RELEVANT ACTIONS MATTER: MOTIVATING AGENTS WITH ACTION USEFULNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse rewards are double-edged training signals in reinforcement learning: easy to design but hard to optimize. Intrinsic motivation guidances have thus been developed toward alleviating the resulting exploration problem. They usually incentivize agents to look for new states through novelty signals. Yet, such methods encourage exhaustive exploration of the state space rather than focusing on the environment's salient interaction opportunities. We propose a new exploration method, called Relevant Actions Matter (RAM), shifting the emphasis from state novelty to state with relevant actions. While most actions consistently change the state when used, *e.g.* moving the agent, some actions are only effective in specific states, *e.g.*, *opening* a door, *grabbing* an object. RAM detects and rewards actions that seldom affect the environment. We evaluate RAM on the procedurally-generated environment MiniGrid, against state-of-the-art methods. Experiments consistently show that RAM greatly reduces sample complexity, installing the new state-of-the-art in MiniGrid. [1]

1 INTRODUCTION

We consider the reinforcement learning (RL) problem in which an agent learns to interact with its environment optimally w.r.t. a cumulative function of reward signals collected along its trajectories (Sutton and Barto, 2018). To do so, an RL agent explores its surrounding, aiming at retrieving the most prominent course of actions, and updates its behavior accordingly. When the environment provides abundant rewards, the agent may successfully collect enough training signals by performing random actions. But as soon as the environment provides scarce rewards, the agent is reduced to inefficiently waver around without being able to update its policy. To palliate this lack of training signals, one common method consists in intrinsically motivating the agent to explore its environment using a self-rewarding mechanism (Schmidhuber, 1991; Oudeyer *et al.*, 2007).

In the online RL literature, a widespread strategy is to augment the sparse *extrinsic* reward from the environment with a generated dense *intrinsic* reward that steers exploration (Chentanez *et al.*, 2005). Hence, the intrinsic reward should encode a degree of "novelty," "surprise," or "curiosity" which is often encoded as an estimate of the agent's visitation frequency of state-action pairs. The agent is incentivized to diversely interact with its environment to collect intrinsic rewards, which may ultimately trigger extrinsic rewards. Nonetheless, establishing intrinsic motivation signals remains double-edged as it introduces human-priors, may lead to sub-optimal policies or foster reward hacking behavior.

All in all, different novelty measures have been studied, where each of them entails different exploration behaviors. For instance, count-based methods keep counts of previous observations to bait the agent to explore unseen states (Lopes *et al.*, 2012; Bellemare *et al.*, 2016; Ecoffet *et al.*, 2019). Yet, these approaches implicitly encourage an exhaustive search of the state space. Differently, curiosity-based methods train a model that encapsulates the environment dynamics, before nudging the agent to visit state-transitions with high prediction errors (Pathak *et al.*, 2017; Burda *et al.*, 2018; Haber *et al.*, 2018; Houthooft *et al.*, 2016) or large change in the value of state features (Raileanu and Rocktäschel, 2019). However, the first category suffers from reward decay across episodes and poor generalization within procedurally-generated environments. We here observe that the second category insufficiently favors exploration towards novel and useful actions.

[1] Source code will be released upon acceptance.

In this paper, we therefore aim to shift the emphasis from state novelty distributions towards novel action distributions to develop new intrinsic motivation signals, and consequently, change the exploration behavior. More precisely, we aim at encouraging the agent to visit states that allow rare and relevant actions, i.e. actions that can only be performed in rare occasions. Imagine that an infant discovers that pushing a button triggers a light; s/he is likely to push everywhere to switch on new lights. By repeating his/her action, the infant may eventually uncover new buttons, and start associating the action *push* to the relevant state features of *buttons*. A similar observation can be made within virtual environments and embodied agents. We expect the agent to first detect rare actions to learn while being nudged towards the states that allow performing such actions.

In this spirit, we propose a new approach we name Relevant Actions Matter (RAM). Instead of uniformly seeking for novel states, RAM encourages exploring states allowing actions that are rarely useful; those rarely relevant actions are generally hard to retrieve by random exploration. In other words, the agent is intrinsically rewarded when successfully performing an action that is usually ineffectual. We observe that this simple mechanism induces a remarkably different exploration behavior differing from the common state-count and curiosity-based patterns.

Formally, RAM keeps track of two quantities for each action: the number of times the action has been used and the number of times the action led to a state change. The resulting intrinsic reward is inversely proportional to the number of times the action has led to a state change. Noticeably, RAM primarily keeps count of actions, and can thus be defined as an action count-based method. Besides, tracking actions (as opposed to states) naturally scales in RL: in the discrete case, there is generally less than a few thousand actions, allowing for an exact count. In the continuous case, actions may easily be discretized without using complex density models (Tang and Agrawal, 2020).

This paper first provides an overview of recent exploration methods before introducing RAM as an action-driven intrinsic motivation method. We then study this approach in the MiniGrid procedurally generated environment (Chevalier-Boisvert *et al.*, 2018). Despite their apparent simplicity, these tasks contain intermediate decisive actions, e.g. picking keys, which have kept in check advanced exploration methods (Raileanu and Rocktäschel, 2019; Campero *et al.*, 2020). We empirically show that RAM reduces the sample complexity by a factor of 2 to 10 in a diverse set of environments while resolving the hardest tasks. We then study how RAM amends the agent's behavior and compare it to other methods. Finally, we also analyze whether RAM may lead to unwanted agent behavior when facing environment with multiple interactions, which we refer as the *BallPit-problem*.

2 Related Works

RL algorithms require the agents to acquire knowledge about their environment to update their policy; exploration has thus been one of the longest running problems of RL (Sutton and Barto, 2018). Exploration methods have quickly been categorized into two broad categories: *directed* and *undirected* exploration (Thrun, 1992).

On the one hand, undirected exploration does not use any domain knowledge and ensures exploration by introducing stochasticity in the agent's policy. This approach includes methods such as random walk, ϵ-greedy, or Boltzmann exploration. Although they enable learning the optimal policy in the tabular setting, they require a number of steps that grows exponentially with the state space (Whitehead, 1991). Despite this inherent lack of sample efficiency, they remain valuable task-agnostic exploration strategies in large-scale problems with dense rewards. On the other hand, directed methods incorporate external priors to orient the exploration strategy through diverse heuristics or measures. Among others, uncertainty has been used to guide exploration towards ill-estimated state-action pairs by relying on the Bellman equation (Geist and Pietquin, 2010; O'Donoghue *et al.*, 2018) or by bootstrapping multiple Q-functions (Osband *et al.*, 2016). Despite being theoretically sound, these methods face scaling difficulties. In this paper, we study another directed exploration approach based on reward bonuses to densify the reward signal.

In this setting, the environment reward, namely *extrinsic* rewards, is augmented with an exploration guidance reward signal, namely *intrinsic* rewards (Chentanez *et al.*, 2005). This intrinsic reward spurs exploration by tipping the agent to take a specific course of actions. Furthermore, it makes undirected exploration mechanism applicable again by spreading milestone rewards during training. Inspired by cognitive science, this intrinsic reward often encodes a degree of "novelty," "surprise,"

,"curiosity" (Oudeyer *et al.*, 2007) , "learning progress" (Lopes *et al.*, 2012) or "boredom" (Schmidhuber, 1991). These common intrinsic motivation mechanisms are broadly categorized into three families: count-based, curiosity-based, and goal-based methods.

Count-based exploration aims to catalog visited states (or action-states pairs) along episodes to detect unseen states, and drive the agent towards them. It has first been proposed as an exploration heuristic in the early days of RL (Thrun, 1992) before being framed as an intrinsic exploration reward mechanisms in the tabular case (Strehl and Littman, 2008). Pseudo-counts were then introduced to approximate the state counts (Lopes *et al.*, 2012), where pseudo-counts were estimated through different density models to produce intrinsic rewards. Density models range from raw image downscaling with or without handcrafted state features (Ecoffet *et al.*, 2019), contextual trees (Bellemare *et al.*, 2016), generative neural models, e.g. PixelCNN (Ostrovski *et al.*, 2017), or autoencoders combined with a local hashing function (Tang *et al.*, 2017). Differently, Burda *et al.* (2018) use the prediction error between a randomly initialized network and a trained network as a state-count proxy. Yet, count-based methods may explore the immediate surrounding and heavily depend on the state representation quality. By shifting the emphasis on counting action, we thus address these representation constraints and push for distant interactions.

Curiosity-based exploration aims to encourage the agent to uncover the environment dynamics rather than cataloging states. Inspired by cognitive science, such agents learn a world model predicting the consequences of their actions; and they take an interest in challenging and refining it (Haber *et al.*, 2018; Oudeyer *et al.*, 2007). In RL, this intuition is transposed by taking the current state and action to predict the next state representation; the resulting prediction error is then turned into the intrinsic reward signal. Approaches mostly differ in learning the state representation: Stadie *et al.* (2015) compress raw observation with autoencoders, Burda *et al.* (2019) use random projections, Houthooft *et al.* (2016) capture the environment stochasticity by maximizing mutual information with Bayesian Networks. In parallel, Pathak *et al.* (2017) argue that the state representation should mainly encode features altered by the agent. They thus introduce an inverse model that predicts the action given two consequent states as a training signal. Yet, those intrinsic rewards based on prediction errors may attract the agent into irrelevant yet unpredictable transitions. Another drawback is reward evanescence: the intrinsic reward slowly vanishes as the model is getting better. Schmidhuber (1991) originally proposed to measure the mean error evolution rather than immediate errors to account for the agent progress. Differently, Raileanu and Rocktäschel (2019) replace the error prediction by the difference between consecutive representation states, removing the need to compute a vanishing prediction error. In this paper, we also compare successive states in a similar spirit, but we use it to catalog actions and bias state visitation through a different exploration scheme.

Goal-based methods provide identifiable and intermediate goals to reward the agent upon completion. Such approaches perform an explicit curriculum by slowly increasing the exploration depth through goal difficulties. They often build on top of the UVFA framework to condition the agent policy (Schaul *et al.*, 2015). Goal-based methods may take several forms ranging from hindsight experience replay (Andrychowicz *et al.*, 2017), adversarial goal-generation (Forestier *et al.*, 2017; Campero *et al.*, 2020) and hand-crafted goals.Yet, they may face to unstable training, complex goal definition, or require fully observable environment (Campero *et al.*, 2020). Other forms of intrinsic reward have been explored with empowerment (Mohamed and Jimenez Rezende, 2015) or trajectory diversities (Savinov *et al.*, 2018), but they are facing scalability issues. Hussenot *et al.* (2020) also tried to retrieve intrinsic motivation signals from human trajectories through inverse reinforcement learning. Finally, intrinsic motivation have been explored in hierarchical reinforcement learning (Barto *et al.*, 2004; Kulkarni *et al.*, 2016), but it goes beyond the scope of this paper.

3 Reinforcement Learning Background

Notation The environment is modeled as a Markov Decision Problem (MDP), where the MDP is defined as a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$. At each time step t, the agent is in a state $s_t \in \mathcal{S}$, where it selects an action $a_t \in \mathcal{A}$ according to its policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. It then receives a reward r_t from the environment's reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ and moves to the next state s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$ according to the transition kernel $\mathcal{P}$. Hence, the agent generates a trajectory $\tau = [s_0, a_0, r_0, s_1, r_1, a_1, \ldots, s_T, a_T, r_T]$ of length T. In practice, the policy is often parameterized by a weight vector $\boldsymbol{\theta} \in \Theta$. The goal is then to search for the optimal policy $\pi_{\boldsymbol{\theta}^*}$

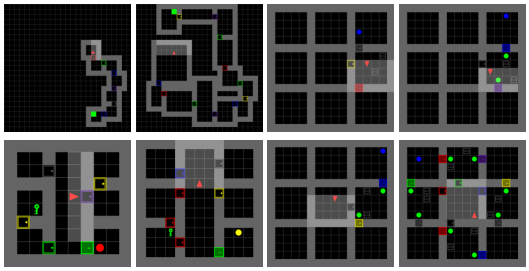

Figure 1: Top left to down right : MultiRoom (N7S4, N12S10), KeyCorridor (S4R3, S5R3), ObstructedMaze (2Dlh, 2Dlhb, 1Q, Full)

that maximizes the expected return $J(\boldsymbol{\theta}) = E^{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0} \gamma^t r(s_t, a_t, s_{t+1})\right]$ by directly optimizing the policy parameters $\boldsymbol{\theta}$.

Intrinsic Motivation In this setting, the reward function is decomposed into an extrinsic reward returned by the environment $r^e(s_t, a_t)$ and a new intrinsic reward $r^i(s_t, a_t, s_{t+1})$. Therefore, the new reward function is defined as $r(s_t, a_t, s_{t+1}) = r^e(s_t, a_t, s_{t+1}) + \beta r^i(s_t, a_t, s_{t+1})$ where β is an hyperparameter to balance the two return signals. In practice, the extrinsic reward is often a sparse task-specific signal while the intrinsic reward is usually a dense training signal that fosters exploration.

4 Relevant Actions Matter

Intuition While most actions consistently move the agent to a new state, some actions do not affect specific states, i.e., the agent remains in the same state after performing it. We hence define an *effective action* if the new state of the environment is different from what it would have been if no action were to be taken. For instance, in tasks involving embodied interaction, such state-action pairs include moving forward while facing a wall or grabbing non-existent objects. Although one may update the MDP only to keep effective actions, such an operation may not always be feasible or desirable in practice. It is thus up to the agent to learn the correct actionable states through exploration. Noticeably, those rare state-actions are often landmarks in the environment dynamics, e.g., triggering buttons or opening doors. One idea is thus to bias the agent to visit states that effectively allow rare actions. RAM encapsulates this exploration pattern by (1) detecting rare but effective actions, (2) rewarding the agent when effectively performing these rare actions. In short, rare and effective actions are the relevant actions that matter.

Method For every action a_i, the agent tracks two quantities. The number U of times an action has been **u**sed during past trajectories, and the number E of times the action was **e**ffective, i.e. change the state $s_t \neq s_{t+1}$[2]. Formally, given the whole history of transitions across episodes $\mathcal{H} = (s_h, a_h, s_{h+1})_{h=0}^{H}$:

$$U^{\mathcal{H}}(a) = \sum_{h=0}^{H} \mathbf{1}_{\{a_h = a\}}, \tag{1}$$

$$E^{\mathcal{H}}(a) = \sum_{h=0}^{H} (\mathbf{1}_{\{a_h = a\}} \times \mathbf{1}_{\{s_h \neq s_{h+1}\}}), \tag{2}$$

where $\mathbf{1}$ is the indicator function and $\times$ the product operator.

Intuitively, the ratio $E^{\mathcal{H}}(a)/U^{\mathcal{H}}(a)$ indicates how often the action a has been effective along the history $\mathcal{H}$. For instance, actions that move an agent would update the state most of the time, therefore $U(a_i) \approx E(a_i)$. On the other hand, grabbing objects only changes the state in rare occurrences, and $U(a_i) \geq E(a_i)$. We then define the reward as:

[2]In noisy or dynamic environment, it is possible to relax or learn this as mentioned in subsection 6.3

$$R(s_t, a_t, s_{t+1}) = \frac{\eta^{1-\frac{E^{\mathcal{H}}(a_t)}{U^{\mathcal{H}}(a_t)}} - 1}{\eta - 1}, \tag{3}$$

where η is a hyperparameter. This function is a continuous approximation of an exponential decay $\exp^{-\eta E^{\mathcal{H}}(a_t)/U^{\mathcal{H}}(a_t)}$. It ranges from 1 when $E^{\mathcal{H}} = 0$ and goes to 0 when $E^{\mathcal{H}} = U^{\mathcal{H}}$. Small η leads to a uniform bonus on all actions whereas, large values favor rare and efficient actions (see subsection A.2 in appendix for an illustration).

Finally, an intrinsic reward mechanism often requires to discount the intrinsic bonus within an episode. Hence, it prevents the agent from overexploiting, and being stuck in local exploration minima. Inspired by theoretically sound count-based methods (Strehl and Littman, 2008), we thus divide the previous ratio by an episodic state-count to define the final RAM intrinsic reward:

$$r^i_{RAM}(s_t, a_t, s_{t+1}) = \begin{cases} 0 & \text{if } s_t = s_{t+1} \\ \frac{R(s_t, a_t, s_{t+1})}{\sqrt{N^\tau(s_{t+1})}} & \text{otherwise,} \end{cases} \tag{4}$$

where $N^\tau(s) = \sum_{h=0}^{t} \mathbf{1}_{\{s=s_h\}}$ is an episodic state count which is reset at the beginning of each episode. In high-dimensional state space, the episodic state count can be replaced by a pseudo-count (Bellemare *et al.*, 2016) or an episodic novelty mechanism (Badia *et al.*, 2020).

Action-based Counter As counting methods may sound anachronistic, we emphasize again that actions are ascertainable in RL, i.e. they can be easily enumerated. As opposed to state-counting which requires complex density models (Ostrovski *et al.*, 2017), discrete action suffers less from the curse of dimensionality, and can easily be binned together in the case of a large action set (Dulac-Arnold *et al.*, 2015). Besides, although RAM relies on an episodic state count, a raw approximation is sufficient as it encodes a reward decay.

5 Experimental settings

We evaluate RAM in the procedurally-generated environments MiniGrid (Chevalier-Boisvert *et al.*, 2018). MiniGrid is a partially observable 2D gridworld with a diverse set of tasks. The RL agent needs to collect items and open locked doors before reaching a final destination. Despite its apparent simplicity, several MiniGrid environments require the agent to perform exploration with few specific interactions, and have kept in check state-of-the-art exploration procedures (Raileanu and Rocktäschel, 2019). For each experiment, we report the rolling mean (over 40k timesteps) and standard deviation over 5 seeds.

5.1 MiniGrid Environment

Each new MiniGrid world contains a combination of rooms that are populated with objects (balls, boxes or keys), and are linked together through (locked/unlocked) doors. Balls and keys can be picked up or dropped and the agent may only carry one of them at a time. Boxes can be opened to discover a hidden colored key. Doors can be unlocked with keys matching their color. The agent is rewarded after reaching the goal tile. At each step, the agent observes a 7x7 representation of its field of view and the item it carries if any. The agent may perform one out of seven actions: move forward, turn right, turn left, pick-up object, drop object, toggle. Noticeably, some actions are ineffective in specific states, e.g. moving forward in front of a wall, picking-up/dropping/toggling objects when none is available. Following (Raileanu and Rocktäschel, 2019; Campero *et al.*, 2020), we focus on three hard exploration tasks, which are illustrated in Figure 2.

MultiRoom(N-S): The agent must navigate through a sequence of empty rooms connected by doors of different colors. A map contains N rooms, whose indoor width and height are sampled within 2 and $S - 2$ tiles. MultiRoom maps entail limited interaction as the agent only has to toggle doors and no object manipulation is required. Yet, this bare-bone environment constitutes a good preliminary trial.

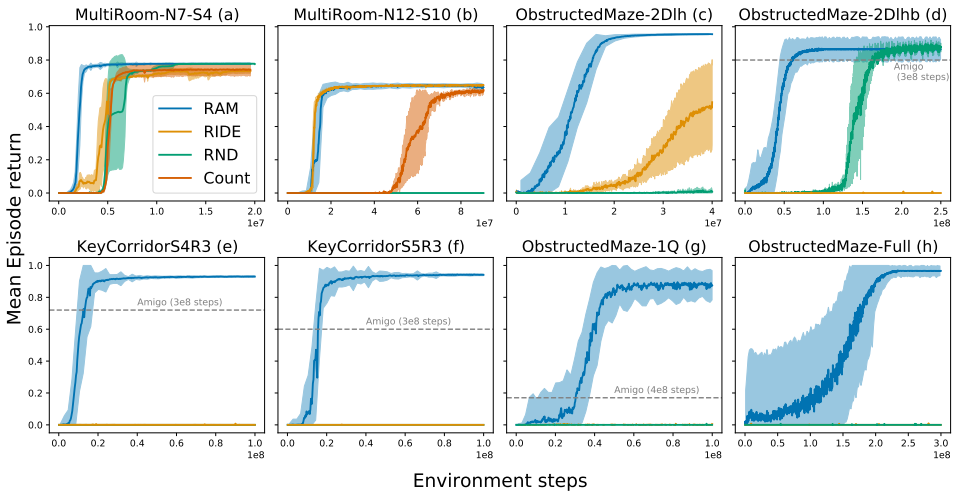

Figure 2: Comparison between intrinsically motivated methods on multiple MiniGrid tasks.

KeyCorridor(S-R): The agent must explore multiple adjacent unlocked rooms to retrieve a key, open the remaining locked room, and collect the green ball. A map contains a large main corridor connected to $2 \times R$ square rooms of fixed indoor dimension $S - 2$. Solving a KeyCorridor map requires the agent to perform a specific sequence of interactions, which makes the task more difficult than MultiRoom.

ObstructedMaze: The agent must explore a grid of rooms that are randomly connected to each others in order to collect a blue ball. Some of the doors are locked and the agent has to either directly collect the keys or toggle boxes to reveal them. Besides, distractor balls are added to block door access. ObstructedMaze can quickly become a hard maze with false leads and complex interactions.

5.2 Experimental Setting

Training We follow the training protocol defined in (Raileanu and Rocktäschel, 2019; Campero *et al.*, 2020). We use 3 convolution layers with a kernel size of 3, followed by 2 fully-connected layers of size 1024, and an LSTM of hidden size 1024. Finally, we use two separate fully-connected layers of size 1024 for the actor's and critic's head. We train our model with the distributed actor-critic algorithm IMPALA (Espeholt *et al.*, 2018) TorchBeast implementation (Küttler *et al.*, 2019). (All parameters in subsection A.3).

Baselines We here cover the three common families of intrinsically motivated reward mechanisms. COUNT (Strehl and Littman, 2008) is a counting method that baits the agent to explore less visited states. In this setting, we use a tabular-count to catalog the state-action pairs. RND (Burda *et al.*, 2018) acts as a states' pseudo-count method. A network is trained to predict randomly projected states and the normalized predicton error is used as intrinsic reward. RIDE (Raileanu and Rocktäschel, 2019) is a curiosity-based model that builds upon (Pathak *et al.*, 2017). It computes the difference between two consecutive states, encouraging the agent to perform actions that lead it to a maximally different states. AMIGO (Campero *et al.*, 2020) is a hierarchical goal-based method, splitting the agent into two components: an adversarial goal-setter and a goal-condition learner that adversarially creates goals.

6 Experimental Results

6.1 Base Environment

Figure 2 displays the results on 8 MiniGrid tasks. Noticeably, RAM outperforms all the baselines in sample complexity, and even solves among the most complex worlds. In MultiRoom, we observe

that RAM outperforms RIDE, RND, and COUNT in the simple setup (N7S4), and matches RIDE's sample complexity performance on the challenging setup (N12S10). Note RAM does not seem to be penalized by the small amount of possible interactions. In KeyCorridor and ObstructedMaze, RIDE, RND, and AMIGO learn in the easiest instances but they struggle as the difficulty, i.e. exploration depth, increases as first observed in (Campero *et al.*, 2020). On the other hand, RAM consistently solves all the environments, even the challenging ObstructedMaze-Full.

We derive two hypotheses from those results: (1) State-count rewards exhaustively explore the state space, reducing the overall exploration coverage (2) Curiosity-based rewards do not emphasize enough salient interactions and then explore new but irrelevant state-action pairs. Although such approaches were successful in many environments, those exploration behaviors may fail as soon as specific interactions must be regularly performed in the exploration process. In the following, we thus try to assess those hypotheses.

6.2 Intrinsic exploration behavior

We first conduct a series of experiments without external reward to study what *type of exploration* each bonus creates. In other words, what are the inductive exploration biases that arise from the different intrinsic reward mechanisms. To do so, we rely on two metrics: the state visit (plotted as heatmaps) and the action distribution (plotted as bar plots).

Rewardless Playground In this spirit, we design a sandbox environment without any specific goal to observe the agent behavior visually, akin to a kindergarten. This environment contains multiple keys, balls, and boxes located in the corners and spawns the agent facing a random direction. **??** shows the agent state visits for during 10^6 training timesteps when only using the intrinsic reward signal.

We observe that RND and RAM are both attracted by the objects and explore the space thoroughly, whereas RIDE and COUNT remain close to the center and seldomly reach the objects. This observation backs our results in ObstructedMaze2Dlh, where RND and RAM are the only methods exploring thoughtfully the environment. It also confirms our hypothesis that standard state-based approaches, e.g., COUNT, may not be pushed enough to perform in-depth exploration. Surprisingly, the curiosity-based method RIDE has not been strongly incentivized to interact with remote objects, suggesting that it may suffer from its dependency on the state representation. However, these experiments do not explain the performance difference between RND and RAM on the most challenging setups. Thus, looking at the action distribution is necessary.

Rewardless KeyCorridorS4R3 We then study the behavior that is solely intrinsically motivated in the KeyCorridor environment to better grasp the RAM performance in this setting. Similarly, we trained the agents on KeyCorridorS4R3 for 10^7 timesteps with only the intrinsic reward signal, and results are displayed in Figure 3.

All the baselines – RIDE, RND, and COUNT – remain mostly stuck in the central corridor, where RAM explores rooms more uniformly. More impressively, the RAM agent naturally picks the key, enters the locked room, and grabs the ball 7% of the times without any extrinsic reward. COUNT, RIDE, and RND all have a success ratio below 0.6%, which may explain why RAM manages to solve this task. Further details can be found in Figure 9 and Table 1.

We also observe a large discrepancy in the action distribution between the different methods. First, we observe that RND and RAM action distributions remain approximately uniform while RIDE and COUNT favor moving actions, reducing the opportunity for interactions. Second, and crucially, the impact distributions $E^{\mathcal{H}}(a)$ differs drastically between RAM and other methods. All agents are trying actions such as *pick, toggle* or *drop*, but those actions are rarely changing the agent's state. These actions are not used in the appropriate context, i.e., in front of an object. It means that rewarding state novelty might not be enough to discover effective actions, thus wasting samples. Although RAM and RND had similar state-visitation and action distribution patterns, only RAM correctly apprehend rarely effective actions, and correctly use them to explore its environment.

6.3 Intrinsic Motivation Pitfalls

The Ball Pit Problem As RAM biases the state visit distribution towards performing rare actions, it may introduce a poor exploration pattern when facing too many of such states. We refer to this

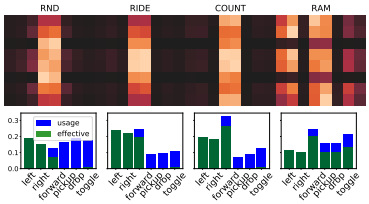

Figure 3: State and action distributions in *reward-less* KeyCorridor (S4R3). $U^{\mathcal{H}}(a)$ and $E^{\mathcal{H}}(a)$ action-count are in blue and green. Only RAM correctly uses pickup/drop/toggle during exploration.

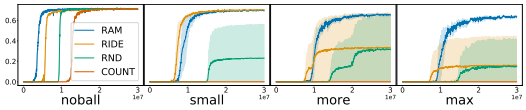

Figure 4: As distractors are added (from left to right), we observe a drop in performance for all methods.

potential issue as the *Ballpit problem*: the agent remains in rooms with plenty of balls to interact with. We created four versions of *Multiroom* (normal, small, more, max), and randomly spawned objects to assess the agent behavior (more details in subsection A.1).

As the number of objects grows, the performance of all algorithms deteriorates. RND, COUNT are mostly affected by this problem, as the number of states is growing exponentially; thus, counting state occurrence is challenging. RIDE is less affected by the BallPit problem, but most surprisingly, RAM is the only one to reach the exit consistently in the most challenging setup. The $E^{\mathcal{H}}(a)/U^{\mathcal{H}}(a)$ ratio correctly balances the exploration bonus, and does not take over the final extrinsic reward.

The Noisy-TV problem State-count based agent are attracted to state-action pairs with random noise. In its current definition, RAM is also affected while computing $E^{\mathcal{H}}$. Similar to (Burda *et al.*, 2018), this effect can be circumvented by using an inverse model, and we leave it for future work.

ColorMaze In Figure 6, we design a map with a sequence of open rooms, colored floor changing every episode, two boxes with one hidden key, and a locked door leading to the reward. All baselines remain in the first part of the maze while RAM quickly reaches the objects and solves the task. This experiment highlights again how shifting the emphasis from exhaustive state-visit to relevant state-visit can be beneficial, and change the exploration pattern.

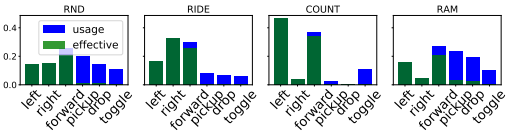

Figure 5: Playground Action Distribution for all baselines. The same pattern can be observed, baseline algorithms scarcely use *pickup, drop and toggle* and when they do, it does not impact the environment. RAM on the other hand discovers how to use pickup and drop.

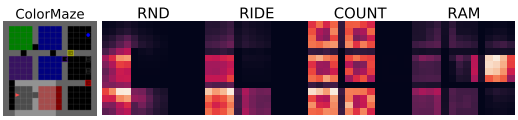

Figure 6: RND, RIDE and COUNT remain within the colored region whereas RAM learns to go straight to the boxes and keys.

7 Conclusion

We introduce Relevant Actions Matter (RAM), a new action-based intrinsic exploration algorithm. As opposed to count-based and curiosity-driven methods, RAM shifts the emphasis from novel state to state with relevant actions, rewarding actions that are rarely effective in the environment. Combined with a simple episodic count, RAM outperforms recent exploration methods on a variety of hard exploratory tasks in a Minigrid environment. This proof of concept illustrates that action-based exploration is a promising approach as it induces surprisingly different exploration patterns. We also pointed out a new category of problems called *BallPit*, which deteriorate performance of many intrinsically motivated reward approaches.

REFERENCES

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NIPS*, 2017.

A. P. Badia, B. Piot, S. Kapturowski, et al. Agent57: Outperforming the atari human benchmark. In *ICML*, 2020.

A. G. Barto, S. Singh, and N. Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *ICDL*, 2004.

M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 2016.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *ICLR*, 2018.

Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

A. Campero, R. Raileanu, H. Küttler, et al. Learning with amigo: Adversarially motivated intrinsic goals. In *ICLR*, 2020.

N. Chentanez, A. G. Barto, and S. P. Singh. Intrinsically motivated reinforcement learning. In *NIPS*, 2005.

M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, Saharia, et al. Babyai: A platform to study the sample efficiency of grounded language learning. In *ICLR*, 2018.

G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, Lillicrap, et al. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv:1901.10995*, 2019.

L. Espeholt, H. Soyer, R. Munos, K. Simonyan, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *ICML*, 2018.

S. Forestier, R. Portelas, Y. Mollard, and P.-Y. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv:1708.02190*, 2017.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of artificial intelligence research*, 39:483–532, 2010.

N. Haber, D. Mrowca, S. Wang, L. F. Fei-Fei, and D. L. Yamins. Learning to play with intrinsically-motivated, self-aware agents. In *NIPS*, 2018.

R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.

L. Hussenot, R. Dadashi, M. Geist, and O. Pietquin. Show me the way: Intrinsic motivation from demonstrations. In *AAMAS*, 2020.

T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*, 2016.

H. Küttler, N. Nardelli, T. Lavril, et al. TorchBeast: A PyTorch Platform for Distributed RL. *arXiv:1910.03552*, 2019.

M. Lopes, T. Lang, M. Toussaint, and P.-Y. Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *NIPS*, 2012.

S. Mohamed and D. Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *NIPS*, 2015.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *NIPS*, 2016.

G. Ostrovski, M. G. Bellemare, A. v. d. Oord, and R. Munos. Count-based exploration with neural density models. In *ICML*, 2017.

P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE on Evolutionary Computation*, 11(2):265–286, 2007.

B. O'Donoghue, I. Osband, R. Munos, and V. Mnih. The uncertainty bellman equation and exploration. In *ICML*, 2018.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

R. Raileanu and T. Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. In *ICLR*, 2019.

N. Savinov, A. Raichuk, D. Vincent, R. Marinier, M. Pollefeys, T. Lillicrap, and S. Gelly. Episodic curiosity through reachability. In *ICLR*, 2018.

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *ICML*, 2015.

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *the international conference on simulation of adaptive behavior: From animals to animats*, 1991.

B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *Workshop on Deep Reinforcement Learning (NIPS)*, 2015.

A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Y. Tang and S. Agrawal. Discretizing continuous action space for on-policy optimization. In *AAAI*, 2020.

H. Tang, R. Houthooft, D. Foote, A. Stooke, Chen, et al. # exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*, 2017.

S. B. Thrun. *Efficient exploration in reinforcement learning*. CMU-CS-92-102, 1992.

S. D. Whitehead. Complexity and cooperation in q-learning. In *Machine Learning Proceedings*, 1991.

A APPENDIX

A.1 ENVIRONMENTS INTRODUCED IN THE PAPER

Playground is designed to visually assess how exploration strategies behave when lots of object are present. Playground is a 14x14 grid, objects are always spawned at the same location but the color changes from episode to episode. An episode lasts 200 steps, no external reward is given and the agent is always spawned in the center, facing a random direction.

Ballpit environment The Ballpit environment was designed to assess how each algorithm is affected by the *Ballpit problem*. We build it on top of MultiRoomN4S6, a simple environment solved by all baselines. **No Ball** is the baseline environment. In **Small** instances, 1 random object is spawned, **More** contains 3 random objects and **Max** rooms are completely filled without blocking the agent path. We emphasize that the agent does not need to interact with anything (with the exception of door) to solve the environment and objects are never in the way of the agent.

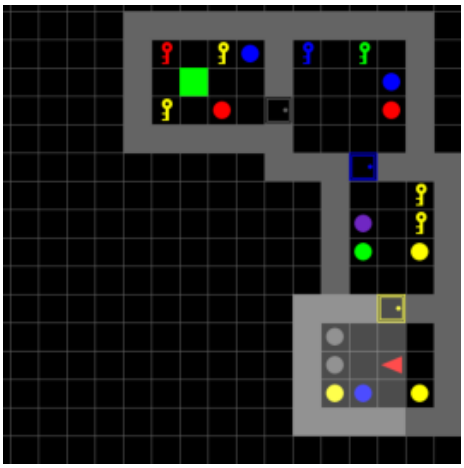

Figure 7: Illustration of BallPit-Max

ColorMaze ColorMaze is a variation of ObstructedMaze2Dlh. Before accessing the room containing keys within boxes, the agent must cross 4 colored rooms. A distractor room is placed at the beginning, leading to nothing. The colors are picked randomly at the beginning of an episode and remains the same until the episode's end. The agent must catch the blue ball within a 576 steps limit to receive the reward. One main difference between MultiRoom and ColorMaze is the absence of doors between colored room, which make this environment harder for RAM as no interaction is possible until the first door in room 4.

A.2 DECAY ILLUSTRATION

RAM reward is a function of the action ratio : $E^{\mathcal{H}}(a)$ the number of times an action impacted the state over $U^{\mathcal{H}}(a)$ the number of usage. The function acts as an exponential decay starting from 1 when the ratio is 0 (the action never impacted the environment so we want to reward it highly when it is actually modifying the state) and giving 0 reward when the ratio is 1 (The action a modifies the state all the time; thus, a is of low interest).

In Figure 8, we illustrate how the parameter η shape the intrinsic reward.

$$R(s_t, a_t, s_{t+1}) = \frac{\eta^{1-\frac{E^{\mathcal{H}}(a_t)}{U^{\mathcal{H}}(a_t)}} - 1}{\eta - 1} \tag{5}$$

Algorithm	RND	COUNT	RIDE	RAM
% of extrinsic reward collection	0%	0.4%	0.6%	**7%**

Table 1: Rewardless KeyCorridor : Percentage of times the agent collects extrinsic rewards while only receiving intrinsic rewards (average over 420 episodes)

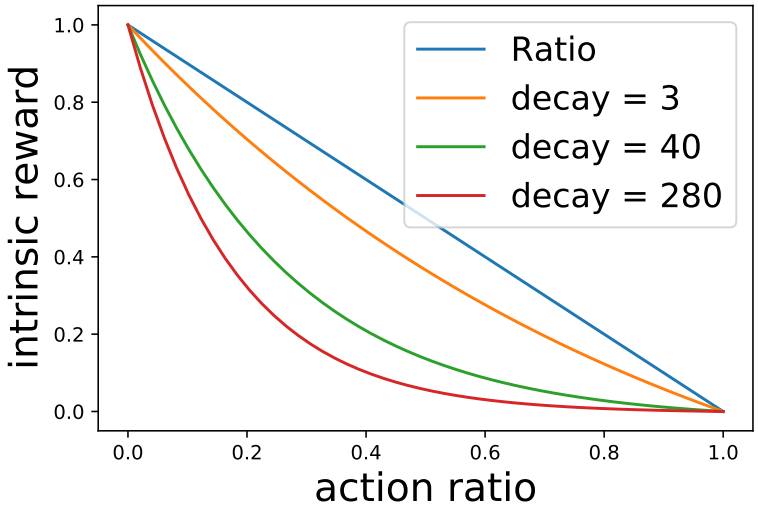

Figure 8: Ratio decay function illustrated for different values of η

A.3 RAM PARAMETERS

Parameters	Value
Intrinsic reward scaling	0.05
Ratio decay η	40
Learning rate	1e-5
Optimizer	RMSprop
Entropy cost	0.001
batch size	32
LSTM unroll length	100
Number of actors	42
Max norm gradient	40
Baseline cost	0.5
Discount factor	0.99

All parameters are kept the same across all environments, except for MultiRoomN12-S10 where the entropy cost is 0.0001.

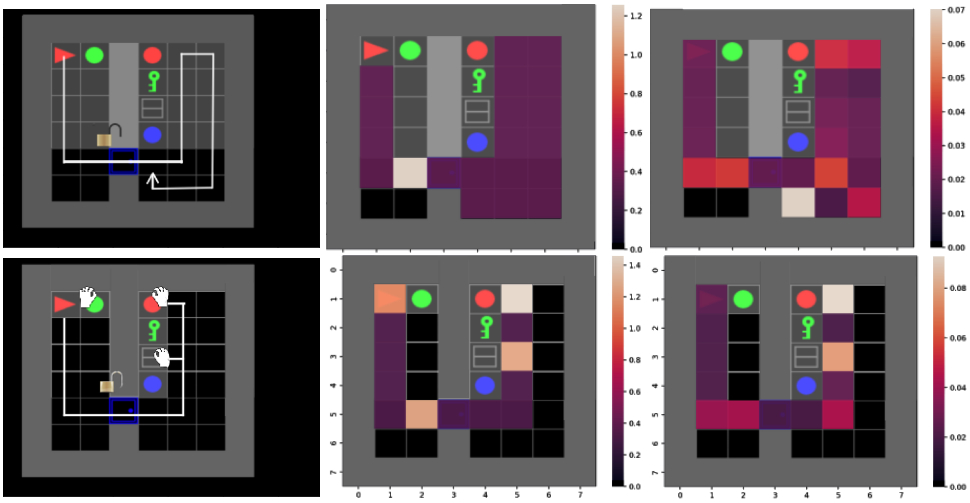

Figure 9: Two scenarios where we force paths and only inspect the intrinsic bonus given. (left) Forced path : Interaction scenario (middle) RAM (left) RIDE. RAM's intrinsic reward is less dense than other state-based methods such as RIDE. It may thus provides intermediate reward milestones that encourage deep exploration, rather than a stream of bonus which may blur salient interaction.