







Safe Reinforcement Learning Through Regret and State Restorations in Evaluation Stages

Timo P. Gros^{1,2,3} , Nicola J. Müller^{1,3} , Daniel Höller¹ ,
and Verena Wolf^{1,2} 

¹ Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{timopgros, nmueller, hoeller, wolf}@cs.uni-saarland.de

² German Research Center for Artificial Intelligence (DFKI),
Saarbrücken, Germany

{timo_philipp.gros, verena.wolf}@dfki.de

³ Center for European Research in Trusted Artificial Intelligence (CERTAIN),
Saarbrücken, Germany

Abstract. Deep reinforcement learning (DRL) has succeeded tremendously in many complex decision-making tasks. However, for many real-world applications standard DRL training results in agents with brittle performance because, in particular for safety-critical problems, the discovery of both, safe and successful strategies is very challenging. Various exploration strategies have been proposed to address this problem. However, they do not take information about the current safety performance into account; thus, they fail to systematically focus on the parts of the state space most relevant for training. Here, we propose *regret and state restoration in evaluation-based deep reinforcement learning* (RARE), a framework that introduces two innovations: (i) it combines safety evaluation stages with state restorations, i.e., restarting episodes in formerly visited states, and (ii) it exploits estimations of the regret, i.e., the gap between the policies' current and optimal performance. We show that both innovations are beneficial and that RARE outperforms baselines such as deep Q-learning and Go-Explore in an empirical evaluation.

Keywords: Safe Reinforcement Learning · Evaluation Stages ·
Regret · State Restorations

1 Introduction

Over the past decade, deep reinforcement learning (DRL) has made remarkable advancements in various complex decision-making tasks. Notably, it has demonstrated exceptional success in domains such as board games, including chess and go [39–41]. Additionally, its practical applications have extended to domains such as vehicle routing [32], robotics [22], and autonomous driving [36].

Despite these successes, DRL still suffers from significant weaknesses, especially in safety-critical applications. Safety objectives typically yield reward structures giving a positive signal for goal states and (highly) negative signal for unsafe states. However, DRL is known to perform poorly with such sparse reward structures since goal states are typically hard to reach [2, 23, 29, 35, 38].

There are sophisticated exploration strategies to handle sparse reward problems [4, 9, 12, 13]. While these strategies effectively explore the state space, their training process is executed without consideration of safety properties. The integration of safety considerations into the algorithmic framework is referred to as safe reinforcement learning [15].

In this work we introduce a safe deep reinforcement learning framework capable of both considering safety-properties and handling sparse rewards. We intertwine two ideas: (i) we systematically restore the agent’s starting state to promising states and (ii) we regularly evaluate the performance of the agent during training to inform the state restorations. Our state restoration procedure (innovation (i)) draws inspiration from Go-Explore [12] and ensures a systematic exploration of the state space. During training, we store states deemed relevant for learning in an archive and sample initial states for subsequent training episodes from this state archive.

In regular intervals during training, we evaluate their performance using deep statistical model-checking (DSMC) [18], which is a scalable verification technique [19] for the underlying Markov decision process and its current policy. This allows us to accurately estimate the corresponding regret (innovation (ii)). We propose two different techniques to focus the training on archived states

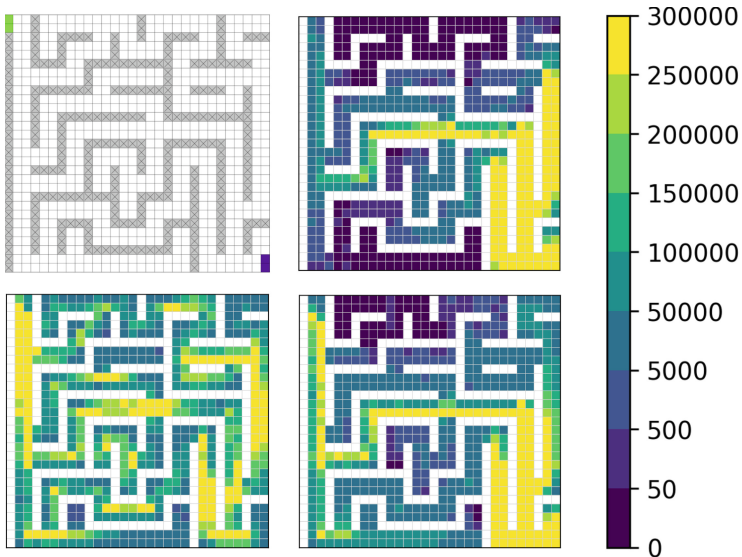


Fig. 1. Heatmaps visualizing how often states from each grid cell have been used to update the policy during training: map with start and goal (top left), state counts for DQN (top right), Go-Explore (bottom left), and RARE (bottom right).

where the regret is high, i.e., where the policy is far from optimal: starting more training episodes in such states or giving them a higher priority within the replay buffer that we use for batch updates of the agent’s neural network.

We consider an illustrative example in Fig. 1. The top left illustration depicts a map of the Racetrack, one of our benchmarks. The agent starts in one of the two purple cells in the lower right corner and must navigate through the grid up to the goal cells (marked in green).

The heatmaps show how often states from each grid cell have been used to update the policy. The top right illustration shows that standard DRL algorithms such as Deep Q-learning [30] (DQN), only start training from the initial states and fail to explore the state space sufficiently. Hence, the trained agent reaches the goal line only with a probability of 0.5. Algorithms with sophisticated exploration strategies, such as Go-Explore [12] (bottom left), reach the goal because they use state restorations, resulting in an increased goal-reaching probability of 0.65. However, they reconsider states for updates in a less systematic way compared to RARE (bottom right). RARE monitors the evaluated performance for a selected subset of states and computes the corresponding regrets. It is thus able to focus the learning on the most relevant parts. This safety-focused exploration allows RARE to achieve a goal-reaching probability of 0.74, which is significantly higher than those of DQN and Go-Explore.

To summarize, we propose the RARE framework, which is based on two innovations: (i) we combine deep statistical model-checking (DSMC) evaluation stages [16, 20] with state restorations, where we start new training episodes in carefully selected states based on the previously evaluated performance, and (ii) we exploit estimations of the regret [27], i.e., the gap between the current and optimal performance, to focus the training on high-regret states. We present our framework in Sect. 3 and provide an empirical evaluation comparing RARE to the baselines deep Q-learning, deep Q-learning with prioritized replay, and Go-Explore on two benchmarks in Sect. 4.

2 Background

Prior to presenting our contributions, this section covers the necessary background.

2.1 Markov Decision Processes and Deep Q-Learning

We consider discrete Markov decision processes (MDPs) with finite sets of states \mathcal{S} , actions \mathcal{A} , and an initial distribution μ over the set of initial states $\mathcal{I} \subseteq \mathcal{S}$. For states $s_t, s_{t+1} \in \mathcal{S}$, a transition from state s_t to s_{t+1} when choosing action $a_t \in \mathcal{A}$ corresponds to an experience $(s_t, a_t, r_{t+1}, s_{t+1})$, where $r_{t+1} \in \mathbb{R}$ is the obtained reward. Our goal is to compute a deterministic policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the sum of the discounted accumulated rewards, also called the *return*, $G_t = \sum_{k=t+1}^T \gamma^{k-t-1} R_k$, where R_k is the random variable of the k -th reward, T is the final time step, and $\gamma \in (0, 1]$ denotes the discount factor, which

balances the importance of immediate and future rewards. For a fixed policy π and a given state s_t , we define the *Q-value* of state s_t and action a_t

$$\begin{aligned} q_\pi(s_t, a_t) &= \mathbb{E}_\pi [G_t | S_t = s_t, A_t = a_t] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s_t, A_t = a_t \right] \end{aligned}$$

as the expected return G_t when taking action a_t in state s_t and following the policy π afterward.

Value-based algorithms, such as DQN, approximate an optimal policy π_* by learning the Q-values. Deep Q-learning uses a neural network (NN) to learn $q_*(s_t, a_t)$ of the optimal policy π_* . Let θ_i denote the NN's parameters in the i -th training iteration and let $Q_{\theta_i}(s_t, a_t)$ be the corresponding estimated Q-values. The network is updated according to the loss function

$$L(\theta_i) = \mathbb{E} \left[\left(r_{t+1} + \gamma \cdot \max_{a'} Q_{\theta'}(s_{t+1}, a') - Q_{\theta_i}(s_t, a_t) \right)^2 \right]$$

where the expectations are taken over experiences $(s_t, a_t, r_{t+1}, s_{t+1})$ uniformly sampled from an experience replay buffer \mathcal{B} [30]. To prevent unstable performance, it is common to optimize this network by using a network from a former iteration, the so-called target network with parameters θ' [30]. The soft update rule is $\theta = (1 - \tau) \cdot \theta_i + \tau \cdot \theta'$ with $\tau \in (0, 1)$ [14, 42].

The experiences are generated from an ϵ -greedy policy that chooses a random action with probability ϵ and an action yielding the highest Q-value with probability $1 - \epsilon$. Starting from a high initial value, ϵ is exponentially reduced during training until it meets a specified threshold.

A popular extension of the DQN algorithm, called *deep Q-learning with prioritized experience replay* (DQNPR) [37], is based on the assumption that experiences with low individual losses do not contribute as much to the learning process as experiences with high losses since they carry less relevant information. Hence, for each experience $(s_t, a_t, s_{t+1}, r_{t+1})$, DQNPR computes a sampling priority δ proportional to its loss. During network updates, each experience gets sampled with a probability proportional to δ , such that experiences with high losses are used more frequently to update the policy than experiences with small losses.

2.2 DSMC Evaluation Stages

DSMC evaluation stages [16, 20] leverage deep statistical model checking [18, 19] to analyze the performance of DRL agents. In deep statistical model checking, a policy π_θ represented by a neural network with weights θ resolves the non-determinism in the underlying MDP and, thus, Monte-Carlo simulations of the resulting Markov chain can be used to obtain an estimation of the expected value of any property of interest. Given $\epsilon_{err} > 0$ and some $\kappa \in (0, 1)$, DSMC achieves an error bound of $P(\text{error} > \epsilon_{err}) < \kappa$, where *error* is the difference between the true and the estimated value.

The evaluation stages (ES) are conducted at regular intervals during training to evaluate any *evaluation function* E using DSMC. Such an evaluation function can be any function of interest, making the incorporation of safety-critical approaches into the learning progress possible. The corresponding value for a state s is called the *evaluation value* $E_{\pi_\theta}(s)$.

Gros et al. [16,20] proposed two different methods for exploiting the information gained by DSMC evaluation during training:

1. *Evaluation-based initial distribution* (EID) evaluates the MDP’s initial states \mathcal{I} and shifts the initial distribution μ to start with a higher probability in areas with a lower evaluation value and vice versa.
2. In many DRL algorithms, a replay buffer is used. Following the idea of DQNPR, this replay buffer can be prioritized. However, instead of using the TD-error as the sampling priority, *evaluation-based prioritized replay* (EPR) bases a sample’s priority δ on the evaluation value of the initial state of the episode in which the sample was gathered.

Note that these approaches assume a large set of initial states \mathcal{I} , which sufficiently covers the state space.

2.3 Benchmarks

Racetrack is a commonly used reinforcement learning benchmark [7,17,21,43]. The task is to steer a car on a two-dimensional map from the starting line to reach a goal line without crashing into a wall or leaving the map. When the goal is reached or the car crashes, the episode terminates. The agent can change the velocity of the car by (limited) accelerating and decelerating, making

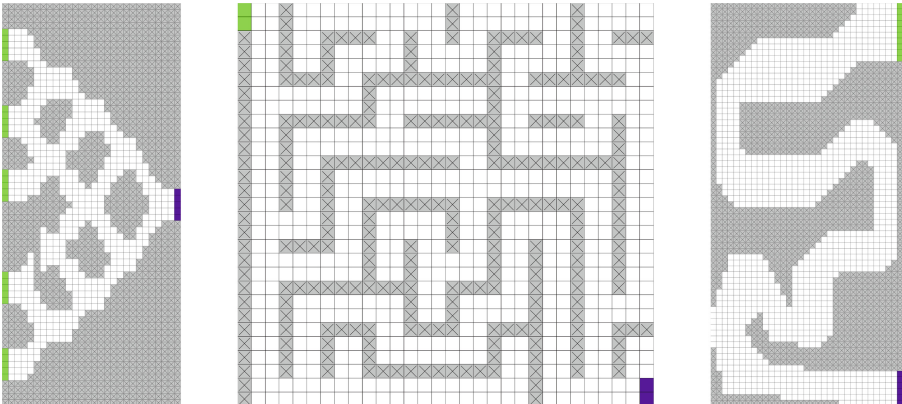


Fig. 2. Racetrack maps used in this paper: River (left), Maze (middle), and Hansenberg (right). The starting line is purple, the goal line is green, and the walls are gray.

foresighted decisions necessary. In addition, with some non-zero probability the selected acceleration will fail and, as a result, the velocity will remain unchanged.

The goal is to maximize the probability of reaching a goal state, which is one of the states at the goal line. A reward structure that assigns a reward of 1 to the goal states and 0 for all other states requires discounting to avoid solutions where the agent delays or even avoids goal-reaching. Moreover, for large maps such sparse reward structures are difficult to solve using standard DRL algorithms. Therefore, it is common to use a reward that is positive when reaching the goal, negative when crashing into a wall, and a zero elsewhere. Here, we use a reward of 100 for goal states and -20 for walls. Figure 2 shows three different maps.

MiniGrid is a benchmark widely used in the DRL community [10, 11, 13, 27, 34]. A MiniGrid environment corresponds to a discrete grid world where the agent needs to navigate through the grid and possibly interact with objects to solve the task. Figure 3 shows our custom DynObsDoor environment, where, starting at the top left corner, the agent needs to walk past walls and avoid collisions with randomly moving obstacles to reach the green goal cell. Furthermore, it must open the yellow door in the middle of the grid. Here, the goal is to maximize the probability of solving the task. To ensure early reward signals during training, we use a discounted reward of 1 when winning, -1 when losing, and zero else.

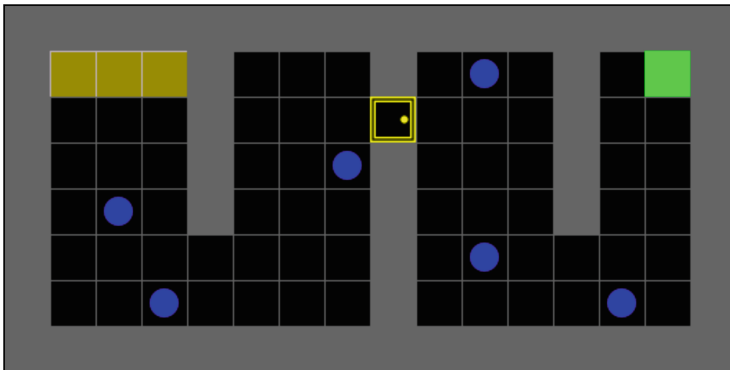


Fig. 3. The DynObsDoor environment. The starting cells are yellow, the goal cell is green, and the walls are gray. The blue dots are randomly moving obstacles. (Color figure online)

Additional statistics about the benchmarks can be found in the Appendix A.

3 Regret and State Restoration

This sections presents our framework *regret and state restoration in evaluation-based deep reinforcement learning* (RARE). We distinguish two variants:

(i) *regret and state restoration in evaluation-based initial distribution* (RAREID), and (ii) *regret and state restoration in evaluation-based prioritized replay* (RAREPR). As illustrated in Fig. 4, both share the idea of conducting evaluation stages and exploiting this information for effective training in two different ways. An additional description of our algorithm as pseudo-code can be found in Appendix B.

3.1 Overview

Throughout DRL training, we alternate between learning and evaluation stages at C-step intervals. During an evaluation stage, we evaluate all initial states \mathcal{I} and archived states \mathcal{A}_j , the latter were collected during the previous learning stage. In the subsequent learning stage, we sample the episodes’ starting states from the previously evaluated set of states, i.e., to all states $s \in \mathcal{I} \cup \mathcal{A}_j$. The key difference between RAREID and RAREPR lies in the utilization of the evaluation stages’ results during the learning stages: RAREID biases the sampling of episodes’ starting states $\mathcal{D}(\mathcal{A}_j \cup \mathcal{I})$ towards states with low evaluation values while RAREPR biases the policy updates by increasing the sampling priorities δ of the replay buffer \mathcal{B} for episodes with low evaluation values.

Initially, the archive is empty, i.e., $\mathcal{A}_0 = \{\}$, $\mathcal{D}(\mathcal{A}_0 \cup \mathcal{I})$ equals the MDP’s initial distribution $\mu(\mathcal{I})$ (RAREID), and the buffer’s sampling priorities δ are constant. In the following iterations, \mathcal{D} is determined in the last step of the previous evaluation stage, as explained below.

3.2 Learning Stage

During training we carry out the following steps:

(L1) Probabilistically Select State from Archive: In the case of RAREID, we sample a state s according to $\mathcal{D}(\mathcal{A}_j \cup \mathcal{I})$, as defined in step (E4). It assigns probabilities to states $s \in \mathcal{A}_j \cup \mathcal{I}$ according to the outcome of the evaluation stage. For RAREPR we also restart episodes at archived states, but we simply start episodes by first choosing between the original initial states (\mathcal{I}) and archived states (\mathcal{A}_j) with equal probability of 0.5 and subsequently select an initial state according to μ in case of \mathcal{I} and uniformly in case of \mathcal{A}_j .

(L2) Restore State: We restore the just sampled state s by either using the environment’s restore option, if available, or running a goal-conditioned policy from an initial state until s is reached [12].¹

(L3) Generate Experiences: We generate new experiences by applying the respective DRL algorithm starting from s .

¹ The goal-conditioned policy is tasked with reaching the sampled state s , after which the training continues with the regular policy π_θ . We follow the concept introduced by the Go-Explore algorithm [12].

(L4) *Determine Relevant States and Expand Archive:* For all states visited during the last episode, we check whether they are *relevant*, i.e., whether they contain information valuable for learning, and add those states to the new archive \mathcal{A}_{j+1} .² We apply two benchmark-independent heuristics to determine the relevance of a state:

1. *Value Heuristic:* We check for all states visited during the episode the smoothness of the corresponding state values, i.e., when transitioning from a state s_t to its successor s_{t+1} , we expect

$$|V_{\pi_\theta}(s_t) - (V_{\pi_\theta}(s_{t+1}) + r_t)| \quad (1)$$

to be small, where π_θ is the current policy and the state values are estimated by the NN. A significant difference indicates a high relevance because the network needs further updates related to the value of this state as the network has a poor estimation of the current state’s value.

2. *Novelty Heuristic:* We leverage recent work of *random network distillation* (RND) [9]. At the beginning of training, we randomly initialize a neural network that returns a real-valued output for each state input. We use our agent’s training observations to fit a second neural network to predict the output of the first one. As a result, for sufficiently explored states, the disparity between these networks is minimal. However, the difference is significant for infrequently or never encountered states. Consequently, this provides a reliable estimate of the relevance of a given state.

(L5) *Learn from Experiences:* We update the agent’s network using observations in the replay buffer \mathcal{B} . In the case of RAREID, we sample uniformly from \mathcal{B} , and for RAREPR, we sample based on the priorities δ determined by the most recent evaluation stage and defined in step (E4).

3.3 Evaluation Stage

After the learning stage, we add all states from the old archive \mathcal{A}_j to the new one \mathcal{A}_{j+1} , which is thus consisting of all relevant states. For further processing, this resulting expanded archive \mathcal{A}_{j+1} is then provided to the evaluation stage, which consists of four steps:

(E1) *Reduce Archive:* The number of interesting states may vary throughout the learning stages. Thus, we reduce the archive to a fixed size before performing the evaluation. We employ two different strategies to reduce the archive size, but keeping the states most relevant for learning.

² Note that the learning stage uses the states from the current archive \mathcal{A}_j for restarting episodes. However, archive \mathcal{A}_{j+1} contains the states currently stored for the next evaluation stage.

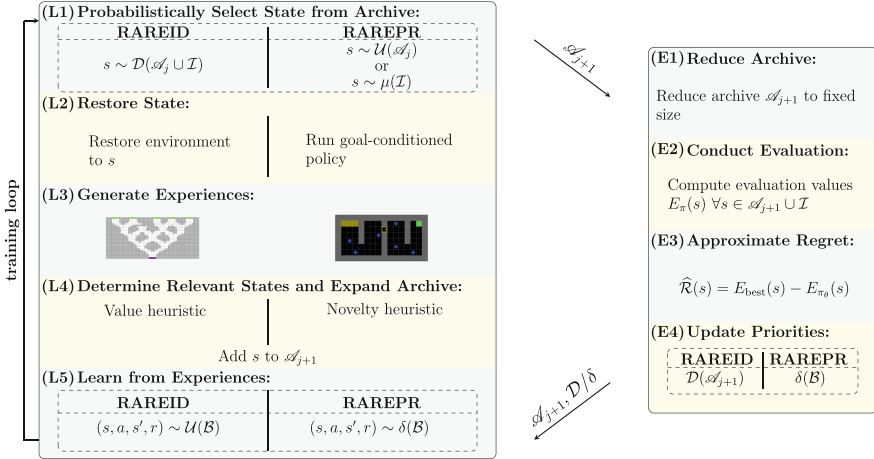


Fig. 4. Graphical overview of RARE: the left part (L1–L5) shows the learning stage, the right (E1–E4) the evaluation stage.

1. *Cluster Strategy*: This strategy uses the numerical state representation of the states to separate them into different clusters. As we want the archive to contain as much different states as possible, we only keep the most relevant state from each cluster w.r.t. to its heuristic value (see L4).
2. *Maximal Distance Strategy*: This strategy aims at reducing the archive to the states that cover the largest part of the state space possible. First, this strategy selects the most interesting state $s^* \in \mathcal{A}_{j+1}$ to be kept, meaning we initialize the reduced archive as $\mathcal{A}'_{j+1} = \{s^*\}$. Next, the strategy enforces that every further state s that is added to the reduced archive must fulfill

$$s = \operatorname{argmax}_{s \in \mathcal{A}_{j+1}} \min_{s' \in \mathcal{A}'_{j+1}} \|s, s'\|_2, \quad (2)$$

i.e., s has the largest minimum Euclidean distance on the numerical state representation to all of the already selected states $s' \in \mathcal{A}'_{j+1}$. At the end of the strategy, we set $\mathcal{A}_{j+1} = \mathcal{A}'_{j+1}$.³

(E2) Conduct Evaluation: We evaluate each state contained in the archive or the set of initial states, i.e., $s \in \mathcal{A}_{j+1} \cup \mathcal{I}$, w.r.t. the evaluation function using DSMC.

³ The application of the Euclidean distance assumes that the state representation is based on physical attributes, such as coordinates or velocities, which is often used in RL benchmarks. This method would also be effective for imaged-based state representation. Alternative distance metrics might need to be considered if this assumption is not fulfilled.

(E3) *Approximate Regret*: For each state $s \in S$, the regret is defined as the difference between the state values of the optimal and the current policy [6, 33], i.e.,

$$\text{Regret}(s) = v_*(s) - v_{\pi_\theta}(s), \quad (3)$$

where v_{π_θ} is the current policy with network weights θ .

As we are able to evaluate an arbitrary evaluation function and not just the value function, we here introduce the *evaluation regret*

$$\mathcal{R}(s) = E_*(s) - E_{\pi_\theta}(s), \quad (4)$$

where $E_*(s)$ is the evaluation value of the optimal policy given that we start in state s . Similarly, $E_{\pi_\theta}(s)$ is the evaluation value of the current policy π_θ with s as the initial state. Naturally, the true value of E_* is unknown. Thus, we follow the idea of Jiang et al.’s *MaxMC* method [27] and approximate the evaluation regret of states $s \in \mathcal{A}_{j+1} \cup \mathcal{I}$ as

$$\widehat{\mathcal{R}}(s) = E_{\text{best}}(s) - E_{\pi_\theta}(s), \quad (5)$$

where E_{best} denotes the best evaluation value encountered for all states in close Euclidean proximity to the given states’ description in previous evaluation stages. If no such state has formerly been evaluated, E_{best} is set to 1, ensuring the agent’s emphasis on this state during the next learning stage. Note that we linearly interpolate the evaluation values to $[0, 1]$ [20]. Further, we also linearly interpolate $\widehat{\mathcal{R}}$ to the same interval.

(E4) *Update Priorities*: We calculate a distribution for sampling episodes’ starting states $\mathcal{D}(\mathcal{A}_{j+1} \cup \mathcal{I})$ (RAREID) or the priorities to be used for the replay buffer δ (RAREPR) based on the estimated evaluation regret, respectively. While we want the agent to focus on states with a high regret, the initial states \mathcal{I} , as the task’s original objective, are of special interest. We define

$$\psi = \text{clip} \left(\frac{1}{|\mathcal{I}|} \sum_{s \in \mathcal{I}} E_{\pi_\theta}(s), 1 - \psi_{\max}, \psi_{\min} \right) \quad (6)$$

as the clipped average evaluation value of the initial states \mathcal{I} , where ψ_{\max} and ψ_{\min} are hyperparameters.

Considering RAREID, we set the distribution $\mathcal{D}(\mathcal{A}_{j+1} \cup \mathcal{I})$ such that the probability $p(s)$ to start in a certain state $s \in \mathcal{A}_{j+1} \cup \mathcal{I}$ is given by

$$p(s) = \begin{cases} (1 - \psi) \cdot \frac{\widehat{\mathcal{R}}(s) + \epsilon_p}{\sum_{s' \in \mathcal{A}_{j+1} \cup \mathcal{I}} (1 - \widehat{\mathcal{R}}(s') + \epsilon_p)} & s \in \mathcal{I} \\ \psi \cdot \frac{\widehat{\mathcal{R}}(s) + \epsilon_p}{\sum_{s' \in \mathcal{A}_{j+1} \cup \mathcal{I}} (1 - \widehat{\mathcal{R}}(s') + \epsilon_p)} & \text{else} \end{cases}, \quad (7)$$

where ϵ_p is a hyperparameter to ensure all samples have a non-zero probability. Moreover, $p(s)$ increases for states with a high evaluation regret and vice

versa. The additional weighting with ψ ensures that the initial states are considered often enough to prevent catastrophic forgetting [28], depending on their current evaluation.

Considering RAREPR, the replay priority of each state is

$$\delta(s_t) = \begin{cases} (1 - \psi) \cdot (\widehat{\mathcal{R}}(s_0) + \epsilon_p)^\alpha & \text{if } s_0 \in \mathcal{I} \\ \psi \cdot (\widehat{\mathcal{R}}(s_0) + \epsilon_p)^\alpha & \text{else} \end{cases}, \quad (8)$$

where ϵ_p is the minimal priority, and s_0 is the initial state of the corresponding episode of experience $(s_t, a_t, r_{t+1}, s_{t+1})$. The priority is higher if the evaluation regret is higher and vice versa. Again, the weighting with ψ considers the initial states particularly.

3.4 Regret and State Restoration in Evaluation-Based Deep Q-Learning

We use Mnih et al.’s deep Q-learning [30] as a base algorithm to implement RARE. Note that RARE can easily be adapted to any kind of (deep) reinforcement learning; particularly RAREID to any such algorithm and RAREPR to any algorithm using a replay buffer.

4 Empirical Evaluation

Next we provide a empirical evaluation of RARE, comparing it with three baselines: DQN and DQNPR, since our approach is based on them, and Go-Explore, which also uses the idea of state restoration.

We conduct experiments on Racetrack and MiniGrid as introduced in Sect. 2.3, using two different evaluation functions: (i) the average cumulative reward, and (ii) the goal-reaching probability as a typical safety objective.

The following results were all obtained by using DSMC with $\kappa = 0.05$ and $\epsilon_{err} = 0.01$ (goal-reaching probabilities), $\epsilon = 1$ (return for Racetrack), and $\epsilon_{err} = 0.01$ (return for MiniGrid). We perform multiple trainings and report the average result over all agents that were able to solve the task, i.e., to reach the goal at all.⁴ All agents were trained by using an Intel Xeon E5-2698 v4 processor with 100 GB RAM. For further experiment details, we refer to Appendix C.

4.1 Racetrack

Figure 5 provides results for three different maps of the Racetrack. We report the performance from the initial states, i.e., the benchmark’s original task. In Fig. 5 (left) the return was used as evaluation function and we provide the average obtained return. While on the River-deadend map, RARE performs roughly equal to the baselines (with an outlier by Go-Explore), both RAREID and

⁴ Otherwise we report the training as failed.

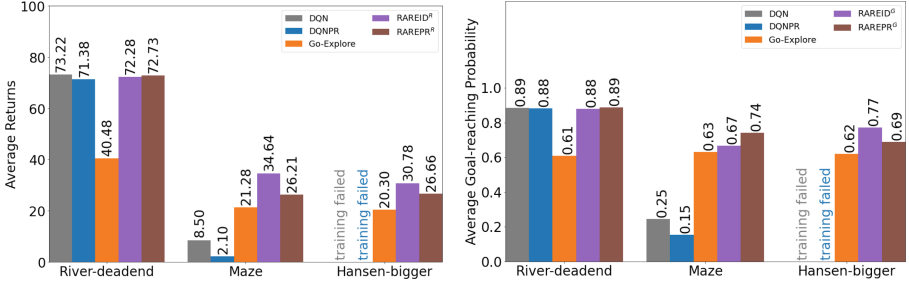


Fig. 5. Average return (left) and goal-reaching probability (right) achieved by DQN, DQNPR, Go-Explore, RAREID, and RAREPR on Racetrack. ‘Training failed’ refers to the case that the agent was not able to find the goal during training.

RAREPR outperform the baselines on the more sophisticated maps Maze and Hansen-bigger. In Fig. 5 (right) we give the goal-reaching probability, which was also used as the evaluation function. Again RAREID and RAREPR outperform the baselines on the maps Maze and Hansen-bigger.

4.2 MiniGrid

Figure 6 shows the agents’ performance for the MiniGrid benchmark. We compare the returns (left) and goal-reaching probabilities (right). Accordingly, these were also the evaluation functions used for RARE.

In contrast to Racetrack, here, the primary advantage of RARE over DQN and DQNPR is that RARE is capable of solving the benchmark, while both DQN and DQNPR fail to find the goal. Go-Explore is able to find the goal, but still, both RARE algorithms clearly show superior performance.

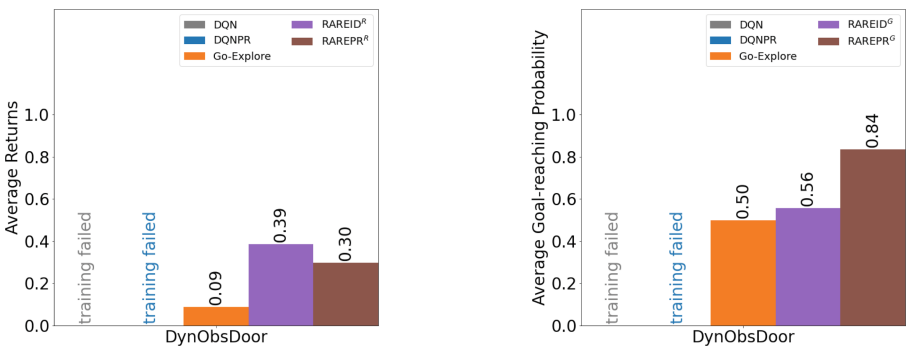


Fig. 6. Average return (left) and goal-reaching probability (right) achieved by DQN, DQNPR, Go-Explore, RAREID, and RAREPR on MiniGrid. Training failed means that the agent was not able to find the goal during training.

5 Ablation Study

This section is dedicated to examining the impact of the two parts of the RARE algorithm, namely (i) using state restorations, and (ii) using the regret estimation. To do so, we specifically designed the *Maze-extended* map, shown in Fig. 7 (top left), where, due to the environment’s uncertainty, the narrow connections between the bottom cells and the rest of the map lead to a reduced maximum achievable goal-reaching probability from these bottom cells. Note that this map is intended to show the influence of the two novelties included, while the maps used in the paper are more general and have been used previously in the literature.

We remove the regret estimation from RARE by replacing $\widehat{\mathcal{R}}(s)$ with $(1-E(s))$ in Eqs. (7) and (8), i.e., we compute the priorities and distribution only based on the evaluation values instead of computing them based on the regret. For the sake of clarity, we write REID (state restoration in evaluation-based initial distribution) or REPR (state restoration in evaluation-based experience replay), respectively, when we refer to the algorithms without the regret estimation.

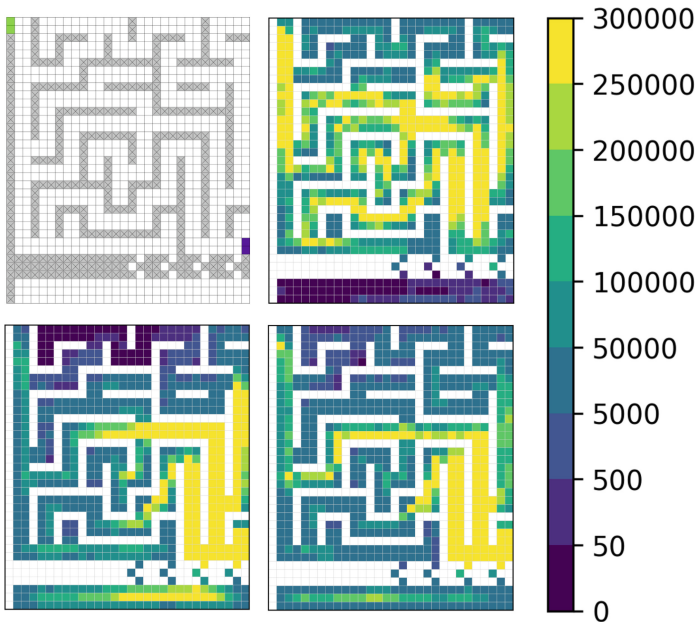


Fig. 7. Racetrack’s Maze-extended map (top left), followed by heatmaps visualizing how often states from each grid cell have been used for policy updates during training using Go-Explore (top right), REPR (bottom left), and RAREPR (bottom right).

We compare Go-Explore, REID, REPR, RAREID, and, RAREPR agents that were trained on the Maze-extended map and evaluated using DSMC with

$\kappa = 0.05$ and $\epsilon = 0.01$ for the goal-reaching probability, or $\epsilon = 1$ for the return, respectively. For each evaluation-based algorithm, we trained agents using the return and the goal-reaching probability as the evaluation function. We trained for multiple random seeds and report the averaged results.

State Restorations with Evaluation Stages. Consider Fig. 7, which depicts how often each grid cell was considered during training. In the first heatmap (top right), we see that Go-Explore explores the map almost uniformly, with the exception of the tight alley at the bottom of the map, where it does not find a solution at all.

In contrast, the next heatmap (bottom left) shows that REPR focuses more on important regions of the state space. Further, it repeatedly considers the alley at the bottom, which is the region of the state space that is most difficult to solve.

Regret Approximation. From Fig. 7 now additionally take into account the last heatmap (bottom right), depicting RAREPR’s consideration of grid cells. While REPR, in contrast to Go-Explore, was able to solve the more difficult bottom part, these grid cells were considered too often during training, as there the maximal goal-reaching probability and, thus, also the evaluation value, is lower than it is for the rest of the map. By additionally using the regret approximation, RAREPR considers these states often enough, but does not overtrain where performance cannot be improved further.

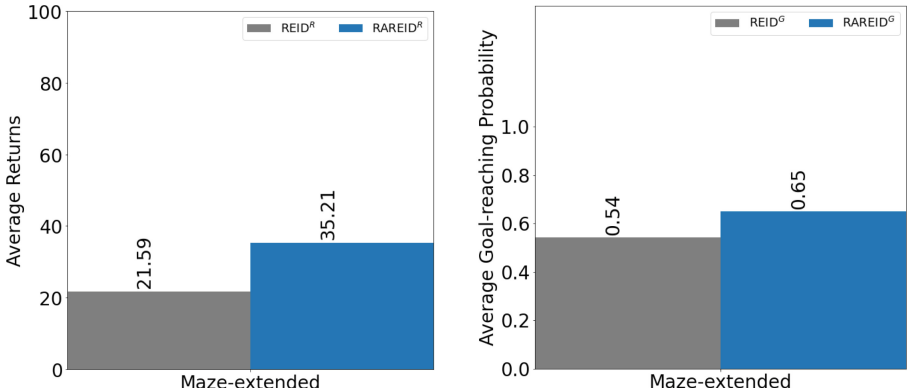


Fig. 8. Average return (left) and goal-reaching probability (right) achieved by REID, and RAREID on the Maze-extended map.

To strengthen that finding, we additionally compare the performance with and without using the regret approximation. Figure 8 shows that RAREID achieves a significantly increased performance for both average return and goal-reaching probability compared to REID.

6 Related Work

DSMC evaluation stages have already been proposed by Gros et al. [20]. However, that work was based on the assumption of having a broad set of initial states which sufficiently cover the state space. In contrast, RARE operates independently of the number or the distribution of initial states by evaluating promising visited states (including states $s \notin \mathcal{I}$) and restoring subsequent training episodes’ starting states to them. Moreover, instead of only considering their performance, we here propose to consider the estimated regret as an indicator for relevance.

Our work directly relates to *Go-Explore*, as our approach of state restorations combined with evaluation stages was inspired by the work of Ecoffet et al. [12]. Similarly to RARE, Go-Explore stores all visited states in its archive, yet instead of subsequently drawing states where safety performance is poor, as RARE does, Go-Explore draws the least frequently seen states. Thus, Go-Explore seeks to explore as much of the state space as possible, whereas our RARE method focuses the training on parts of the state space most relevant to safety objectives. Also, Go-Explore does not take the regret into account.

The RARE approach constitutes a framework within safe reinforcement learning. According to García and Fernández’s taxonomy of safe reinforcement learning [15], it falls into the category risk-directed exploration. In contrast to other safe RL approaches that influence the exploration, RARE operates without any external knowledge and is further able to influence the exploration process according to any safety property. Moreover, former risk-directed exploration algorithms ensure safety by already avoiding unsafe parts of the state space during the training, i.e., they use risk-averted exploration. However, this is contingent upon the presence of an entity capable of identifying unsafe states. If this entity is derived from a learning process [3, 8], it could potentially lead to the erroneous avoidance of certain parts of the state space that are incorrectly classified as unsafe. In contrast, our algorithm especially trains where the safety properties are (currently) harmed without any prior knowledge needed. Thus, the framework learns how to behave in these parts of the state space by explicitly confronting the unsafe states, which can best be described as *risk-confronting exploration*. To the best of our knowledge, we are the first ones to pursue this sub-approach of risk-directed exploration.

Recent work of Hasanbeig, Abate and Kroening [24] on safe reinforcement learning introduces a technique to include a property expressed as an LTL formula and synthesizes policies to optimize the probability of fulfilling that LTL property. However, this method, while allowing for the specification of complex tasks, does not address the problems tied to safety-critical reward structures, such as very sparse rewards not suited for learning without additional exploration. Similarly, Hasanbeig, Abate and Kroening use LTL properties to derive meaningful reward functions for unknown environments. Applying their method to the property used in this paper (optimizing goal-reaching probability without getting stuck in an unsafe state) yields the exact reward function we employ: positive when reaching the goal, negative when harming safety, and zero oth-

erwise. Consequently, this otherwise effective approach proves unhelpful in our case.

Further, state restorations are related to the well-established idea of *importance splitting*, where a restart is conducted from rare but relevant paths [26, 31]. A comparison of both methods cannot be made straightforwardly, as importance splitting is based on the assumption of knowing the state space, while RARE is not.

7 Conclusion and Future Work

In this paper we introduced the RARE framework and proposed two variants of it, namely RAREID and RAREPR. RARE uses a combination of two ideas intended to improve policy quality when using deep reinforcement learning in safety-critical applications. First, state restorations combined with DSMC evaluation stages, and second, utilizing the regret estimation. Our empirical evaluation shows that RARE outperforms the standard baseline of deep Q-learning and the related approach of Go-Explore.

In the future, we plan to incorporate a latent space representation into our framework. This will enable automatic clustering of the observed states. We think this is promising because it might help with (i) getting a better selection of interesting states in the archives, and (ii) enabling an even better estimation of the maximal evaluation value and, thus, also improving our regret approximation.

In view of Anderson’s recent work [3], we aim to investigate whether combining RARE with model-based deep reinforcement learning might allow learning a shield [1, 5, 25] to further improve the policies’ safety performance at test time. Concretely, RARE could be used to gather experiences for learning an environment model that accurately captures the state space regions most relevant to safety. At test time, the model would then be used to compute safe actions whenever the policy returns an action likely leading to an unsafe state.

Also, even though this framework was specially designed to operate on sparse reward tasks, a comparison on dense reward benchmarks is of interest, as we expect our technique also to be beneficial in such settings.

Acknowledgments. This work was partially funded by the European Union’s Horizon Europe Research and Innovation program under the grant agreement TUPLES No 101070149, by the German Research Foundation (DFG) under grant No. 389792660, as part of TRR 248, see <https://perspicuous-computing.science>, by the German Research Foundation (DFG) - GRK 2853/1 “Neuroexplicit Models of Language, Vision, and Action” - project number 471607914, and by the European Regional Development Fund (ERDF) and the Saarland within the scope of (To)CERTAIN.

We thank the anonymous reviewers for their careful reading of our manuscript and their insightful comments and suggestions.

A Benchmark Statistics

Racetrack. In Racetrack, the number of states is given through all the possible positions in the map and the maximal achievable velocity. As the most of the states cannot be reached with the maximal velocity, this is an upper bound and not an exact number.

Map	River	Maze	Hansen-bigger
States	99764	155136	299008

MiniGrid. In MiniGrid, the number of states is given through all possible combinations of the agent’s position, it’s direction, whether the door is open, and the positions of the moving obstacles. The latter is the responsible for the huge state space even for relatively small maps. For DynObsDoor, we have $\sim 3.04 \cdot 10^{10}$ states.

B Pseudo-code

The components of the RARE algorithm are highlighted in [blue](#).

Algorithm 1. Regret and State Restoration in Evaluation-based Deep Reinforcement Learning

```

1: initialize archive  $\mathcal{A}_0 = \{\}$ 
2: initialize  $\psi = 0.0$ 
3: for episodes  $e = 0$  to  $E - 1$  do
4:   sample  $s_0$  according to  $\begin{cases} p(s_0) & // \text{[RAREID]} \\ \mu(s_0) & // \text{[RAREPR]} \end{cases}$ 
5:   for steps  $t = 0$  to  $T - 1$  do
6:     apply heuristic  $h$  to  $s_t$  & add  $(s_t, h(s_t))$  to  $\mathcal{A}_{j+1}$ 
7:     with probability  $\epsilon$  select random action  $a_t \in \mathcal{A}(s_t)$ 
8:     otherwise with probability  $1 - \epsilon$  select  $a_t = \operatorname{argmax}_{a \in \mathcal{A}(s_t)} Q_{\theta_t}(s_t, a)$ 
9:     execute  $a_t$ ; observe  $s_{t+1}$  and  $r_{t+1}$ 
10:    compute  $\delta = \begin{cases} \text{constant} & // \text{[RAREID]} \\ \text{Equation 8} & // \text{[RAREPR]} \end{cases}$ 
11:    store  $(s_t, a_t, r_{t+1}, s_{t+1}, \delta)$  in replay buffer  $\mathcal{B}$ 
12:    every  $K$  steps do
13:      sample mini-batch of experiences  $(s_j, a_j, r_{j+1}, s_{j+1}, \delta)$ 
      from  $\mathcal{B}$  w.r.t.  $\delta$ 
14:      set target  $y_j = \begin{cases} r_{j+1} & s_{j+1} \text{ terminal} \\ r_{j+1} + \gamma \cdot \max_{a'} Q_{\theta'}(s_{j+1}, a') & \text{else} \end{cases}$ 
15:      perform gradient descent step on loss  $(y_j - Q_{\theta}(s_j, a_j))^2$ 
16:      soft-update the network weights  $\theta' = (1 - \tau) \cdot \theta_t + \tau \cdot \theta'$ 
17:    end every
18:  end for
19:  if  $e > W$  then
20:    every  $L$  episodes do
21:       $\mathcal{A}_{j+1} = \text{reduceArchive}(\mathcal{A}_{j+1})$ 
22:      compute  $\widehat{\mathcal{R}}(s) = E_{\text{best}}(s) - E_{\pi_{\theta}}(s)$  for all  $s \in \mathcal{A}_{j+1} \cup \mathcal{I}$ 
23:      update  $\psi$ 
24:    end every
25:  end if
26: end for

```

C Hyperparameters

Hyperparameters that are used in multiple algorithms but only have one table entry have the same value in all instances.

Parameter	Description	Value
-----------	-------------	-------

DQN:

E	Number of episodes (Racetrack)	100.000
E	Number of episodes (MiniGrid)	40.000
T	Maximum episode length (Racetrack)	100
T	Maximum episode length (MiniGrid)	200
γ	Discount factor	0.99
K	Q-network update frequency	4
	Batch size	512
τ	Soft update coefficient	0.001
ϵ_{start}	Initial exploration coefficient of ϵ -greedy policy	1
ϵ_{decay}	Decay factor of ϵ in each episode	0.999
ϵ_{end}	Value of ϵ at the end of the training	0.05
$ \mathcal{B} $	Size of replay buffer	10^8
α_{Adam}	Learning rate of Adam optimizer (Racetrack)	$8 \cdot 10^{-4}$
α_{Adam}	Learning rate of Adam optimizer (MiniGrid)	0.0001
	Probability of acceleration failing (Racetrack, River / Maze / Hansen)	0.5/ 0.25/ 0.25

DQNPR:

α	Prioritization coefficient for sampling priorities	1
ϵ_p	Minimum priority	10^{-6}

RAREID and RAREPR:

W	Number of pre-training episodes	10.000
L	Evaluation frequency	10.000
ψ_{min}		0.2
ψ_{max}		0.2
	Archive size after reduction (Racetrack, River / Maze / Hansen)	127/ 151/ 292
	Archive size after reduction (MiniGrid)	17
ϵ_p	Minimum priority	0.2
ϵ_{err}	Error in DSMC's evaluation during training (Racetrack, GRP / Return)	0.05/ 4
ϵ_{err}	Error in DSMC's evaluation during evaluation (Racetrack, GRP / Return)	0.01/ 1
ϵ_{err}	Error in DSMC's evaluation during training (MiniGrid, GRP / Return)	0.05/ 0.1
ϵ_{err}	Error in DSMC's evaluation during evaluation (MiniGrid, GRP / Return)	0.01/ 0.01
κ	Probability that DSMC's error is at most ϵ_{err}	0.05

Go-Explore

	Number of demonstrations in robustification	10
	Number of episodes during each exploration phase	100
	Maximum number of archived states (Racetrack, River / Maze / Hansen)	509/ 606/ 1168
	Maximum number of archived states (MiniGrid)	69

References

1. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI), pp. 2669–2678. AAAI Press (2018)
2. Amit, R., Meir, R., Ciosek, K.: Discount factor as a regularizer in reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning (ICML), pp. 269–278. PMLR (2020)
3. Anderson, G., Chaudhuri, S., Dillig, I.: Guiding safe exploration with weakest pre-conditions. In: The Eleventh International Conference on Learning Representations (2022)
4. Andrychowicz, M., et al.: Hindsight experience replay. In: Advances in Neural Information Processing Systems, pp. 5048–5058 (2017)
5. Avni, G., Bloem, R., Chatterjee, K., Henzinger, T.A., Könighofer, B., Pranger, S.: Run-time optimization for learned controllers through quantitative games. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 630–649. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_36
6. Azar, M.G., Osband, I., Munos, R.: Minimax regret bounds for reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning (ICML), pp. 263–272. PMLR (2017)
7. Baier, C., Christakis, M., Gros, T.P., Groß, D., Gumhold, S., Hermanns, H., Hoffmann, J., Klauck, M.: Lab conditions for research on explainable automated decisions. In: Heintz, F., Milano, M., O’Sullivan, B. (eds.) TAILOR 2020. LNCS (LNAI), vol. 12641, pp. 83–90. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-73959-1_8
8. Bharadhwaj, H., Kumar, A., Rhinehart, N., Levine, S., Shkurti, F., Garg, A.: Conservative safety critics for exploration. In: Proceedings of the 9th International Conference on Learning Representations (ICLR). OpenReview (2021)
9. Burda, Y., Edwards, H., Storkey, A.J., Klimov, O.: Exploration by random network distillation. In: Proceedings of the 7th International Conference on Learning Representations (ICLR). OpenReview (2019)
10. Campero, A., Raileanu, R., Küttler, H., Tenenbaum, J.B., Rocktäschel, T., Grefenstette, E.: Learning with AMIGo: adversarially motivated intrinsic goals. In: Proceedings of the 9th International Conference on Learning Representations (ICLR). OpenReview (2021)
11. Chevalier-Boisvert, M., et al.: BabyAI: a platform to study the sample efficiency of grounded language learning. In: Proceedings of the 7th International Conference on Learning Representations (ICLR). OpenReview (2019)
12. Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K.O., Clune, J.: First return, then explore. *Nature* **590**(7847), 580–586 (2021)
13. Flet-Berliac, Y., Ferret, J., Pietquin, O., Preux, P., Geist, M.: Adversarially guided actor-critic. In: Proceedings of the 9th International Conference on Learning Representations (ICLR). OpenReview (2021)
14. Fujita, Y., Nagarajan, P., Kataoka, T., Ishikawa, T.: ChainerRL: a deep reinforcement learning library. *J. Mach. Learn. Res.* **22**, 77:1–77:14 (2021)
15. García, J., Fernández, F.: A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **16**, 1437–1480 (2015)
16. Gros, T.P., et al.: DSMC evaluation stages: fostering robust and safe behavior in deep reinforcement learning - extended version. *ACM Trans. Model. Comput. Simulat.* **33**(4), 17:1–17:28 (2023). <https://doi.org/10.1145/3607198>

17. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Köhl, M.A., Wolf, V.: MoGym: using formal models for training and verifying decision-making agents. In: Shoham, S., Vitzel, Y. (eds.) CAV 2022, Part II, pp. 430–443. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13188-2_21
18. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Deep statistical model checking. In: Gotsman, A., Sokolova, A. (eds.) FORTE 2020. LNCS, vol. 12136, pp. 96–114. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50086-3_6
19. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Analyzing neural network behavior through deep statistical model checking. *Int. J. Softw. Tools Technol. Transfer* **25**(3), 407–426 (2023)
20. Gros, T.P., Höller, D., Hoffmann, J., Klauck, M., Meerkamp, H., Wolf, V.: DSMC evaluation stages: fostering robust and safe behavior in deep reinforcement learning. In: Abate, A., Marin, A. (eds.) QEST 2021. LNCS, vol. 12846, pp. 197–216. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85172-9_11
21. Gros, T.P., Höller, D., Hoffmann, J., Wolf, V.: Tracking the race between deep reinforcement learning and imitation learning. In: Gribaudo, M., Jansen, D.N., Remke, A. (eds.) QEST 2020. LNCS, vol. 12289, pp. 11–17. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59854-9_2
22. Gu, S., Holly, E., Lillicrap, T.P., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3389–3396. IEEE Press (2017)
23. Hare, J.: Dealing with sparse rewards in reinforcement learning. arXiv preprint [arXiv:1910.09281](https://arxiv.org/abs/1910.09281) (2019)
24. Hasanbeig, M., Abate, A., Kroening, D.: Logically-constrained reinforcement learning. arXiv preprint [arXiv:1801.08099](https://arxiv.org/abs/1801.08099) (2018)
25. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe reinforcement learning using probabilistic shields. In: Proceedings of the 31st International Conference on Concurrency Theory (CONCUR), pp. 3:1–3:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2020)
26. Jegourel, C., Legay, A., Sedwards, S.: Importance splitting for statistical model checking rare properties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 576–591. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_38
27. Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J.N., Grefenstette, E., Rocktäschel, T.: Replay-guided adversarial environment design. In: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS), pp. 1884–1897 (2021)
28. Kirkpatrick, J., et al.: Overcoming catastrophic forgetting in neural networks. arXiv preprint [arXiv:1612.00796](https://arxiv.org/abs/1612.00796) (2016)
29. Knox, W.B., Stone, P.: Reinforcement learning from human reward: discounting in episodic tasks. In: Proceedings of the 21st IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 878–885. IEEE Press (2012)
30. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015)
31. Morio, J., Pastel, R., Le Gland, F.: An overview of importance splitting for rare event simulation. *Eur. J. Phys.* **31**(5), 1295 (2010)

32. Nazari, M., Oroojlooy, A., Snyder, L.V., Takác, M.: Reinforcement learning for solving the vehicle routing problem. In: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS), pp. 9861–9871 (2018)
33. Parker-Holder, J., et al.: Evolving curricula with regret-based environment design. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 17473–17498. PMLR (2022)
34. Raileanu, R., Rocktäschel, T.: RIDE: rewarding impact-driven exploration for procedurally-generated environments. In: Proceedings of the 8th International Conference on Learning Representations (ICLR). OpenReview (2020)
35. Riedmiller, M.A., et al.: Learning by playing solving sparse reward tasks from scratch. In: Proceedings of the 35th International Conference on Machine Learning (ICML), pp. 4341–4350. PMLR (2018)
36. Sallab, A.E., Abdou, M., Perot, E., Yogamani, S.: Deep reinforcement learning framework for autonomous driving. *Electron. Imaging* **2017**(19), 70–76 (2017)
37. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. In: Proceedings of the 4th International Conference on Learning Representations (ICLR) (2016)
38. Schwartz, A.: A reinforcement learning method for maximizing undiscounted rewards. In: Proceedings of the 10th International Conference on Machine Learning (ICML), pp. 298–305. Morgan Kaufmann (1993)
39. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. *Nature* **529**(7587), 484–489 (2016)
40. Silver, D., et al.: A General reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**(6419), 1140–1144 (2018)
41. Silver, D., et al.: Mastering the game of go without human knowledge. *Nature* **550**(7676), 354–359 (2017)
42. Stooke, A., Abbeel, P.: rlpyt: a research code base for deep reinforcement learning in PyTorch. arXiv preprint [arXiv:1909.01500](https://arxiv.org/abs/1909.01500) (2019)
43. Sutton, R.S., Barto, A.G.: Reinforcement Learning - An Introduction, Adaptive Computation and Machine Learning. MIT Press (1998)