines a pair of methods that harness curiosity as a way to improve exploration. One is in fact ICM, which, as we know, extracts features from observations with the help of a trained inverse dynamics model (thus, these features are called IDF). The other simply replaces the observation embeddings of ICM with random features (RF), which are generated by a neural network whose weights are initialized at random and then are kept fixed the entire training session (they are never updated). Both these techniques were tested twice in the same static 3D maze (containing a single task reward of +1 given at its goal); once without and another time with a noisy TV. The results of these 4 runs leave no doubt about the detrimental effects of this TV on these methods. Without it, they comfortably learn a policy capable of reaching the goal most of the time; yet, when such a distractor is present, they appear completely lost.

an ubiquitous stochastic distractor; that is, one that partakes of every transition. And an immediate consequence of this is that when sticky actions are employed it is impossible to learn a perfectly accurate forward dynamics model of any part of an environment. Burda et al. (2019a) express their concern regarding the limitations under stochastic dynamics of across-training curiosity-driven methods, which rely on the prediction error of a forward dynamics model to compute the bonuses that help them overcome reward sparsity. These include algorithms built on a rather straightforward usage of the prediction error (Stadie et al., 2015), but also those applying more elaborated schemes such as ICM (Pathak et al., 2017). As stated earlier, the unpredictability of the world is detrimental to this family

(a) Atari joystick.



(b) Visual representation of sticky actions.

Figure 13: The implementation of sticky actions emulates a well-known issue occurring in physical controllers where buttons have a tendency to get stuck. As the diagram shows, given two actions (green and blue), at each time step either the one selected by the learning agent (squares) gets executed or the one from the previous step (circles) is repeated, with probabilities $1 - \zeta$ and ζ respectively.

of methods, and when the dynamics of the environment is all-around stochastic it can cause curiosity bonuses everywhere to remain high. A phenomenon that may hinder the development of far-reaching exploration policies. In practice, though, these learners will still conduct some exploration; yet, its overall performance in terms of coverage and sample efficiency will degrade considerably. That stochastic dynamics, and sticky actions specifically, has a negative impact on a technique along the lines of ICM is emphatically validated by Figure 15. There we see the outcomes of two sets of experiments, executed by the authors of this document, in which said algorithm was asked to solve the maze navigation problem depicted in Figure 14. In half of the trials, the avatar was restricted to always enact the actions commanded by the learned policy (non-sticky scenario) and, in the other half, it was allowed to sometimes ignore such orders and repeat its previous move instead (sticky scenario). Critically, in both experimental setups the environment and the learner were configured in exactly the same way. Every single hyperparameter was identical except obviously for the probability of sticky actions taking place, which was set to 0.0 and to 0.25 in the former and latter scenarios, respectively. These results evidence that the advanced exploration capabilities offered by ICM are categorically undermined by sticky actions. In their absence, this technique worked as desired; it discovered all rooms and sectors of the maze while mastering a policy that arrives at the goal every time. Yet, in the sticky setup, not only it never found the jewel, but it also barely made it out of the initial room; meaning that around 80% of the world remained uncharted even after 40 million frames of experience.

4.7 Action-Prediction Degeneracy

The problem of predicting which action caused a given transition is said to be degenerate when there is no single answer. Such a situation emerges in environments whose dynamics

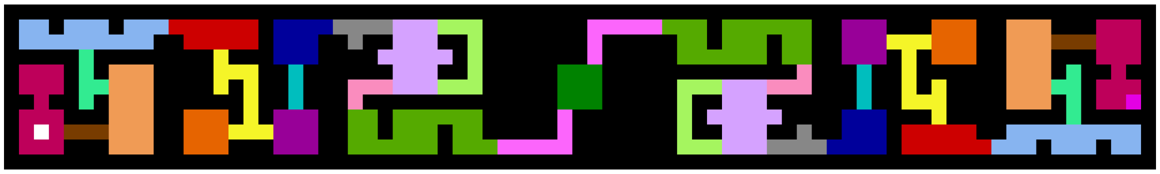


Figure 14: Environment used for testing the effects of sticky actions on ICM. At its core, this domain is a maze structured as a 11x77 gridworld. It is further divided into seven 11x11 rooms and 35 sectors (groups of adjacent cells of the same color). The task to be completed here is the following. Starting randomly from any cell within the carmine sector of the leftmost room, the avatar (single white cell) must navigate every room and at least 31 sectors of the maze to collect the jewel (single magenta cell) located in the rightmost room. The moment this gem is taken, the learning agent is handed a reward of +1 and the episode is terminated. No other extrinsic reward exists in this domain. Plus, if the jewel remains out of reach, the ongoing episode automatically ends after 1000 time steps. To interact with this world, the agent is allowed to see only the room it is currently in (i.e., it has partial observability). In particular, every observation it receives is a 44x44 image, which means that each cell of the maze is represented as a 4x4 patch of solid color. In terms of actions, the avatar is entitled to travel exactly one cell up, down, right or left per time step (4 discrete moves in total). There is no noop action and, furthermore, to collect the jewel, the avatar just needs to step over it. Importantly, if the agent tries to move into an impassable wall (black cells), nothing happens, it simply stays in the same cell.

permit more than one primitive action to produce the exact same transition. The classical example is that of an agent pressing a solid block straight into the ground or against a sturdy wall; no matter what the strength of the applied force is, the block will not move (see Figure 16). This kind of degeneracy becomes a worrying issue for any algorithm that internalizes an inverse dynamics model anywhere within the machinery that allows it to thoroughly explore sparse-reward environments. That is, a model that given a sequence of states or observations gathered from a past trajectory tries to postdict the actions performed by the interacting agent at each transition. Models of this sort have been employed in previous works for different purposes. For example, Haber et al. (2018) use one such model to generate a dense curiosity bonus that guides exploration. Specifically, each transition is assigned a bonus reward equal to the prediction error of an inverse dynamics model that is learned alongside a behavior policy. This motivates such a policy to visit those regions of an environment where its control is less understood. Moreover, since this model is continuously updated from trajectories executed by the learning agent, the bonus of a region will tend to decrease as visitations to that region increase. In an ideal scenario, this exploration mechanism would explore an entire environment, starting with regions closer to the domain's initial state and moving to further away regions as the bonuses of the former ones drop to zero. Unfortunately, that idyllic vision is shattered when degeneracy renders it impossible to predict with perfect accuracy which action provoked an observed transition. That keeps

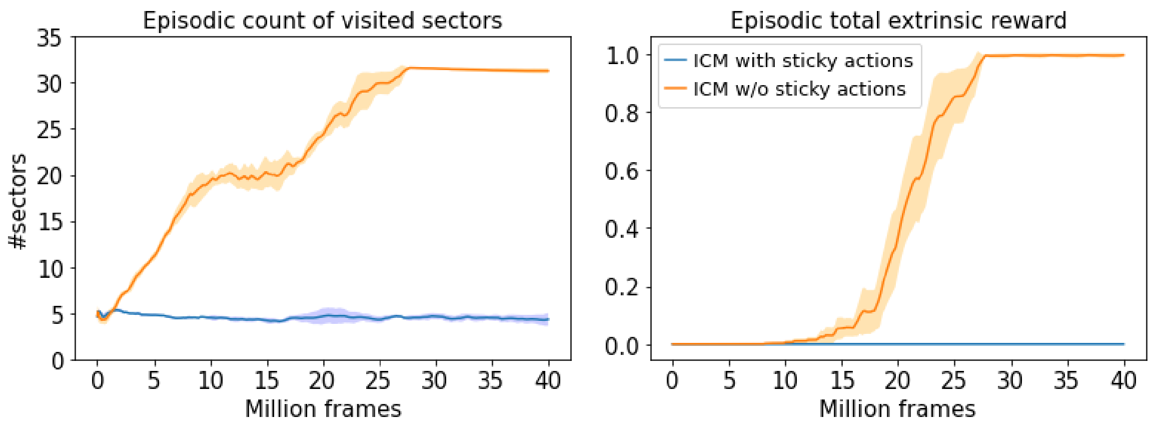


Figure 15: Evaluation of ICM in the maze navigation task shown in Figure 14, in both the presence and absence of sticky actions. For these experiments, the implementation of ICM replicated the setup employed by Pathak et al. (2017), but with the upcoming modifications. During preprocessing, grayscale conversion and frame stacking of 4 were still applied (to produce 44x44x4 observations); yet, no action repeat was enforced. The last layer within the embedding network of both, the policy and the curiosity module, was assigned a stride of 1 and no padding, which resulted in representation vectors of dimensionality 512. Accordingly, the hidden layers of the forward and inverse dynamics models as well as the output layer of the former were also enlarged to 512 units. Within the policy, the previous embeddings were sent, rather than to an LSTM, to a stack of 4 dense layers, each with 512 neurons and ReLU activation. PPO was used as base learner; its parameters were configured as follows: 16 workers, 4 epochs per update, 4 mini batches per epoch, rollout length of 80 steps per worker, γ of 0.99, λ of 1.0, clipping ϵ of 0.05, value function coefficient of 0.5 and entropy coefficient of 0.01. Optimization was carried out by Adam with a learning rate of 5×10^{-5} and a maximum gradient norm of 40. Lastly, all hyperparameters of the curiosity module remained unchanged except for β, which was lowered to 0.005. The graphs shown here reveal what happens when the previous instantiation of ICM is tested in the abovesaid maze environment under two regimes: without sticky actions and with them (with probability ζ=0.25). Both scenarios were run 3 times, each with a different seed. Correspondingly, these curves report means and variances across those 3 runs of simple moving averages over 5 million time steps computed at the level of each individual training session. These results demonstrate a stark contrast between the performances of ICM due to sticky actions. When the latter were not integrated into the task, ICM managed to learn a strategy that gathers the jewel in nearly every episode. Conversely, in the sticky scenario, this same algorithm not once in 40k episodes landed on said jewel. Even worse, on average it frequented merely 5 sectors of the maze (i.e., just the entire initial room), leaving the remaining 6 rooms and 30 sectors mostly unexplored (barring a few short-lived shallow incursions into the second room).

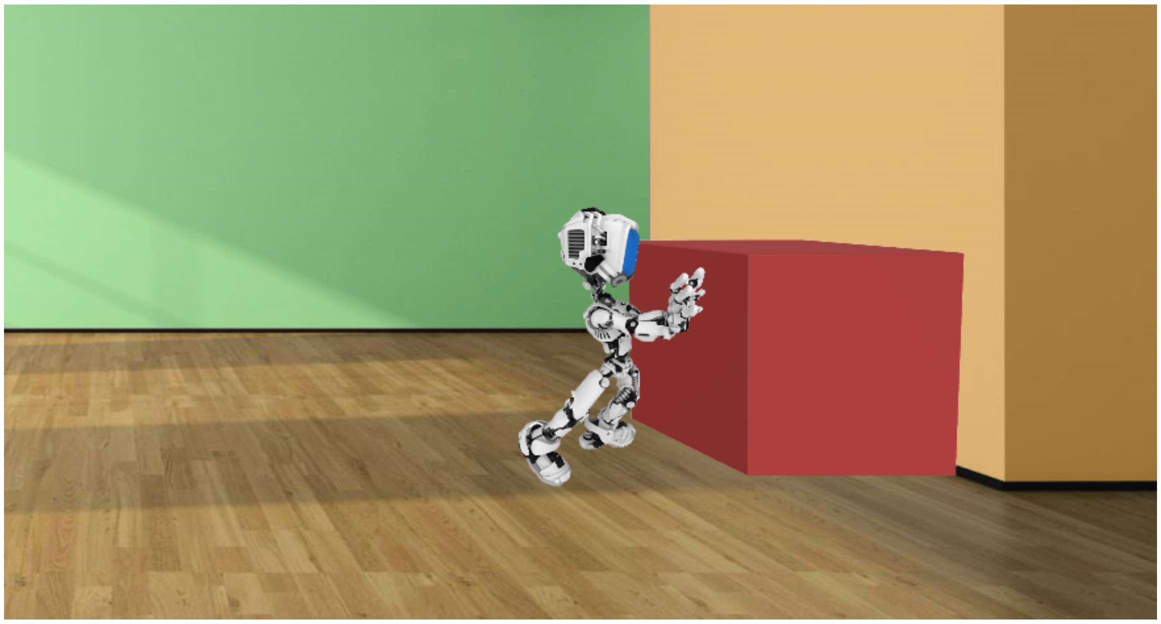


Figure 16: Robot hopelessly pushing a solid block into a thick concrete wall. Even as it varies the magnitude of the exerted force, no effect is perceptible.

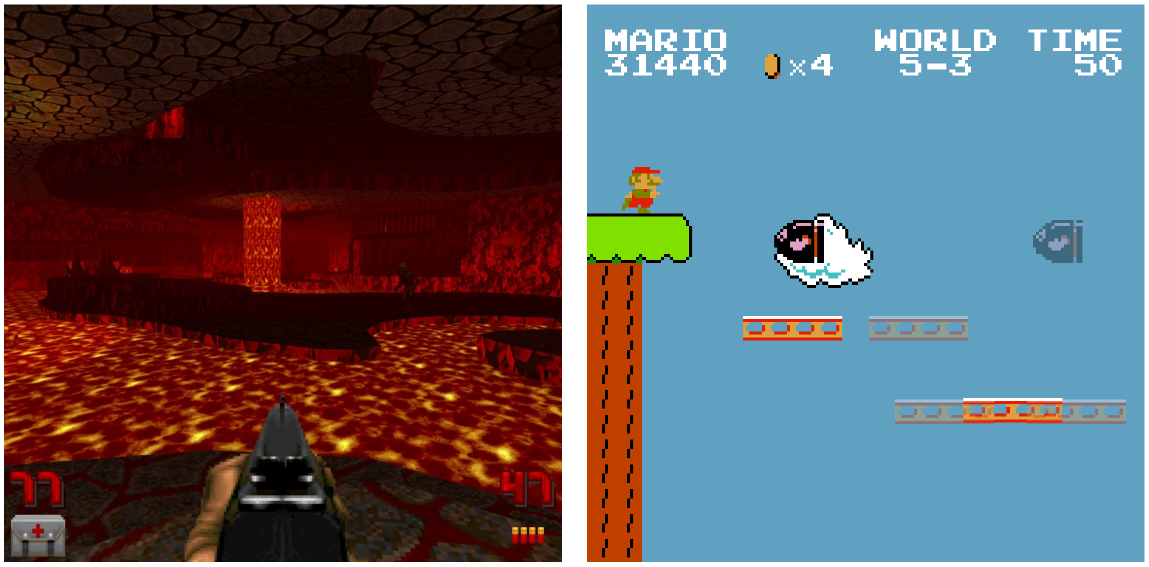
bonuses relatively high all over the world (even near the initial state), which ultimately disincentivizes the agent from exploring remote locations in the environment. Pathak et al. (2017) also compute a curiosity bonus that depends on a one-step inverse dynamics model. However, this time the prediction error of such a model has no direct influence on the value of the bonus. Instead, this incentive is proportional to the prediction error of another forward dynamics model, which is trained concurrently with everything else. And the role of the inverse model is purely to ground a representation shared by both models. Again, a degenerate action-prediction problem might have some impact here, since this algorithm has no explicit mechanism that would negate the occurrence of a fluctuating biased policy. That is, a policy that arbitrarily favors some actions over others at degenerate transitions and whose preference flip-flops from one update to the next (as doing so changes nothing). In the end, the variability in the training data induced by such a volatile behavior would cause all models to oscillate and converge slower. In other words, the inverse dynamics predictor, its representation and the policy itself would take longer to settle down than otherwise would in a non-degenerate domain of similar complexity (e.g., with the same number of actions or transitions).

4.8 Deadly States

This feature's definition is quite self-explanatory from its moniker (a couple of examples are shown in Figure 17). Some environments simply possess states where an episode immediately terminates if the interacting agent visits them, and such termination can happen much earlier than the episode's timeout. Furthermore, in sparse-reward environments, the agent usually gets no recompense when falling into these deadly states. The presence of these states can strongly disrupt the exploration behavior of various approaches created to handle reward sparsity. Indeed, it should not be surprising that a learner that easily covers the entire state space of a domain without deadly states fails to do the same or works much more inefficiently when such states are added. For example, Dann et al. (2019) present the notion of risky exploration. This situation occurs when an environment expects the learning agent to acquire a complex skill (i.e., a sequence of a few actions highly unlikely to be performed by chance), but small deviations from a perfect execution lead to life loss. The authors argue that such a setting appears, for instance, as an agent tries to jump over the first moving skull in Montezuma's revenge, which additionally requires the agent to move toward the deadly skull before doing the jump. According to them, risky exploration poses a serious challenge to approaches relying on across-training reward bonuses that decrease in response to increasing visitations (e.g., Bellemare et al., 2016; Pathak et al., 2017; Burda et al., 2019b). For these agents, seeing that they are internally compensated for every time step they survive, dying also means that they will miss out on collecting a sizable cumulative bonus. Therefore, they have a big incentive for staying alive, even if that means putting the surveying of the environment on the back burner. Going into more detail about the mechanics of it all; as these algorithms gradually explore their environment, states or transitions lying just before a recently discovered life-threatening obstacle will necessarily be associated with higher bonuses. Consequently, such learners will be motivated to revisit that deadly obstacle. Then, after a few failures they will realize that the obstacle is not easily surmountable and that their best local policy is to reach its edge and then sit tight, taking no risky actions while soaking up the bonus of that zone. But as visitations to the edge increase, the bonus there will get depleted, until eventually it will be more rewarding to stop a little before the edge and soak up the bonus of preceding states or transitions. This process will repeat over and over again, pushing these learners further and further away from the deadly obstacle. Still, at some point, these methods will become interested again in the obstacle and the cycle will restart. All in all, the phenomenon of risky exploration induces, at best, a highly inefficient exploration, and at worst, a decisive impasse in which deadly states are never overcome because of a shortage of experiences.

4.9 Procedural Generation

Some environments dictate that every episode is to present a drastically different scenario to the interacting agent in terms of: map configuration (geometries, textures, colors, etc.), existing elements (keys, doors, deadly states, distractors, etc.), dynamics, among other attributes. But that at the same time, all episodes are to convey the same task at an abstract level (e.g., maze navigation). In practice, this is achieved by means of a dedicated procedural generation code that randomizes over a list of attributes and their parameters to create a unique episode every time. Because of this randomization no two episodes will be alike in these domains (see Figure 18). And in addition, it will be highly unlikely to observe the same state or transition twice throughout training. If on top of all that such environments also have sparse rewards, then we get a very ambitious and exciting problem: learning exploration strategies with generalization capabilities. Presently, since these domains have been little studied so far, it is unclear whether or not existing algorithms of any flavor are



(a) Lava rivers in Doom.

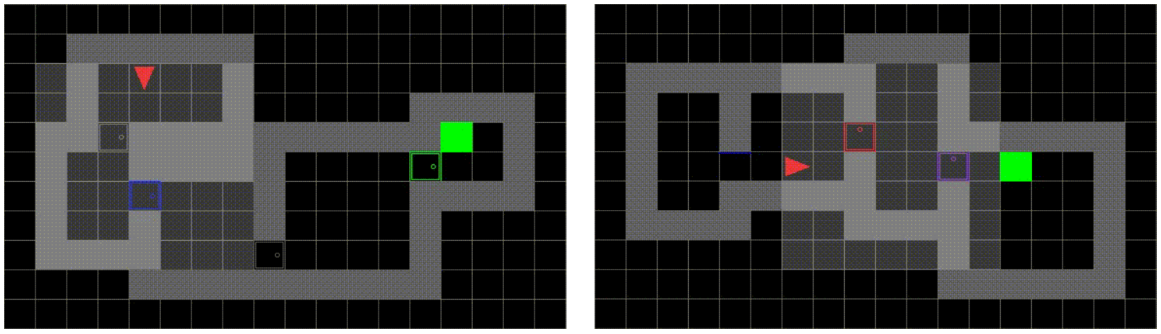
(b) Abysses and Bullet Bills in Super Mario Bros.

Figure 17: Examples of states appearing in environments commonly employed by the reinforcement learning community that produce the death of an interacting agent.

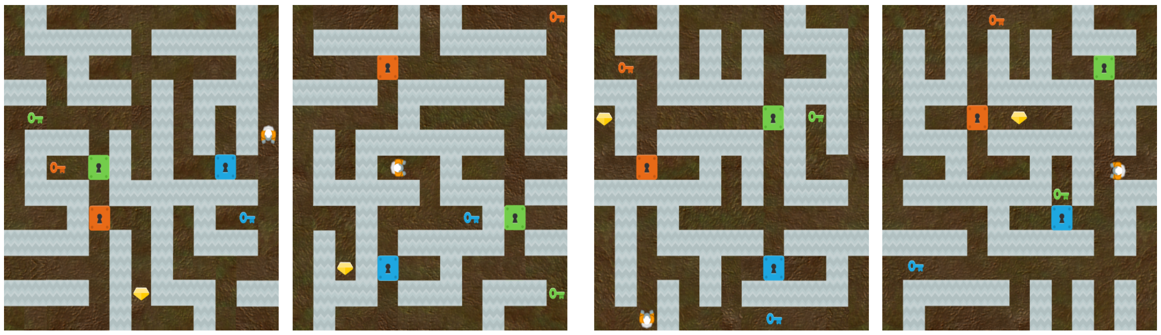
well suited to dealing with them. For instance, learners seeking new experiences across training episodes, such as ICM (Pathak et al., 2017) and RND (Burda et al., 2019b), might not be motivated to do deep exploration as every initial observation or transition is already brand new by default. Several algorithms that develop an automatic curriculum are also at odds with procedurally generated environments. Particularly, those that tell the learning agent which goal to reach at the beginning of each episode and that select such a goal from a buffer of previously seen states (e.g., Veeriah et al., 2018; Eysenbach et al., 2019a). Clearly, if all episodes are different from each other, the technique of using past states directly as goals will mostly produce unattainable ones.

Despite these adversities, a handful of works have already experimented with such sparse-reward procedurally generated environments, and they have even proposed some promising solution methods. For example, EC (Savinov et al., 2019) sees a continuous increase of successful episodes over time when trained in a partially observable 3D visual navigation task, in which the map layout and the textures of walls and floors change between episodes. Notably, such an increase rises significantly higher and is more consistent across runs than what is observed for other algorithms, namely PPO (Schulman et al., 2017) and ICM (Pathak et al., 2017). As described in Section 4.5, EC uses a learned action distance classifier (that takes two states or observations as input) together with a memory buffer to give itself bonuses whenever it reaches observations that are far from most others seen earlier during the current episode. Arguably, as long as EC is able to generalize such a distance classifier, it should do well in any given procedurally generated environment. Luckily, in many domains this is actually feasible and not necessarily hard. Like in the one assessed by the authors, where observations belonging to rooms with alike and different textures could be taken to be close to and far from each other, respectively. A similar solution concept was employed by Puigdomenech Badia et al. (2020b) to develop NGU, which thoroughly explores fully observable 2D mazes of varying geometries. NGU multiplies two exploration bonuses. One rewards novelty across episodes, and it is implemented simply by invoking RND without any extra modifications. Meanwhile, the second bonus rewards episodic novelty. Its computation is grounded on a learned controllable representation (similar to ICM). And, to put it in very simple terms, the bonus assigned to each observation is inversely related to the sum of the similarities measured between its representation and those of all previously perceived observations within the current episode. More succinctly, large bonuses correspond to observations that are less similar to every other one occurring earlier in an episode. A few extra details: similarities are computed according to a predetermined kernel function (non-learnable); their sum considers only the k-th nearest neighbors; and past observations are stored in an episodic memory buffer. RIDE (Raileanu & Rocktaeschel, 2020) shows a good performance in a variety of partially observable 2D visual navigation domains, whose geometries (rooms dimensions, locations of passages between rooms) and dynamic elements (positions of keys, doors, boxes, etc.) are readjusted across episodes. RIDE learns the same forward and inverse dynamics models as ICM, alongside a shared representation. However, its bonus, named impact-driven reward, is instead defined as the Euclidean distance between consecutive state representations, divided by the episodic visitation count associated with the next state. In large or continuous observation spaces, this count can be estimated thanks to a density model as done by Bellemare et al. (2016). Again, just like the two previous methods, RIDE is built around an episodic bonus that will induce a steady coverage-promoting intrinsic reward landscape once its latent representation stabilizes. A landscape that, in addition, will be perfectly tailored to each procedurally generated episode of an environment. Another promising algorithm and one that deviates substantially from the former ones is AGAC (Flet-Berliac et al., 2021), which has been successfully tested in the same benchmarks as RIDE. AGAC trains a policy and a value network, as various other methods do, but critically it also trains an adversary network. The objective ascribed to this adversary is to minimize the Kullback-Leibler divergence between its own and the policy's action distribution. Meanwhile, the policy is set to maximize task reward, as usual, and it also tries to maximize its discrepancy with respect to the adversary (i.e., difference in action log-probabilities). According to the authors, this formulation is helpful in sparse-reward settings because it fosters the diversity of trajectories as the adversary constantly pushes the policy to behave differently from past executions.

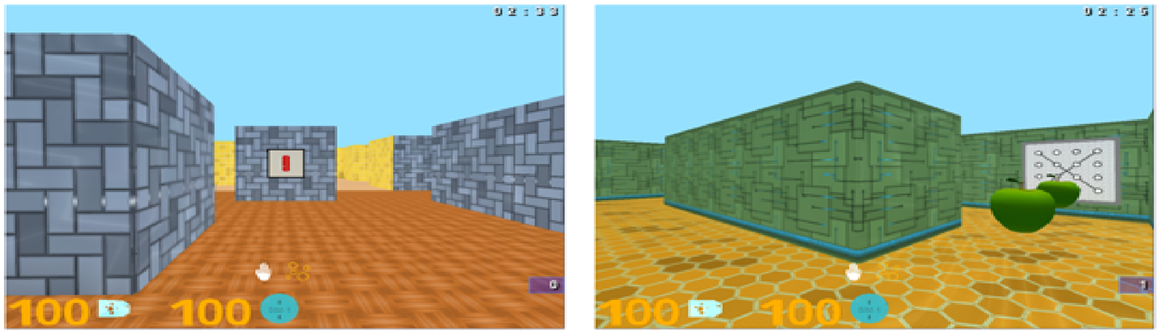
Currently, there is a meaningful amount of stimulating benchmark environments that integrate sparse rewards and procedural generation. At this point, it is important to make one clarification. To categorize a procedurally generated environment as having sparse rewards, it is not enough that it awards few incentives in every episode. It is also imperative that in any episode generated by such a domain the chances of a simple policy (e.g., one that selects actions at random) finding a reward be extremely low. Otherwise, if the distribution of episodes ranges smoothly from those where the learning agent obtains a reward easily to those where this is a herculean feat, then this may create a learning curriculum that can be exploited even by algorithms lacking advanced exploration mechanisms (as noted by Pathak et al., 2017; Savinov et al., 2019). That said, the following is a list of sparse-reward procedurally generated benchmarks used in previous studies. MiniGrid (Chevalier-Boisvert et al., 2018)



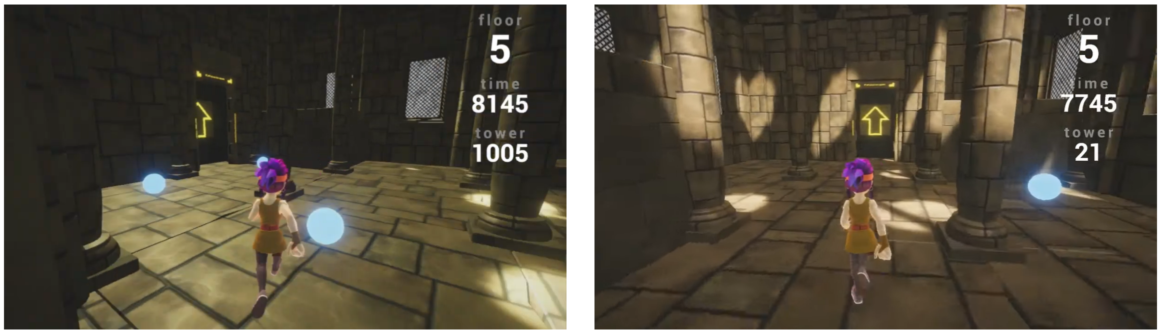
(a) MultiRoomNXSY (MiniGrid).



(b) Heist (Procgen).



(c) Initial observations produced by the random_maze domain (DeepMind Lab).



(d) Entrance to the 6th floor (Obstacle Tower).

Figure 18: Side-by-side comparison of procedurally generated episodes from various domains.

provides various 2D gridworld navigation tasks, where the interacting agent receives partial egocentric views of the environment (readily convertible into RGB images). The tasks more frequently employed by researchers are: MultiRoomNXSY, KeyCorridorSXRX and ObstructedMaze. MultiRoomNXSY rewards a learning agent upon arriving at the other end of a chain of rectangular rooms connected to each other by means of single-cell unlocked doors. The main attributes that change from one episode to the next include the dimensions and the relative position of each room as well as the placement and color of each door (see Figure 18(a)). KeyCorridorSXRX and ObstructedMaze likewise require the learning agent to traverse multiple rooms until finding a singular goal location (always found in a distant room), which also corresponds to the only reward granted by these domains. Additionally, said scenarios demand slightly more complex interactions, such as: grabbing a key from a box to open a door or pushing away a ball that is obstructing a door. In both cases, the map geometry does not vary between episodes, unlike the positions of keys, doors, boxes, balls and goal. Procgen (Cobbe et al., 2020) is a benchmark suite containing 16 environments, all implemented as 2D worlds with videogame-like visuals. Given that Procgen has been conceived primarily to assess generalization, only a couple of environments have sparse rewards. A particularly interesting one is Heist, in which the learning agent must travel through a network of locks and keys encased within a maze until encountering a gem. Procedural generation controls the shape of the maze plus the position of all items, as shown in Figure 18(b). The NetHack learning environment (Kuettler et al., 2020) is a testing platform framed around the terminal-based dungeon-crawler game of the same name. In this game, a hero is tasked with retrieving the Amulet of Yendor. To do so, it must descend over 50 procedurally generated levels of a dungeon and escape from it afterward. The game's dynamics is profoundly complex as it is partially observable and comprises 93 actions along with hundreds of unique elements to interact with. Many aspects of the game are readjusted across episodes, including the dungeon's layout and the exact location of places and items of interest. DeepMind Lab (Beattie et al., 2016) offers several visually rich 3D navigation tasks to reinforcement learning researchers. One readily available domain presents the learning agent with a new 3D maze to solve in each episode. The agent perceives the world through a first-person view camera (which can be occluded by walls) and receives a reward solely when it reaches a goal location. Mazes created in separate episodes differ in their geometry, the agent's initial position, the goal location and the textures applied to various sectors of the world (see Figure 18(c)). Likewise, Obstacle Tower (Juliani et al., 2019) is a 3D environment with high visual fidelity, whose gameplay mimics multi-level adventure videogames. It consists of up to 100 floors that the learning agent must journey through. Moreover, each floor is composed of multiple rooms and each room can contain some taxing subtask, such as: solving a puzzle, defeating enemies, evading obstacles or finding a key. As information, the agent is given images taken by a third-person camera that follows it everywhere plus an auxiliary vector encompassing abstract variables (e.g., keys in its possession, remaining time). Researchers are permitted to choose the reward structure of the environment; either sparse (+1 per completed floor) or dense (additional +0.1 per solved subtask). Finally, as seen in Figure 18(d), each floor of Obstacle Tower contains procedurally generated elements, namely: lighting, textures, room layout and floor plan.

5. Conclusions

In this study, we have consolidated a thoughtful review of numerous environmental features that have a heavy impact on the effectiveness of deep reinforcement learning in sparse-reward environments. In total, we have covered nine features; two, resettability and reversibility, that have been exploited by previous works and other seven that have been reported as challenging for one or more families of algorithms with advanced exploration schemes. Pertaining to each feature, we have delivered clear definitions and useful example environments. In addition, we have summarized experimental results and discussions found in previous studies that shine a light on which algorithms and which components are affected by a certain feature and what mechanisms underlie this interplay. We anticipate that the benefits of this compilation effort will be twofold. First, the bulk of information contained here could assist researchers interested in analyzing the generality of their novel proposals. They could do this by validating from a theoretical perspective if their formulations share the same specificities or weaknesses as previous ones or if similar mechanisms take place in them. Taking this a step further, we would also like to see such a generality assessment carried out empirically in a set of benchmark environments, where each one of them is intentionally crafted to highlight exactly one of the features herein reported. And second, through its presentation of demanding environmental features, this work points researchers toward various open problems involving sparse rewards. Some impact only a reduced group of algorithms (e.g., action-prediction degeneracy), while others reveal more ubiquitous shortcomings that enclose approaches from diverse lines of research. And within this latter category, we regard the phenomena of retracing steps, distractors and procedural generation as the most fundamental and stimulating challenges of the lot, whose resolution would lead to the next generation of exploration strategies.

References

Atari (2021). Atari VCS/2600 scoreboard. http://www.ataricompendium.com/game_library/high_scores/high_scores.html. Accessed: 2021-12-30.

Bacon, P.-L., Harb, J., & Precup, D. (2017). The option-critic architecture. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1).

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2020). Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., & Petersen, S. (2016). Deepmind lab. *CoRR*, *abs/1612.03801*.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. In Precup, D., & Teh, Y. W. (Eds.), *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, p. 1479–1487, Red Hook, NY, USA. Curran Associates Inc.

Burda, Y., Edwards, H., Pathak, D., Storkey, A. J., Darrell, T., & Efros, A. A. (2019a). Large-scale study of curiosity-driven learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Burda, Y., Edwards, H., Storkey, A. J., & Klimov, O. (2019b). Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Catacora Ocana, J. M., Capobianco, R., & Nardi, D. (2022). Exploration-intensive distractors: Two environment proposals and a benchmarking. In Bandini, S., Gasparini, F., Mascardi, V., Palmonari, M., & Vizzari, G. (Eds.), *AIxIA - 2021 Advances in Artificial Intelligence - XXth International Conference of the Italian Association for Artificial Intelligence, Virtual Event, December 1-3*.

Chevalier-Boisvert, M., Willems, L., & Pal, S. (2018). Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid.

Choi, J., Guo, Y., Moczulski, M., Oh, J., Wu, N., Norouzi, M., & Lee, H. (2019). Contingency-aware exploration in reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Cobbe, K., Hesse, C., Hilton, J., & Schulman, J. (2020). Leveraging procedural generation to benchmark reinforcement learning. In III, H. D., & Singh, A. (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR.

Dann, M., Zambetta, F., & Thangarajah, J. (2019). Deriving subgoals autonomously to accelerate learning in sparse reward domains. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 881–889.

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

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018). IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Dy, J., & Krause, A. (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR.

Eysenbach, B., Salakhutdinov, R. R., & Levine, S. (2019a). Search on the replay buffer: Bridging planning and reinforcement learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 32*, pp. 15220–15231. Curran Associates, Inc.

Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2019b). Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*.

Flet-Berliac, Y., Ferret, J., Pietquin, O., Preux, P., & Geist, M. (2021). Adversarially guided actor-critic. In *International Conference on Learning Representations*.

Florensa, C., Degrave, J., Heess, N., Springenberg, J. T., & Riedmiller, M. A. (2019). Self-supervised learning of image embedding for continuous control. *CoRR*, *abs/1901.00943*.

Florensa, C., Held, D., Geng, X., & Abbeel, P. (2018). Automatic goal generation for reinforcement learning agents. In Dy, J., & Krause, A. (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 1515–1528. PMLR.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., & Abbeel, P. (2017). Reverse curriculum generation for reinforcement learning. In Levine, S., Vanhoucke, V., & Goldberg, K. (Eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, Vol. 78 of *Proceedings of Machine Learning Research*, pp. 482–495. PMLR.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, *3*(4), 128–135.

Ghosh, D., Gupta, A., & Levine, S. (2019). Learning actionable representations with goal conditioned policies. In *International Conference on Learning Representations*.

Gregor, K., Rezende, D. J., & Wierstra, D. (2016). Variational intrinsic control. *CoRR*, *abs/1611.07507*.

Grinsztajn, N., Ferret, J., Pietquin, O., Preux, P., & Geist, M. (2021). There is no turning back: A self-supervised approach for reversibility-aware reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Gu, S., Holly, E., Lillicrap, T. P., & Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pp. 3389–3396. IEEE.

Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., & Levine, S. (2019). Learning to walk via deep reinforcement learning.. In *Robotics: Science and Systems*.

Haber, N., Mrowca, D., Wang, S., Fei-Fei, L., & Yamins, D. L. K. (2018). Learning to play with intrinsically-motivated, self-aware agents. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 8398–8409, Red Hook, NY, USA. Curran Associates Inc.

Hasselt, H. v., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, p. 2094–2100. AAAI Press.

Heinrich, J., & Silver, D. (2016). Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, *abs/1603.01121*.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning.. In McIlraith, S. A., & Weinberger, K. Q. (Eds.), *AAAI*, pp. 3215–3222. AAAI Press.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., & Silver, D. (2018). Distributed prioritized experience replay. In *International Conference on Learning Representations*.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2016). Vime: Variational information maximizing exploration. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, p. 1117–1125, Red Hook, NY, USA. Curran Associates Inc.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.

Juliani, A., Khalifa, A., Berges, V.-P., Harper, J., Teng, E., Henry, H., Crespi, A., Togelius, J., & Lange, D. (2019). Obstacle tower: A generalization challenge in vision, control, and planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, p. 2684–2691. AAAI Press.

Kim, Y., Nam, W., Kim, H., Kim, J.-H., & Kim, G. (2019). Curiosity-bottleneck: Exploration by distilling task-specific novelty. In Chaudhuri, K., & Salakhutdinov, R. (Eds.), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 3379–3388. PMLR.

Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. In Solla, S. A., Leen, T. K., & Müller, K. (Eds.), *Advances in Neural Information Processing Systems 12*, pp. 1008–1014. MIT Press.

Kuettler, H., Nardelli, N., Miller, A. H., Raileanu, R., Selvatici, M., Grefenstette, E., & Rocktaeschel, T. (2020). The NetHack Learning Environment. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., & Tenenbaum, J. B. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, p. 3682–3690, Red Hook, NY, USA. Curran Associates Inc.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E. P., Levine, S., & Salakhutdinov, R. (2019). Efficient exploration via state marginal matching. *CoRR*, *abs/1906.05274*.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2015). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, *17*, 39:1–39:40.

Levy, A., Platt, R., & Saenko, K. (2019). Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*.

Machado, M. C., Bellemare, M. G., & Bowling, M. (2017). A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 2295–2304. JMLR.org.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., & Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *J. Artif. Int. Res.*, *61*(1), 523–562.

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., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

Mohamed, S., & Rezende, D. J. (2015). Variational information maximisation for intrinsically motivated reinforcement learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, p. 2125–2133, Cambridge, MA, USA. MIT Press.

Nachum, O., Gu, S., Lee, H., & Levine, S. (2018). Data-efficient hierarchical reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 3307–3317, Red Hook, NY, USA. Curran Associates Inc.

O'Donoghue, B., Osband, I., Munos, R., & Mnih, V. (2018). The uncertainty bellman equation and exploration. In Dy, J. G., & Krause, A. (Eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 3836–3845. PMLR.

Oh, J., Guo, Y., Singh, S., & Lee, H. (2018). Self-imitation learning. In Dy, J., & Krause, A. (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80 of *Proceedings of Machine Learning Research*, pp. 3878–3887. PMLR.

Osband, I., Blundell, C., Pritzel, A., & Roy, B. V. (2016). Deep exploration via bootstrapped dqn. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, p. 4033–4041, Red Hook, NY, USA. Curran Associates Inc.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 488–489.

Puigdomenech Badia, A., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., & Blundell, C. (2020a). Agent57: Outperforming the atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 507–517. PMLR.

Puigdomenech Badia, A., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., & Blundell, C. (2020b). Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*.

Racaniere, S., Weber, T., Reichert, D. P., Buesing, L., Guez, A., Rezende, D., Puigdomènech Badia, A., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., & Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 5694–5705, Red Hook, NY, USA. Curran Associates Inc.

Raileanu, R., & Rocktaeschel, T. (2020). Ride: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., & Levine, S. (2018). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania.

Roderick, M., Grimm, C., & Tellex, S. (2018). Deep abstract q-networks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, p. 131–138, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Savinov, N., Raichuk, A., Vincent, D., Marinier, R., Pollefeys, M., Lillicrap, T., & Gelly, S. (2019). Episodic curiosity through reachability. In *International Conference on Learning Representations*.

Schmidhuber, J. (1991). Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pp. 1458–1463 vol.2.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In Bach, F., & Blei, D. (Eds.), *Proceedings of the 32nd International Conference on Machine Learning*, Vol. 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, *abs/1707.06347*.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., & Pathak, D. (2020). Planning to explore via self-supervised world models. In *ICML*.

Shelhamer, E., Mahmoudieh, P., Argus, M., & Darrell, T. (2016). Loss is its own reward: Self-supervision for reinforcement learning. *CoRR*, *abs/1612.07307*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L. R., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, *550*, 354–359.

Song, Y., Wang, J., Lukasiewicz, T., Xu, Z., Zhang, S., Wojcicki, A., & Xu, M. (2020). Mega-reward: Achieving human-level play without extrinsic rewards. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5826–5833.

Stadie, B. C., Levine, S., & Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, *abs/1507.00814*.

Stanton, C., & Clune, J. (2018). Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems. *CoRR*, *abs/1806.00553*.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., & Fergus, R. (2018). Intrinsic motivation and automatic curricula via asymmetric self-play. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Mach. Learn.*, *3*(1), 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to Reinforcement Learning* (1st edition). MIT Press, Cambridge, MA, USA.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999a). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, p. 1057–1063, Cambridge, MA, USA. MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999b). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, *112*(1–2), 181–211.

Taiga, A. A., Fedus, W., Machado, M. C., Courville, A., & Bellemare, M. G. (2020). On bonus based exploration methods in the arcade learning environment. In *International Conference on Learning Representations*.

Veeriah, V., Oh, J., & Singh, S. (2018). Many-goals reinforcement learning. *CoRR*, *abs/1806.09605*.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., & Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 3540–3549. JMLR.org.

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., Gulcehre, C., 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., & Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, *8*(3–4), 229–256.