Information Fusion 85 (2022) 1–22

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full length article

Exploration in deep reinforcement learning: A survey☆

Pawel Ladosz [a], Lilian Weng [b], Minwoo Kim [a], Hyondong Oh [a,*]

[a] Department of Mechanical Engineering, Ulsan National Institute of Science and Technology (UNIST), 50 UNIST-gil, Ulsan, Republic of Korea
[b] OpenAI LP, 3180 18th St, San Francisco, CA 94110, United States of America

ARTICLE INFO

Keywords:
Deep reinforcement learning
Exploration
Intrinsic motivation
Sparse reward problems

ABSTRACT

This paper reviews exploration techniques in deep reinforcement learning. Exploration techniques are of primary importance when solving sparse reward problems. In sparse reward problems, the reward is rare, which means that the agent will not find the reward often by acting randomly. In such a scenario, it is challenging for reinforcement learning to learn rewards and actions association. Thus more sophisticated exploration methods need to be devised. This review provides a comprehensive overview of existing exploration approaches, which are categorised based on the key contributions as: reward novel states, reward diverse behaviours, goal-based methods, probabilistic methods, imitation-based methods, safe exploration and random-based methods. Then, unsolved challenges are discussed to provide valuable future research directions. Finally, the approaches of different categories are compared in terms of complexity, computational effort and overall performance.

1. Introduction

In numerous real-world problems, the outcomes of a certain event are only visible after a significant number of other events have occurred. These types of problems are called sparse reward problems since the reward is rare and without a clear link to previous actions. We note that sparse reward problems are common in a real world. For example, during search and rescue missions, the reward is only given when an object is found, or during delivery, the reward is only given when an object is delivered. In sparse reward problems, thousands of decisions might need to be made before the outcomes are visible. Here, we present a review on a group of techniques that can solve this issue, namely exploration in reinforcement learning.

In reinforcement learning, an agent is given a state and a reward from the environment. The task of the agent is to determine an appropriate action. In reinforcement learning, the appropriate action is such that it maximises the reward, or it could be said that the action is exploitative. However, solving problems with just exploitation may not be feasible owing to reward sparseness. With reward sparseness, the agent is unlikely to find a reward quickly, and thus, it has nothing to exploit. Thus, an exploration algorithm is required to solve sparse reward problems.

The most common technique for exploration in reinforcement learning is random exploration [1]. In this type of approach, the agent decides what to do randomly regardless of its progress. The most commonly-used technique of this type, called ϵ-greedy, uses the time decaying parameter ϵ to reduce exploration over time. This can theoretically solve the sparse reward problem given a sufficient amount of time. However, this is often impractical in real-world applications because learning times can be very large. However, we note that even just with random exploration, deep reinforcement learning has shown some impressive performance in Atari games [2], Mujoco simulator [3], controller tuning [4], autonomous landing [5], self-driving cars [6] and healthcare [7].

Another solution for exploration could be reward shaping. In reward shaping, the designer 'artificially' imposes a reward more often. For example, for search and rescue missions, agents can be given a negative reward every time they do not find the victim. However, reward shaping is a challenging problem that is heavily dependent on the experience of the designer. Punishing the agent too much could lead to the agent not moving at all [8], while rewarding it too much may cause the agent to repeat certain actions infinitely [9]. Thus, with the issues of random exploration and reward shaping, there is a need for more sophisticated exploration algorithms.

While exploration in reinforcement learning was considered as early as 1991 [10,11], it is still under development. Recently, exploration has shown a significant gain in performance compared to non-exploratory

☆ This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Republic of Korea (2020R1A6A1A03040570), National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2020R1F1A1049066), and Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF), Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science and ICT, the Republic of Korea (2020M3C1C1A01082375).

* Corresponding author.
E-mail addresses: pladosz@unist.ac.kr (P. Ladosz), lilian@openai.com (L. Weng), red9395@unist.ac.kr (M. Kim), h.oh@unist.ac.kr (H. Oh).

https://doi.org/10.1016/j.inffus.2022.03.003
Received 15 July 2021; Received in revised form 6 December 2021; Accepted 18 March 2022
Available online 31 March 2022

algorithms: Diversity is all you need (DIYAN) [12] improved on MuJoCo benchmarks; random network distillation (RND) [13] and pseudocounts [14] were the first to score on difficult Montezuma's Revenge problem; and Agent57 [15] is the first to beat humans in all 57 Atari games.

This review focuses on exploratory approaches which fulfil at least one of the following criteria: (i) determines the exploration degree based on the agent's learning, (ii) actively decides to take certain actions in hopes of finding new outcomes, and (iii) motivates itself to continue exploring despite a lack of environmental rewards. In addition, this review focuses on approaches that have been applied to deep reinforcement learning. Note that this review is intended for beginners in exploration for deep reinforcement learning; thus, the focus is on the breadth of approaches and their relatively simplified description. Note also that, throughout the paper, we will use 'reinforcement learning' as it is a more general term rather than 'deep reinforcement learning'.

Several review articles exist in the field of reinforcement learning. Aubert et al. [16] presented an overview of intrinsic motivation in reinforcement learning, Li [17] presented a comprehensive overview of techniques and applications, Nguyen et al. [18] considered an application to multi-agent problems, Levine [19] provided a tutorial and extensive comparison with probabilistic inference methods and [20] provided an extensive description of the key breakthrough methods in reinforcement learning, including ones in exploration. However, none of the aforementioned reviews focused on exploration or considered it in great detail. The only other review focused on exploration is from 1999 and is now outdated and inaccurate [21].

The contributions of this study are as follows. First, the systematic overview of exploration in deep reinforcement learning is presented. As mentioned above, no other modern review exists with this focus. Second, a categorisation of exploration in reinforcement learning is provided. The categorisation is devised to provide a good way of comparing different approaches. Finally, future challenges are identified and discussed.

2. Preliminaries

2.1. Introduction to reinforcement learning

2.1.1. Markov decision process

We consider a standard reinforcement setting in which an agent interacts with a stochastic and fully observable environment by sequentially choosing actions in a discrete time step to maximise cumulative rewards. This series of processes is called *Markov decision process* (MDP). An MDP has a tuple of (S, A, P, R, γ), where S is a set of states, A is a set of actions the agent can select, P is a transition probability that satisfies the Markov property given as:

$$p(s_{t+1}|s_1, a_1, s_2, a_2, \dots, s_t, a_t) = p(s_{t+1}|s_t, a_t), \tag{1}$$

R is a set of rewards, and $\gamma \in (0, 1]$ is a discount factor. At each time step t, an agent receives states $s_t \in S$ from the environment and selects the best possible actions $a_t \in A$ according to policy $\pi(a_t|s_t)$, which maps from states s_t to actions a_t. The agent receives a reward $r_t \in R$ from the environment to take an action a_t. The goal of the agent is to maximise the discounted expected reward $G_t = \sum_{k=0}^{\infty} \gamma^k r_{k+t}$ from each state s_t.

2.1.2. Value-based methods

Given that the agent follows policy π, a state-value function is defined as $V^\pi(s_t) = \mathbb{E}_\pi[G_t|S_t = s]$. Similarly, the action-value function, $Q^\pi(s_t, a_t) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$, is an expected estimate value for a given state s_t for taking an action a_t. Q-learning is a typical type of *off-policy learning* that updates a target policy π using samples generated by any stochastic behaviour policy in an environment. Following the *Bellman equation* and *temporal difference* (TD) for the action-value function, the Q-learning algorithm is recursively updated using the following equation:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a' \in A} Q(s_{t+1}, a') - Q(s_t, a_t)], \tag{2}$$

where a' follows the target policy $a' \sim \pi(\cdot|s_t)$ and α is the learning rate. While updating Q-learning, the next actions a_{t+1} are sampled from the behaviour policy which follows an ϵ-greedy exploration strategy, and among them, the action that makes the largest Q-value, a', is selected.

2.1.3. Policy-based methods

In contrast to value-based methods, policy-based methods directly update the policy parameterised by θ. In reinforcement learning, because the goal is to maximise the expected return throughout states, the objective function for the policy is defined as $J(\theta) = \mathbb{E}_{\pi_\theta}[G_t]$. Williams et al. [22] suggested the REINFORCE algorithm which updates the policy network by taking a gradient ascent in the direction of $\nabla_\theta J(\theta)$. The gradient of the objective function is expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) G_t], \tag{3}$$

where p^π denotes the state distribution. A general overview of reinforcement learning can be found in [23].

2.2. Exploration

Exploration can be defined as the activity of searching and finding out about something [24]. In the context of reinforcement learning, "something" is a reward function and the "searching and finding" is an agent's attempt to try to maximise the reward function. Exploration in reinforcement learning is of particular importance because a reward function is often complex and agents are expected to improve over their lifetime. Exploration can take various forms such as randomly taking certain actions and seeing the output, following the best known solution, or actively considering moves that are good for novel discoveries.

Problems that can be solved by exploration are common in nature. Exploration is the act of searching for a solution to a problem. We note that exploration is the most useful in problems in which a route to the actual solution (i.e. reward) is obstructed by the local minima (maxima) or areas of flat rewards. These conditions mean that discovering the true nature of rewards is challenging. The following examples are intuitive illustrations of those problems: (i) search and rescue—the agent needs to explore to find a target (victim); the agent is only rewarded when it finds the victim; otherwise, the reward is 0; and (ii) delivery—trying to deliver an object in the unknown areas; the agent is only rewarded when the appropriate drop-off point has been found; otherwise, the reward is 0. Exploration could be considered as a ubiquitous problem that is highly relevant to many domains with ongoing research.

2.3. Challenging problems

In this section, some of the challenging problems for exploration in reinforcement learning are described, namely noisy-TV and sparse reward problems.

2.3.1. Noisy-TV

In a noisy-TV [13] problem, the agent is stuck in exploring an infinite number of states which lead to no reward. This phenomenon can be easily explained with an example. Imagine a state consisting of a virtual TV where the agent can operate the remote, but operating the remote controller leads to no reward. A new random image is generated on the TV every time a remote is operated. Thus, the agent will experience novelty all the time. This keeps the agent's attention high infinitely but clearly leads to no meaningful progress. This kind of behaviour can also be described as a couch potato problem.

2.3.2. Sparse reward problems

Sparse rewards are a classical problem in exploration. In the sparse reward problem, the reward is relatively rare. In other words, there is a long gap between an action and a reward. This is problematic for reinforcement learning because for a long time (or at all times) it has no reward to learn from. The agent cannot learn any useful behaviours and eventually converges to a trivial solution. As an example, consider a maze where the agent has to complete numerous steps before reaching the end and being rewarded. The larger the maze is, the less likely it is for the agent to see the reward. Eventually, the maze will be so large that the agent will never see the reward; thus, it will have no opportunity to learn.

2.4. Benchmarks

In this section, the most commonly used benchmarks for reinforcement learning are briefly introduced and described. We highlight four benchmarks: Atari Games, VizDoom, Minecraft, and Mujoco.

2.4.1. Atari games

The Atari games benchmark are a set of 57 Atari games combined under the Atari Learning Environment (ALE) [25]. In Atari games, the state space is normally either images or random-access memory (RAM) snapshots. The action space consists of five joystick actions (up, down, left, right, and action button). Atari games can be largely split into two groups: easy (54 games) and difficult exploration (3 games) [26]. In the easy exploration problem, the reward is relatively easy to find. In hard exploration problems, the reward is not often given, and the association between states and rewards is complex.

2.4.2. VizDoom

VizDoom [27] is a benchmark based on the Doom game. The game has a first-person perspective (i.e., view from characters' eyes), and the image seen by the character is normally used as a state space. The action space is normally eight directional control and two action buttons (picking up key cards and opening doors). Note that more actions can be added, if needed. One of the key advantages of VizDoom is the availability of easy-to-use tools for editing scenarios and low computational burden.

2.4.3. Malmo

Malmo [28] is a benchmark based on the game Minecraft. In Minecraft, environments are built using same-shaped blocks, similar to how Lego bricks are used for building. Similar to VizDoom, it is also from the first-person perspective, and the image is the state space. The key advantage of Malmo is its flexibility in terms of the environment structure, domain size, custom scripts, and reward functions.

2.4.4. Mujoco

MuJoCo [29] represents multi-joint dynamics with contact. Mujoco is a popular benchmark used for physics-based simulations. In reinforcement learning, Mujoco is typically used to simulate walking robots. These are typically cheetah, ant, humanoids, and their derivatives. The task of reinforcement learning is to control various joint angles and forces to develop walking behaviour. Normally, the task is to walk as far as possible or to reach a specific goal.

3. Exploration in reinforcement learning

Exploration in reinforcement learning can be split into two main streams: efficiency and safe exploration. In efficiency, the idea is to make exploration more sample efficient so that the agent can explore in as few steps as possible. In safe exploration, the focus is on ensuring safety during exploration. We suggest splitting efficiency-based methods further into imitation-based and self-taught methods. In imitation-based learning, the agent learns how to utilise a policy from an expert to improve exploration. In self-taught methods, learning is performed from scratch. Self-taught methods can be further divided into planning, intrinsic rewards, and random methods. In planning methods, the agent plans its next action to gain a better understanding of the environment. In random methods, the agent does not make conscious plans; rather, it explores and then sees a consequence of this exploration. We distinguish intrinsic reward methods into two categories: (i) reward novel states–reward agents for visiting novel states; and (ii) reward diverse behaviours-reward agents for discovering novel behaviours. Note that intrinsic rewards are a part of a larger notion of intrinsic motivation. For an extensive review of intrinsic motivation, see [16,30]. In planning methods, two distinguished categories are considered: (i) goal-based: an agent is given an exploratory goal to reach; and (ii) probability- probabilistic models are used for an environment. Review of the entire categorisations is represented in Fig. 1. From the following, each category is described in detail. The main objective of the categorisation is to highlight the key contribution of each approach. Note that a certain approach could be a combination of various techniques. For example, Go-explore [31] utilises reward

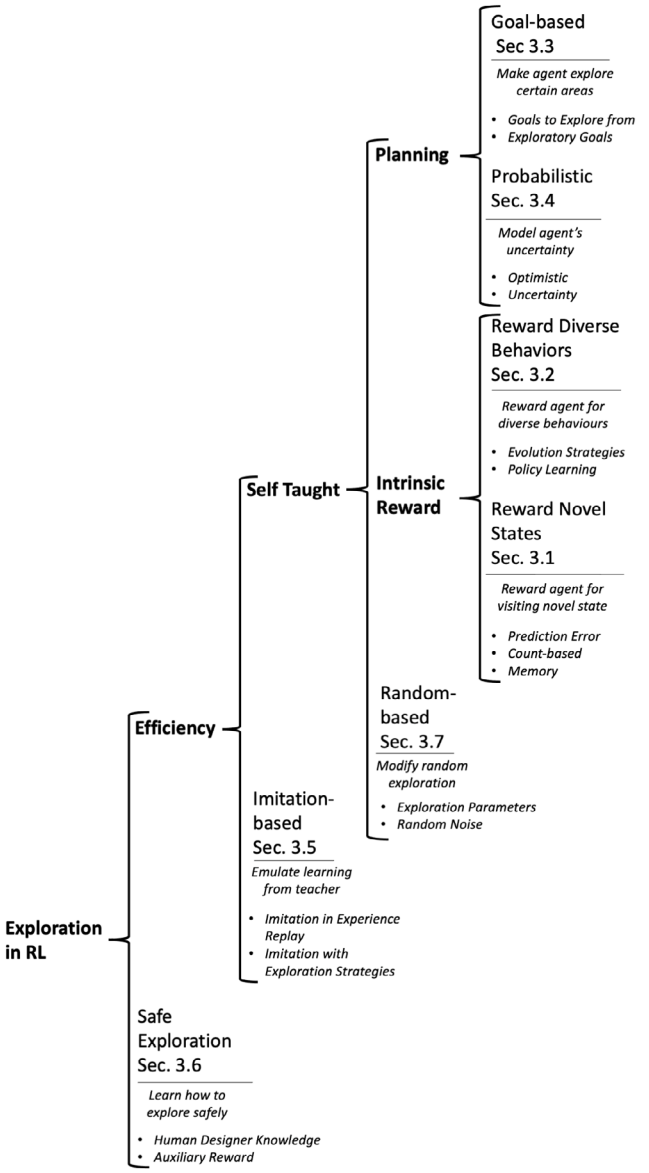

Fig. 1. Overview of exploration in reinforcement learning.

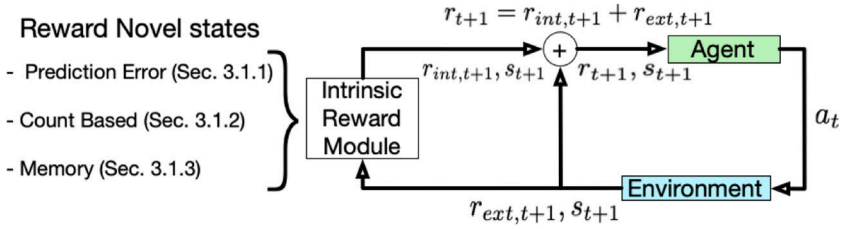

Fig. 2. Overview of the reward novel state methods. In general, in reward novel states, the agent is given additional reward r_{int} for discovering novelty. This additional reward is generated from intrinsic reward module r_{ext}.

novel states methods, but the main contribution is best described by goal-based methods.

3.1. Reward novel states

In this section, approaches on reward novel state are discussed and compared. Reward novel state approaches give agents a reward for discovering new states. This reward is called an intrinsic reward. As can be observed in Fig. 2, the intrinsic reward (r_{int}) supplements rewards given by the environment (r_{ext} called an extrinsic reward). By rewarding novel states, agents will incorporate exploration into their behaviours [30].

These approaches were generalised in [30]. In general, there are two necessary components: "an adaptive predictor or compressor or model of the growing data history as the agent is interacting with its environment to provide an intrinsic reward, and a general reinforcement learner to learn behaviours" [30]. In this division, the reinforcement learner is asked to invent things which predictor does not know yet. In our review, the former is simply referred to as an intrinsic reward module, and the latter is referred to as an agent.

There are different ways of classifying intrinsic rewards [16,32]. Here, we largely follow the classification of [16] with the following categories: (i) prediction error methods, (ii) count-based methods and (iii) memory methods.

3.1.1. Prediction error methods

In prediction error methods, the error of a prediction model when predicting a previously visited state is used to compute the intrinsic reward. For a certain state, if a model's prediction is inaccurate, it means that a given state has not been seen often and the intrinsic reward is high. One of the key questions that needs to be addressed is how to use the model's error to compute the intrinsic reward. To this end, Achiam et al. [33] compared two intrinsic reward functions: (i) how big the error is in a prediction model and (ii) the learning progress. The first method has shown better performance and is therefore recommended, which can be formalised as:

$$r_{int} = f(z(s_{t+1}) - M(z(s_t, a_t))) \tag{4}$$

where s represents a state, M is an environmental model, t and $t+1$ are two consecutive time steps, z is an optional model for state representation, and f is an optional reward scaling function.

The simplest method of this type was described in [10,11]. The intrinsic reward is measured as the Euclidean distance between the prediction of a state from a model and that state. This simple idea was revisited in [34]. Generative adversarial networks (GAN [35]), distinguishing real from fake states as a prediction error method, were proposed in [36]. Since then many other approaches were devised, which can be further divided into: (i) state representation prediction, (ii) a priori knowledge and (iii) uncertainty about the environment.

State representation prediction methods. In state representation prediction methods, the state is represented in a higher-dimensional space. Then, a model is tasked with predicting the next state representation given the previous state representation. The larger the error is in the prediction, the larger the intrinsic reward is. One way of providing state representation is using an autoencoder [37]. Both pre-trained and online trained autoencoders were considered and showed similar performance. Improvements to autoencoder-based approaches were proposed in [38,39], where a slow-trained autoencoder was added. Thus, the intrinsic reward decays slower and the agent explores for longer while increasing the chance of finding the optimal reward.

Another method of providing state representation involves utilising fixed networks with random weights. Then, another network is used to predict the outputs of randomly initialised networks as shown in Fig. 3. The most popular approach of this type is called random network distillation (RND) [13]. A similar approach was considered in [40].

A state representation method derived from inverse dynamic features (IDF) was used in [41]. In IDF, the representation comes from forcing an agent to predict the action as illustrated in Fig. 4. IDF was compared against the state prediction method and random representation in [42] with the following conclusions: IDF had the best performance and it scales the best to the unseen environments. IDF was utilised in [43], where the Euclidean distance between two consecutive state representations was used as an intrinsic reward, as shown in Fig. 4. Intuitively, the more significant the transition is, the larger the change is in IDF's state representation. In another study, RND and IDF were combined into a single intrinsic reward [44].

A compact representation using information theory was proposed in [45]. Information theory is used to represent states that are close to the representation space in the environment space. Information theory can also be used to create a bottleneck latent representation [46]. Bottleneck latent representation occurs when mutual information between the input to the network and latent representation is minimised.

A priori knowledge methods. In some types of problems, it makes sense to use certain parts of the state space as an error and use it for computing the intrinsic reward. Those parts could be depth point cloud, position, and sound, and they rely on a priori knowledge from the designer.

Depth point cloud prediction error was used in [47]. The scalability of this approach was analysed by [48]. It was found that the performance was good in the same environment with different starting positions, but it did not scale to a new scenario. Positions in a 3D space can also be used [49]. An approach using the position was proposed in [50]. The environment is split into the x–y grid where each node's intrinsic reward is placed. When the episode terminates, the rewards are restored to a default value.

Sound as a source of intrinsic reward was used in [51]. To model sounds, the model is trained to recognise when the sound and the corresponding frame match. If the model indicates misalignment between frames and sounds, it means that the state is novel.

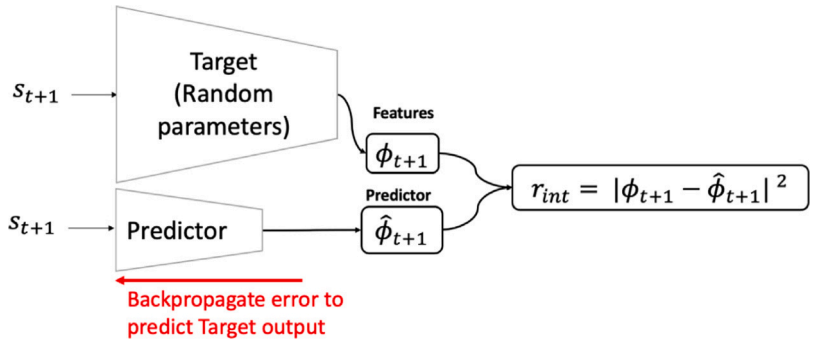

Fig. 3. RND overview. The predictor is trying to predict output of a randomly parameterised target.

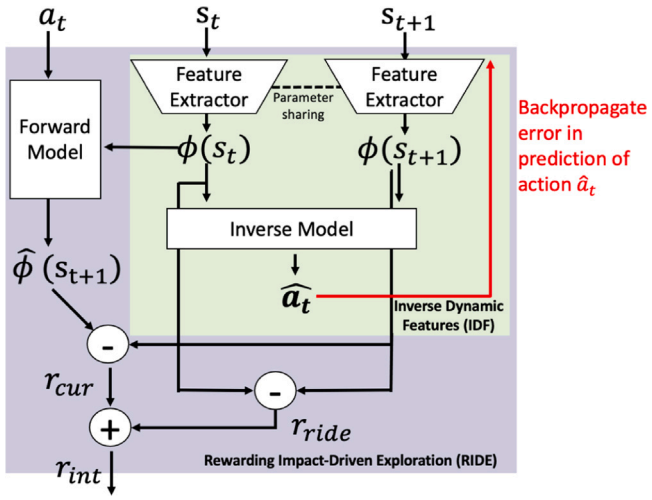

Fig. 4. IDF and rewarding impact driven exploration (RIDE) overview. In IDF, the features are extracted based on the network predicting the next action. In RIDE, the intrinsic reward is based on the difference in state representation (adapted from [43]).

Uncertainty about the environment methods. In these methods, the intrinsic reward is based on the uncertainty the agent has. If the agent is exploring highly uncertain areas of the environment, the reward is high. Uncertainty can be utilised using the following techniques: Bayesian, ensembles of models and information-theoretic approaches.

Bayesian approaches are generally intractable for large problem spaces; thus, approximations are used. Kotler et al. [52] presented a close to optimal approximation method using the Dirichlet probability distribution over state, action, and next state triplet. Another approximation could be to use ensembles of models as proposed in [53]. The intrinsic reward is given based on model disagreement as shown in Fig. 5. The models were initialised with different random weights and were trained on different mini-batches to maintain diversity.

In information-theoretic approaches, the intrinsic reward is computed using the information gained from agent actions. The higher the gain is, the more the agent learns, and the higher the intrinsic reward is. The general framework for these types of approaches was presented in [54,55]. One of the most popular information-theoretic approaches is variational information maximisation exploration (VIME) [56]. In this approach, the information gain is approximated as a Kullback–Leibler (KL) divergence between the weight distribution of the Bayesian neural network, before and after seeing new observations. In [57], maximising mutual information between a sequence of actions leads to a state that is rewarded. Rewarding this mutual information gain means maximising the information contained in the action sequence about a state. Mutual information gain was combined with the state prediction error into a single intrinsic reward in [58,59].

Discussion. The key advantages of prediction error methods are that they rely only on a model of the environment. Thus, there is no need for buffers or complex approximation methods. Each of the four different categories of methods has unique advantages and challenges.

While predicting the state directly requires little to no a priori knowledge, the model needs to learn how to recognise different states. Additionally, they struggle when many states are present in the environment. State representation methods can cope with large state spaces at the cost of increased designer burden and reduced accuracy. Moreover, in a state representation method, the agent cannot affect the state representation, which can often lead to different states being represented similarly. Utilising a priori knowledge relies on defining a special element of the state space as a source of an error for computing the intrinsic reward. These methods do not suffer from problems with the speed of prediction and state recognition. However, they rely on the designer experience to define parts of the state space appropriately. Finally, in uncertainty about the environmental approaches, the agent's uncertainty is used to generate the intrinsic reward. The key advantage of this approach is its high scalability and automatic transition between exploration and exploitation. Prediction error methods have also shown the ability to solve the couch-potato (noisy-TV) problem by storing observations in a memory buffer [60]. An intrinsic reward is given only when observation is sufficiently far away (in terms of time steps) from the observations stored in the buffer. This mitigates the couch potato problem since repeatedly visiting states close to each other is not rewarded.

3.1.2. Count-based methods

In count-based methods, each state is associated with the visitation count number $N(s)$. If the state has a low count, the agent will be given a high intrinsic reward to encourage revisiting. The method of computing the reward based on the count was discussed in [61]. It has been shown that $1/N(s)$ guarantees a faster convergence rate than the commonly used $1/\sqrt{N(s)}$.

In problems with large number of states, counting visits to states is difficult because it requires saving the count for each state. To solve this problem, count is normally done on a reduced-size state representation.

Count on state representation methods. In count on state representation methods, the states are represented in a reduced form to alleviate memory requirements. This allows storing the count and a state with minimal memory in a table, even in the case of a large state space.

One of the popular methods of this type was proposed in [62], where static hashing was used. Here, a technique called SimHash [63] was used, which represents images as a set of numbers called a hash. To generate an even more compact representation, in [64], the state was represented as the learned x–y position of an agent. This was achieved using an attentive dynamic model (ADM). Successor state representation (SSR) [65] is a method which combines count and representation. The SSR is based on the count and order between the states. Intuitively, the SSR can be used as a count replacement.

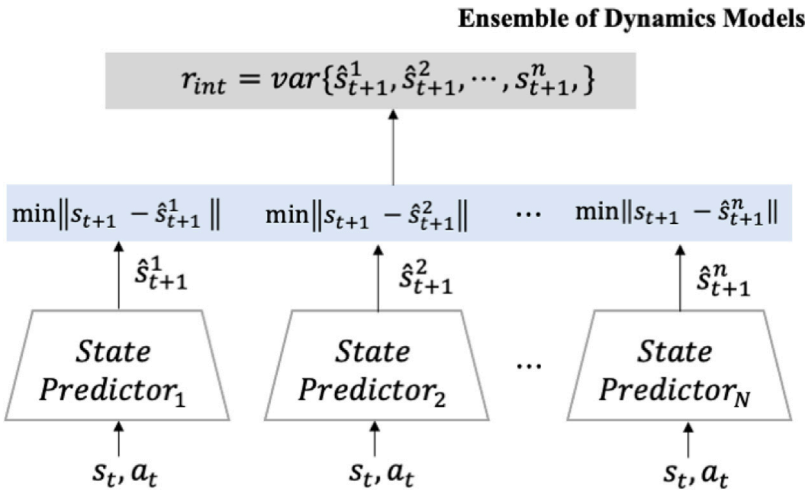

Fig. 5. Overview of self-supervised exploration via the disagreement method. The intrinsic reward is based on disagreement between models (adapted from [53]).

It is also possible to approximate count on state representation by using a function. For example, Bellemare et al. [14] proposed an approximation based on a density model. The density models include context tree switching (CTS) [14], Gaussian mixture models [66] or PixelCNN [67]. Martin et al. [68] proposed an improvement in the approximate count by making counts on the feature space rather than raw inputs.

Discussion. Count-based methods approximate the intrinsic reward by counting the number of times a given state has been visited. To reduce computational efforts of count-based methods, counts are generally associated with state representations rather than states. This, however, relies on being able to efficiently represent states. State representations can still require a lot of memory and careful design.

3.1.3. Memory methods

In these methods, an intrinsic reward is given on how easy it is to distinguish a state from all others. The easier it is to distinguish from the others, the more novel the given state is. As comparing states directly is computationally expensive, several approximation methods have been devised. Here, we categorise them into comparison models and experience replay.

Models can be trained for comparing state-to-state to reduce the computational load. One example method is to use exemplar model [69] developed in [70]. Exemplar models are a set of n classifiers, each of which is trained to distinguish a specific state from the others. Training multiple classifiers is generally computationally expensive. To further reduce the computational cost, the following two strategies are proposed: updating a single exemplar with a each new data point and sampling k exemplars from a buffer.

Instead of developing models for comparison, a limited size of experience replay was combined with prediction error methods in [71]. To devise intrinsic rewards, two rewards are combined: (i) intrinsic episodic experience replay is used to store states and compare them to others; and (ii) intrinsic motivation RND [13] is used to determine the state's long-term novelty. Additionally, multiple policies are trained, each with a different ratio between the extrinsic and intrinsic reward. A meta learner to automatically choose different ratios of extrinsic and intrinsic rewards at each step was proposed in [15].

Discussion. In memory-based approaches, the agent derives an intrinsic reward by comparing its current state with states stored in the memory. The comparison model method has the advantage of small memory requirements, but requires careful model parameter tuning. On the other hand, using a limited experience buffer does not suffer from model inaccuracies and has shown a great performance in difficult exploratory Atari games.

3.1.4. Summary

The reward novel state-based approaches are summarised in Table 1. The table utilises the following legend: Legend: A — action space, Ac — action, R — reference, MB — model based, MF — model free, D — discrete, C — continuous, Q — Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]). Prediction error methods are the most commonly-used methods. In general, they have shown very good performance (for example, RND [13] with 6500 in Montezuma's Revenge). However, they normally require a hand-designed state representation method for computational efficiency. This requires problem-specific adaptations, thus reducing the applicability of those approaches. Count-based methods are computationally efficient but they can either require memory to store counts or complex models [14]. Also, counting states in continuous-state domains is challenging and requires combining continuous states into discontinuous chunks. Recently, memory methods have shown good performance in games such as Montezuma Revenge, scoring as much as 11,000 [71]. Memory methods require a careful balance of how much data to remember for comparison. Otherwise, computing the comparison can take a long time.

3.2. Reward diverse behaviours

In reward diverse behaviours, the agent collects as many different experiences as possible, as shown in Fig. 6. This makes exploration an objective rather than a reward finding. These types of approaches can also be called diversity and can be split into evolution strategies and policy learning.

3.2.1. Evolution strategies

Reward diverse behaviours were initially used with an evolutionary-based approach. In evolutionary approaches, a group of sample solutions (population) is tested and evolves over time to get closer to the optimal solution. Note that evolutionary approaches are generally not considered as the part of reinforcement learning but can be used to solve the same type of problems [72,73].

One of the earliest methods called novelty search was devised in [74,75]. In novelty search, the agent is encouraged to generate numerous different behaviours using a metric called diversity measure. The diversity measure must be hand-designed for each environment, limiting transferability between different domains. Recently, novelty search has been combined with other approaches, such as reward maximisation [76] and reward novel state method [77]. In Conti et al. [76], the novelty-search policy is combined with a reward maximisation

Table 1
Comparison of reward novel state approaches.

R	Prior Knowledge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
Pathak et al. [41]		A3C	Prediction error	Vizdoom: very sparse 0.6 (A3C 0)	Vizdoom image	P	MB	D/ D
Stadie et al. [37]	autoencoder	DQN	Prediction error	Atari: Alien 1436 (DQN 300)	Atari images	Q	MB	D/ D
Savinov et al. [60]	pretrained discriminator (non-online)	PPO	Prediction error	Vizdoom: very sparse 1 (PPO 0); Dmlab: very sparse 30 (PPO+ICM 0)	Vizdoom images/ Mujoco joints angles	Ac	MB	C/ C
Burda et al. [13]		PPO	Prediction error	Atari: Montezuma Revenge 7500 (PPO 2500)	Atari images	P	MB	D/ D
Bougie and Ichise [38]		PPO	Prediction error	Atari: Montezuma Revenge 20 rooms found (RND 14)	images	Ac	MB	D/ D
Hong et al. [36]		DQN	Prediction error	Atari: Montezuma Revenge 200 (DQN 0)	Enumerated state id/ Atari Image	Q	MB	D/ D
Kim et al. [45]		TRPO	Prediction error	Atari: Frostbite 6000 (ICM 3000)	Atari Images	Ac	MB	D/ D
Stanton and Clune [50]	agent position, reward grid	A2C	Prediction error	Atari: Montezuma Revenge 3200 (A2C 0)	Atari images	Ac	MB	D/ D
Achiam and Sastry [33]		TRPO	Prediction error	Mujoco: halfcheetah 80 (VIME 40); Atari: Venture 400 (VIME 0)	Atari RAM states/ Mujoco joints angles	Ac	MB	C/ C
Li et al. [44]		A2C	Prediction error	Atari: Asterix 500000 (RND 10000)	Atari images	Ac	MB	D/ D
Kim et al. [46]		PPO	Prediction error	Atari: Montezuma Revenge with distraction 1500 (RND 0)	Atari images	Ac	MB	D/ D
Chien and Hsu [59]		DQN	Prediction error	PyDial: 85 (CME 80); OpenAI: Mario 0.8 (CME 0.8)	Images	Q	MB	D/ D
Li et al. [34]		DDPG	Prediction error	Robot: FetchPush 1 (DDPG 0)	Robot joints angles	Ac	MB	C/ C
Raileanu and Rocktäschel [43]		IMPALA	Prediction error	Vizdoom: 0.95 (ICM 0.95)	Vizdoom Images	Ac	MB	D/ D
Mirowski et al. [47]		A3C	Prediction error	DM Lab: Random Goal 96 (LSTM-A3C 65)	DM Lab images	Ac	MB	C/ C
Tang et al. [62]		TRPO	Count-based	Atari: Montezuma Revenge 238 (TRPO 0); Mujoco: swimmergather 0.3 (VIME 0.15)	Atari images/ Mujoco joints angles	P	MF	C/ C
Martin et al. [68]	Blob-PROST features	SARSA-e	Count-based	Atari: Montezuma Revenge 2000 (SARSA 200)	Blob-PROST features	Q	MB	D/ D
Machado et al. [65]		DQN	Count-based	Atari: Montezuma Revenge 1396 (Psuedo counts 1671)	Atari images	Q	MF	D/ D
Ostrovski et al. [67]		DQN and Reactor	Count-based	Atari: Gravitar 1500 (Reactor 1000)	Atari images	Ac	MB	D/ D
Badia et al. [71]		R2D2	Memory	Atari: Pitfal 15000 (R2D2 −0.5)	Atari images	P	MB	D/ D
Badia et al. [15]		R2D2	Memory	Beat humans in all 57 atari games	Atari images	P	MB	D/ D
Fu et al. [70]	state encoder	TRPO	Memory	Mujoco: SparseHalfCheetah 173.2 (VIME 98); Atari: Frostbite 4901 (TRPO 2869); Doom: MyWayHome 0.788 (VIME 0.443)	Atari images/ Mujoco joints angles	Ac	MB	C/ C

Legend: A — action space, Ac — action, R — reference, MB — model based, MF — model free, D — discrete, C — continuous, Q — Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]).

policy to encourage diverse behaviours and search for the reward. Gravina et al. [77] compared three ways of combining novelty search and reward novel state: (i) novelty search, (ii) sum of reward novel state and novelty search, and (iii) sequential optimisation where the

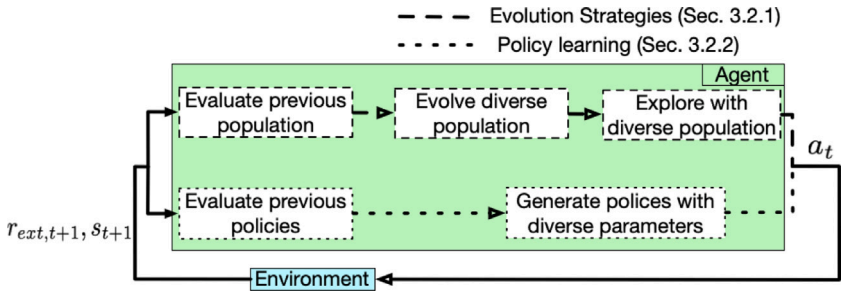

Fig. 6. Overview of reward diverse behaviour-based methods. The key idea is for the agent to experience as many things as possible, in which either evolution or policy learning can be used to generate a set of diverse experiences.

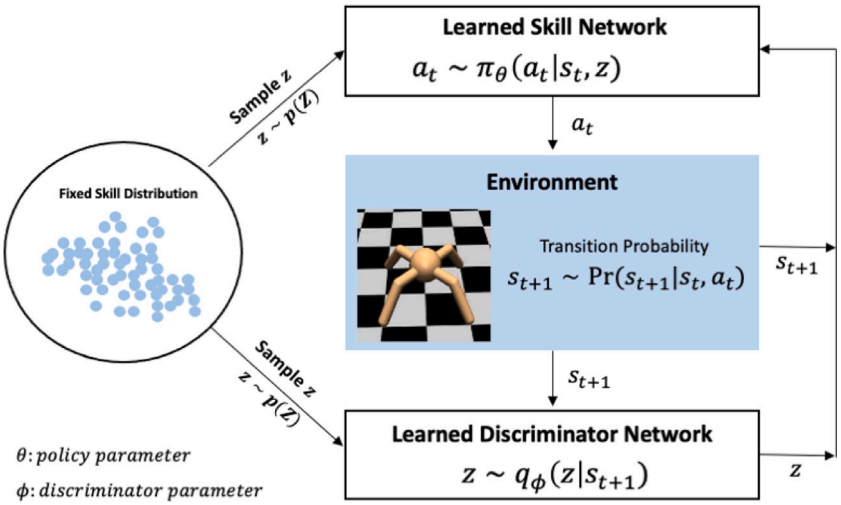

Fig. 7. An overview of Diversity is all you need (DIAYN), where the agent is encouraged to have as many diverse policies as possible (adapted from [12]).

second one performed the best in a simulated robot environment. More detailed reviews of exploration in evolution strategies can be found in [78,79].

Discussion. Initially, novelty search was used as a stand-alone technique; however, recently, combining it with other techniques [76,77] has shown more promise. Such a combination is more beneficial (in terms of reward) as diverse behaviours are more directed towards highly scoring ones.

3.2.2. Policy learning

Recently, diversity measures have been applied in policy learning approaches. The diversity among policies was measured in [80]. Diversity is computed by measuring the distance between policies (either KL divergence or simple mean squared error). Very promising results for diversity are presented in [12], as shown in Fig. 7. To generate diverse policies, the objective function consists of (i) maximising the entropy of skills, (ii) inferring behaviours from the current state, and (iii) maximising randomness within a skill. A similar approach was proposed in [81] with a new entropy-based objective function. A combination of diversity with a goal-based approach was proposed in [82]. In this study, the agent learns diverse goals and goals useful for rewards using the skew-fit algorithm. In the skew-fit algorithm, the agent skews the empirical goal distribution so that rarely visited states can be visited more frequently. The algorithm was tested using both simulations and real robots.

In [83], the agent stores a set of successful policies in an experience replay and then minimises the difference between the current policy and the best policies from storage. To allow exploration at the same time, the entropy of parameters between policies is maximised. The results show an advantage over evolution strategies and PPO in sparse reward Mujoco problems.

Discussion. Diversity in policy-based approaches is a relatively new concept that is still being developed. Careful design of a diversity criterion shows very promising performance, beating standard reinforcement learning with significant margins [12].

3.2.3. Summary

Reward diverse behaviour methods are summarised in Table 2. In evolution strategies approaches, a diverse population is used, whereas in policy learning, a diverse policy is found. Evolution strategies have the potential to find solutions that are not envisioned by designers as they search for the neural network structure as well as diversity. Evolution strategies suffer from the low sample efficiency, making the training either computationally expensive or slow. Policy learning is not able to go beyond pre-specified structures but can also show some remarkable results [12]. Another advantage of the policy learning method is suitability to both continuous and discrete state-space problems.

3.3. Goal-based methods

In goal-based methods, the states of interest for exploration are used to guide the agent's exploration. In this way, the exploration can immediately focus on largely unknown areas. In those types of methods, the agent requires a goal generator, a policy to find a goal, and an exploration strategy (see Fig. 8). The goal generator is responsible for creating goals for the agent. The policy is used to achieve the desired goals. An exploration strategy is used to explore once a goal has been achieved or while trying to achieve goals.

Here, we split goal-based methods into two categories: goals to explore from and exploratory goal methods.

Table 2
Comparison of reward diverse behaviour-based approaches.

R	Prior Knowledge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
Conti et al. [76]	domain specific behaviours	Reinforce	Evolution strategies	Atari: Frostbite 3785 (DQN 1000)	Atari RAM state/ Mujoco joints angles	Ac	MF	C/ C
Gravina et al. [77]		NS population based	Evolution strategies	Robotic navigation: 400 successes	six range finders, pie-slice goal-direction sensor	Ac	MB	C/ C
Lehman and Stanley [74]	measure of policies distance	NEAT	Evolution strategies	maze: 295 (maximum achievable)	six range finders, pie-slice goal-direction sensor	Ac	MF	D/ D
Risi et al. [75]	measure of policies distance	NEAT	Evolution strategies	T-maze: solved after 50,000 evaluations	enumerated state id	Ac	MF	D/ D
Cohen et al. [81]		SAC	Policy learning	Mujoco: Hopper 3155 (DIAYN 3120)	Mujoco joint angles	Ac	MB	C/ C
Pong et al. [82]		RIG	Policy learning	Door Opening (distance to the objective): 0.02 (RIG + DISCERN-g 0.04)	Robots joint angles	Ac	MB	C/ C
Eysenbach et al. [12]		SAC	Policy learning	Mujoco: half cheetah 4.5 (TRPO 2)	Mujoco joints angles	Ac	MF	C/ C
Hong et al. [80]		DQN, DDPG, A2C	Policy learning	Atari: Venture 900 (others 0); Mujoco: SparseHalfCheetah 80 (Noisy-DDPG 5)	Atari images/ Mujoco joints angles	Ac/ Q	MB	C/ C
Gangwani et al. [83]		Itself	Policy learning	Mujoco: SparseHalfCheetah 1000 (PPO 0)	Robot joints angles	Ac	MF	C/ C

Legend: A — action space, Ac — action, R — reference, MB — model based, MF — model free, D — discrete, C — continuous, Q — Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]).

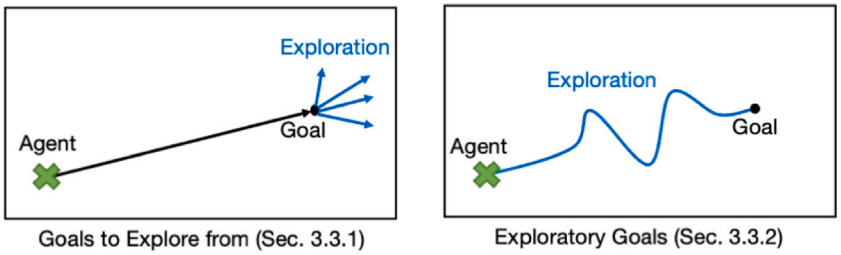

Fig. 8. Illustration of goal-based methods. In goal-based methods, the agent's task is to reach a specific goal (or state). Then, this goal is explored using another exploration method (left) or to generate an exploratory target goal (right). The key concept is to guide agents directly to unexplored areas.

3.3.1. Goals to explore from methods

The main technique used for these methods are (i) memorise visited states and trajectories — storing the past states in a buffer and choosing an exploratory goal from the buffer; and (ii) learn from the goal — assuming the goal state is known but a path to it is unknown.

One of the most famous approach when goal is chosen from a buffer of this type is called the go-explore [31]. The states and trajectories are saved in a buffer and are selected probabilistically. Once the state to explore from has been found, the agent is teleported there and explores it randomly. In [84], teleportation was replaced with policy learning. Go-exploration was extended to continuous domains in [85]. Concurrently, similar concepts were developed in [86–89]. In these approaches, a trajectory from the past is selected as an agent to exploit or explore randomly. If exploration is selected, a sample state from the trajectory is selected as a goal to explore based on the visitation count.

Another goal method was proposed in [90], where the least visited state was selected as a goal from the x–y grid on an Atari game. This reduces the computational effort of remembering where the agent has been significantly.

Learn from goal methods assume that the agent knows how the state with maximum reward looks like, but does not know how to get there. In this case, it is plausible to utilise this knowledge, as described in [91,92]. In [91], the model was trained to predict the backward steps in reinforcement learning. With such a model, the agent can 'imagine' states before the goal and thus can explore from the goal state. Similarly, another scenario, in which the agent can start at the reward position, can be conceived; then, it can also explore the starting position from the goal [92].

Discussion. Memorise visited states and trajectories methods have shown some remarkable results in hard exploration benchmarks such as Montezuma's revenge and pitfall. By utilising a reward state as a goal, as outlined in learn from the goal methods, the exploration problem can be mitigated, as the agent knows where to look for the reward.

3.3.2. Exploratory goal methods

In this subsection, an exploratory goal is given to the agent to try to reach. Exploration occurs when the agent attempts to reach the goal. The following techniques are considered: (i) meta-controllers, (ii) goals in the region of the highest uncertainty, and (iii) sub-goal methods.

Meta-controllers. In meta-controllers, the algorithm consists of two parts: a controller and a worker. The controller has a high-level overview and provides goals that the worker is trying to find.

One of the simple approaches is to generate and sample goals randomly [93]. The random goal selection mechanism was refined in [94] with goal selection based on the learning progress. A similar approach in two phases was proposed by Pere et al. [95]. First, the agent explores randomly to learn the environment representation. Second, the goals are randomly sampled from the learned representation. An approach in which both goal creation and selection mechanisms are devised by a meta-controller was proposed in [96]. In this work, a meta-controller proposes goals within a certain horizon for a worker to find.

In [97], a multi-arm bandit-based method to choose one strategy from a group of hand-designed strategies was proposed. At each episode, every ten steps, the agent chooses a strategy based on its performance. The goal selection mechanism from a group of hand-designed goals is also discussed in Kulkarni et al. [98]. The low-level controller is trained on a state–action space, and the meta-controller is trained on a goal-state space. An approach in which each subtask is learned by one learner was proposed in [99]. To allow any sub-task learner to perform its task from all states, the starting points for learning are shared between sub-task learners.

Sub-goals. In sub-goal methods, the algorithms find the sub-goals for the agent to reach. In general, sub-goal methods can be split into: (i) bottleneck states which lead to many others as exploratory goals, (ii) progress towards the main goal which is likely to lead to the reward and (iii) uncertainty based sub-goals.

One of the early methods of discovering bottleneck states was described in [100] using an ant colony optimisation method. Bottleneck states are said to be the states often visited by ants when exploring (by measuring pheromone levels). To discover bottleneck states, [101] proposed the use of proto-value functions based on the eigenvalue of representations. This allows the computation of eigenvector centrality [102], which has a high value if the node has many connections. This was later improved in [103] by replacing the handcrafted adjacency matrix with successor representations.

To design sub-goals which lead to a reward, Fang et al. [104] proposed progressively generating sub-tasks that are closer to the main task. To this end, two components are used: the learning progress estimator and task generator. The learning progress estimator determines the learning progress on the main task. The task generator then uses the learning progress to generate sub-tasks closer to the main tasks.

In uncertainty based methods, sub-goals the goals for the agent are positioned at the most uncertain states. One of the earliest attempts of this type was proposed by Guestrin et al. [105]. Here, the upper and lower bounds of the reward are estimated. Then, states with high uncertainty regarding the reward are used as exploratory goals. Clustering states using k-means and visiting least-visited clusters were proposed in [106]. Clustering can also help to solve the couch potato problem, as described in [107]. In this approach, the states are clustered using Gaussian mixture models. The agent avoids the couch potato problem by clustering all states from a TV into a single avoidable cluster.

Discussion. There are two main categories of exploratory goal methods: meta-controllers, and sub-goals. The key advantage of meta-controllers is that they allow the agent to set its own goals without excessively rewarding itself. However, training the controller is a challenge, which was not fully solved yet. In sub-goals methods, what constitutes a goal is defined by human designers. This puts a significant burden on the designer to provide suitable and meaningful goals.

3.3.3. Summary

The goal-based methods are summarised in Table 3. Goals to explore from methods have shown very good performance recently [84,89] in difficult exploratory games such as Montezuma's Revenge. The key challenges of these methods are the need to store states and trajectories as well as how to navigate to the goal. This issue is partially mitigated in [89] by using the agent's position as the state representation, however, this is highly problem-specific. Exploratory goal methods are limited as devising an exploratory goal becomes more challenging with increasing sparsity of the reward. This is somewhat mitigated in [94] or [104], but these approaches rely on the ability to parametrise the task.

3.4. Probabilistic methods

In probabilistic approaches, the agent holds a probability over states, actions, values, rewards or their combination and chooses the next action based on that probability. Probabilistic methods can be split into optimistic and uncertain methods [108]. The main difference between them is how they model a probability and how the agent utilises the probability, as shown in Fig. 9. In optimistic methods, the estimation needs to depend on a reward, either implicitly or explicitly. Then, the upper bound of the estimate is used to make the action. In uncertainty-based methods, the estimate is the uncertainty about the environment, such as the value function and state prediction. In the uncertainty-based method, the agent takes actions that minimise environmental uncertainty. Note that uncertainty methods can use estimations from optimistic methods but they utilise them differently.

3.4.1. Optimistic methods

In optimistic approaches, the agent follows *optimism under the uncertainty* principle. In other words, the agent follows the upper confidence bound of the reward estimate. The use of Gaussian process (GP) as a reward model was presented in [109]. The GP readily provides uncertainty, which can be used for reward estimation. The linear Gaussian algorithm can also be used as a model of the reward [110]. Bootstrapped deep-Q networks (DQN) and Thomson sampling were utilised in [111]. Bootstrapped DQNs naturally provide a distribution over rewards and values so that optimistic decisions can be taken.

It is also possible to hold a set of value functions and samples during exploration [112,113]. The most optimistic value function is used by the agent for an episode. At the end of the episode, the distribution of the value functions was updated.

Discussion. In optimistic approaches, the agent attempts to utilise *optimism under the uncertainty* principle. To utilise this principle the agent needs to be able to model the reward. It is possible to do this by either modelling the reward directly or by approximating value functions. Value function approximation can be advantageous as reward sparsity increases. With increased reward sparsity, the agent can utilise the partial reward from value functions for learning.

3.4.2. Uncertainty methods

In uncertainty-based methods, the agent holds a probability distribution over actions and/or states which represent the uncertainty of the environment. Then, it chooses an action that minimises the uncertainty. Here, five subcategories are distinguished: parameter uncertainty, value uncertainty, network ensemble, and information-theoretic.

Parameter uncertainty. In parameter uncertainty, the agent holds uncertainty over the parameters defining a policy. Then, the agent samples from those and follows this policy for a certain time and updates the parameters based on the performance. One of the simplest approaches is to hold a distribution over the parameters of the network [114]. Here, the network parameters were sampled from the weight distribution. Colas et al. [115] split the exploration into two phases: (i) explore randomly and (ii) compare experiences to an expert-created imitation to determine the good behaviour.

In [116], the successor state representation was utilised as a model of the environment. The exploration was performed by sampling parameters from the Bayesian linear regression model which predicts successor representation.

Table 3
Comparison of Goal-based approaches.

R	Prior Knowledge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
Guo et al. [87]		A2C and PPO	Goals to Explore from	Atari: Montezume Revenage 20158 (A2C+CoEX 6600)	Atari images	P	MF	C/ C
Guo and Brunskill [86]		DQN, DDPG	Goals to Explore from	Mujoco: Fetch Push 0.9 after 400 epoch (DDPG 0.5)	Mujoco joints angles	Q	MB	C/ C
Florensa et al. [92]	goal position	TRPO	Goals to Explore from	Mujoco: Key Insertion 0.55 (TRPO 0.01)	Mujoco joints angles	Ac	MF	C/ C
Edwards et al. [91]	goal state information	DDQN	Goals to Explore from	Gridworld 0 (DDQN −1)	Enumerated state id	Q	MF	D/ D
Matheron et al. [85]	state storage method	DDPG	Goals to Explore from	Maze: reach reward after 146,000 steps (TD3 never)	x–y position	Ac	MB	C/ C
Oh et al. [88]		A2C	Goals to Explore from	Atari: Montezuma Revenge 2500 (A2C 0)	Atari images	Ac	MF	D/ D
Guo et al. [89]	access to agent position	Itself	Goals to Explore from	Atari: Pitfall 11,000 (PPO 0); Robot manipulation task: 40 (PPO 0)	Atari images/ agent positions/ robotics joint angles	Ac	MF	C/ C
Ecoffet et al. [31]	teleportation ability	itself	Goals to Explore from	Atari: Montezuma Revenge 46000 (RND 11000)	Atari images	Ac	MB	D/ D
Ecoffet et al. [84]	access to agent position	itself	Goals to Explore from	Atari: Montezuma Revenge 46000 (RND 11000)	Atari images	Ac	MB	D/ D
Hester and Stone [97]	Strategies set	texpl-ore-vanir	Exploratory Goal	Sensor Goal: −53 (greedy −54)	Enumerated state id	Ac	MB	D/ D
Machado et al. [101]	hadcrafted features	itself	Exploratory Goal	4-room domain: 1	Enumerated state id	Ac	MB	D/ D
Machado et al. [103]		itself	Exploratory Goal	4-room domain: 1	Enumerated state id	Ac	MB	D/ D
Abel et al. [106]		DQN	Exploratory Goal	Malmo: Visual Hill Climbing 170 (DQN+boosted 60)	Image/ Vehicle positions	Q	MB	C/ C
Forestier et al. [93]	randomly generated goals	Itself	Exploratory Goal	Minecraft: mountain car 84% explored (ϵ-greedy 3%)	State Id	Ac	MB	C/ C
Colas et al. [94]		M-UVFA	Exploratory Goal	OpenAI: Goal Fetch Arm 0.8 (M-UVFA 0.78)	Robot joints angles	Ac	MB	C/ C
Péré et al. [95]		IMGEP	Exploratory Goal	Mujoco (KLC): ArmArrow 7.4 (IMGEP with handcrafted features 7.7)	Mujoco joints angles	Ac	MF	C/ C
Ghafoorian et al. [100]		Q-learning	Exploratory Goal	Taxi Driver: Found goal after 50 episodes (Q-learning after 200)	State Id	Q	MF	D/ D
Riedmiller et al. [99]	rewards for auxiliary tasks	Itself	Exploratory Goal	Block stacking: 140 (DDPG 0)	Robot joints angles	Ac	MB	C/ C
Fang et al. [104]	tasks parameterisation	itself	Exploratory Goal	GirdWorld: 1 (GoalGAN 0.6)	Robot joints angles, images	Ac	MB	C/ C

Legend: A — action space, Ac — action, R - reference, MB — model based, MF — model free, D — discrete, C — continuous, Q - Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]).

Policy and Q-value uncertainty. In policy and Q-value uncertainty, the agent holds uncertainty over Q-values/actions and samples the appropriate action. Some of the simplest approaches rely on optimisation to determine the distribution parameters. For example, in [117], the cross-entropy method (CEM) was used to control the variance of a Gaussian distribution from which actions were drawn. Alternatively, policies can be sampled [118]. In this study, a set of sampling policies sampled from a base policy were used. At the end of the episode, the best policy was chosen as an update to the base policy.

The most prevalent approach of this type is to use the Bayesian framework. In [119], the hypothesis is generated once and then followed for a certain number of steps, which saves computational time. This idea was further developed in [120], where Bayesian sampling was combined with a tree-based state representation for further efficiency gains. To enable Bayesian uncertainty approaches to deep learning, O'Donoghue et al. [121] derived Bayesian uncertainty such that it can be computed using the Bellman principle and the output of the neural network.

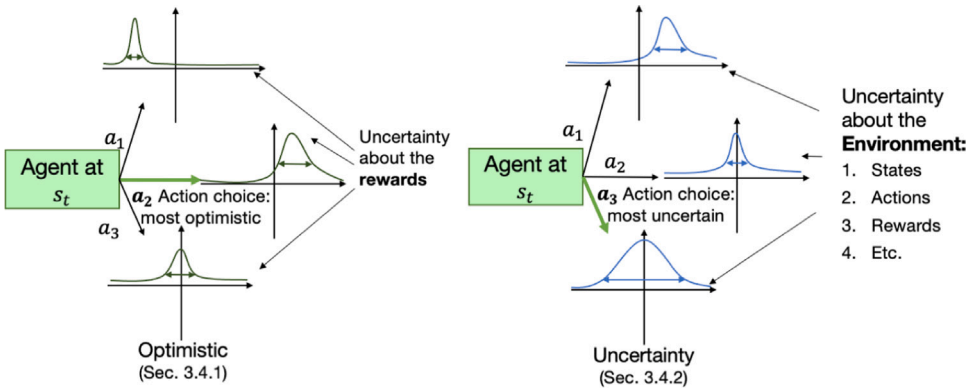

Fig. 9. Overview of probabilistic methods. The agent uses uncertainty over the environment model to either behave optimistically (left) or follow the most uncertain solution (right). Both should lead to a reduction in the uncertainty of the agent.

To minimise the uncertainty about policy and/or Q-values, information-theoretic approaches can be used. Agents choose actions that will result in maximal information gain, thus reducing uncertainty about the environment. An example of this approach, called information-directed sampling (IDS), is discussed in [122]. In IDS, the information gain function is expressed as a ratio between regret and how informative the action is.

Network ensembles. In the network ensemble method, the agent uses several models (initialised with different parameters) to approximate the distribution. Sampling one model from the ensemble to follow was discussed in [123]. In this study, a DQN with multiple heads, each estimating Q-value, was proposed. At each episode, one head was chosen randomly for use.

It is difficult to determine the model convergence by sampling one model at a time. Therefore, multiple models to approximate the distribution over states were devised in [124]. In this approach, Q-values estimated by different models were computed and fitted into a Gaussian distribution. A similar approach was developed in [125], using the information gain among the environmental models to decide where to go. Another ensemble model was presented in [126]. Exploration is achieved by finding a policy which results in the highest disagreement among the environmental models.

Discussion. In parameter sampling, the policy is parameterised (i.e. represented by the neural network), and the probability over parameters is devised. The agent samples the parameters and continues the update-exploitation cycle. In contrast, in policy and Q-value sampling methods, the probability distribution is not based on policy parameters but on actions and Q-values. The advantage of doing this over parameter sampling is faster updates because the policy can be adjusted dynamically. The disadvantage is that estimating the exact probability is intractable, and thus, simplifications need to be made. Another method is to use network ensembles to approximate the distribution over the action/states. This agent can either sample from the distribution or choose one model to follow. While more computationally intensive, this approach can also be updated instantaneously.

3.4.3. Summary

Tabular summary of optimistic and uncertainty approaches is shown in Table 4 and have been extensively compared in [108]. The article concludes that the biggest issue for optimistic exploration is that the confidence sets are built independent of each other. Thus, an agent can have multiple states with high confidence. This results in unnecessary exploration as the agent visits states which do not lead to the reward. Remedying this issue would be computationally intractable. In uncertainty methods, the confidence bounds are built depending on each other; thus, it does not have this problem.

3.5. Imitation-based methods

In imitation learning, the exploration is 'kick-started' with demonstrations from different sources (usually humans). This is similar to how humans learn because we are initially guided in what to do by society and teachers. Thus, it is plausible to see imitation learning as a supplement to standard reinforcement learning. Note that demonstrations do not have to be perfect; rather, they just need to be a good starting point. Imitation learning can be categorised to imitation in experience replay and imitation with exploration strategy as illustrated in Fig. 10.

3.5.1. Imitations in experience replay methods

One of the most common techniques is combining samples from demonstrations with samples collected by an agent in a single experience replay. This guarantees that imitations can be used throughout the learning process while using new experiences.

In [127], the demonstrations were stored in a prioritised experience replay alongside the agent's experience. The transitions from demonstrations have a higher probability of being selected. Deep Q learning from demonstration (DQfD) [128] differs in two aspects from [127]. First, the agent was pre-trained on demonstrations only. Second, the ratio between the samples from the agent's run and demonstrations was controlled by a parameter. A similar work with R2D2 was reported in [129]. Storing states in two different replays was presented in [130]. Every time the agent samples for learning, it samples a certain amount from each buffer.

Discussion. Using one or two experience replays seems to have negligible impact on performance. However, storing in one experience replay is conceptually and implementation-wise easier. Moreover, it allows agents to stop using imitation experiences when they are not needed anymore.

3.5.2. Imitation with exploration strategy methods

Instead of using experience replays, imitations and exploration strategies can be combined directly. In such an approach, imitations are used as a 'kick-start' for exploration.

A single demonstration was used as a starting point for exploration in [131]. The agent randomly explores from a state alongside a single demonstration run. The agent trained from a mediocre demonstration can score highly in Montezuma's Revenge. The auxiliary reward approach was proposed in Aytar et al. [26]. The architecture can combine several YouTube videos into a single embedding space for training. The auxiliary reward is added to every N frame from the demonstration video. The agent that can ask for help from the demonstrator was proposed in [132]. If the agent detects an unknown environment, the human demonstrator is asked to show the agent how to navigate.

Table 4
Comparison of probabilistic approaches.

R	Prior Knowledge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
D'Eramo et al. [111]		bDQN, SARSA	Optimistic	Mujoco: acrobot −100 (Thomson −120)	Mujoco joints angles	Q	MF	C/ C
Osband et al. [112]		LSVI	Optimistic	Tetris: 5000 (LSVI 4000)	Hand tuned 22 features	Ac	MF	D/ D
Jung and Stone [109]			Optimistic	Mujoco: Inverted Pendulum 0 (SARSA −10)	State Id	Ac	MB	D/ C
Xie et al. [110]		MPC	Optimistic	Robotics hand simulation: complete each of 10 poses	joints angles	Ac	MB	C/ C
Osband et al. [113]		LSVI	Optimistic	Cartpole Swing up: 600 (DQN 0)	State Id	Ac	MF	D/ D
Nikolov et al. [122]		bDQN and C51	Uncertainty	55 atari games: 1058% of reference human performance	Atari images	Q	MB	D/ D
Colas et al. [115]	a set of goal policies O	DDPG	Uncertainty	Mujoco: Half Cheetah 6000 (DDP 5445)	Mujoco joints angles	P	MF	C/ C
Osband et al. [123]		DQN	Uncertainty	Atari: James Bond 1000 (DQN 600)	Atari images	Q	MB	D/ D
Tang and Agrawal [114]		DDPG	Uncertainty	Mujoco: sparse mountaincar 0.2 (NoisyNet 0)	Mujoco joints angles	Ac	MF	D/ C
Strens [119]		Dynamic Program-ming	Uncertainty	Maze: 1864 (QL SEMI-UNIFORM 1147)	Enumerated state id	Ac	MB	D/ D
Akiyama et al. [118]	initial policy guess	LSPI	Uncertainty	Ball bating 2-DoF simulation: 67 (Passive learning:61)	Robot angles	Ac	MB	D/ C
Henaff [126]		DQN	Uncertainty	Maze: −4 (UE2 −14)	Enumerated state id	Q	MB	D/ D
Guez et al. [120]	guess of a prior	Policy learning	Uncertainty	Dearden Maze: 965.2 (SBOSS 671.3)	Enumerated state id	Ac	MB	D/ D
O'Donoghue et al. [121]	prior distribution	DQN	Uncertainty	Atari: Montezuma Revenge 3000 (DQN 0)	Atari images	Q	MB	D/ D
Shyam et al. [125]		SAC	Uncertainty	Chain: 100% explored (bootstrapped-DQN 30%)	Enumerated state id/ Mujoco joints angles	Ac	MB	C/ C
Stulp [117]		PI^2	Uncertainty	Ball batting: learned after 20 steps	Robot joints angles	Ac	MB	C/ C
Janz et al. [116]		DQN	Uncertainty	49 Atari games: 77.55% superhuman (Bootstrapped DQN 67.35%)	Atari images	Q	MB	D/ D

Legend: A — action space, Ac — action, R — reference, MB — model based, MF — model free, D — discrete, C — continuous, Q — Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]).

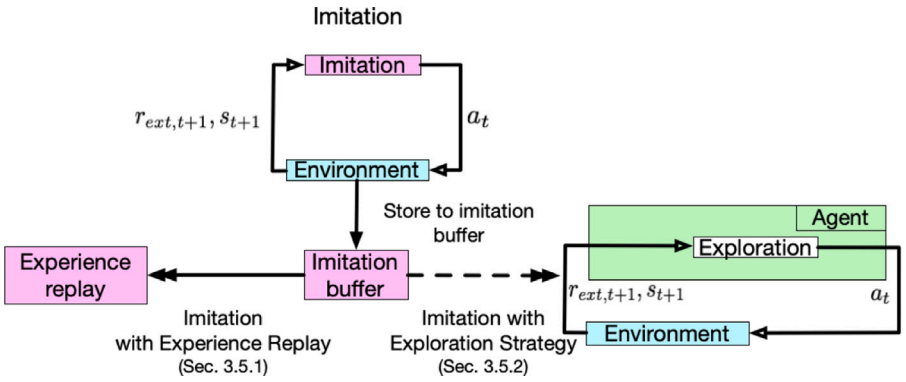

Fig. 10. Overview of imitation-based methods. In imitation-based methods, the agent receives demonstrations from expert on how to behave. These are then used in two ways: (i) directly learning on demonstrations or (ii) using demonstrations as a start for other exploration techniques.

Table 5
Comparison of imitation-based approaches.

R	Prior Knowledge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
Hester et al. [128]	imitation trained policy	DQN	Imitations in Experience Replay	Atari: Pitfall 50.8 (Baseline 0)	Atari Images	Q	MF	D/ D
Vecerik et al. [127]	demonstrations	DDPG	Imitations in Experience Replay	Peg insertion: 5 (DDPG −15)	Robot joints angles	Ac	MF	C/ C
Nair et al. [130]	demonstrations	DDPG	Imitations in Experience Replay	Brick stacking: Pick and Place 0.9 (Behaviour cloning 0.8)	Robot joints angles	Ac	MF	C/ C
Gulcehr et al. [129]	demonstrations	R2D2	Imitations in Experience Replay	Hard-eight: Drawbridge 12.5 (R2D2:0)	Vizdoom Images	Ac	MF	D/ D
Aytar et al. [26]	youtube embbeding	IMPALA	Imitation with Exploration Strategy	Atari: Montezuma's Revenge 80k (DqfD 4k)	Atari Images	Ac	MF	D/ D
Salimans and Chen [131]	single demonstration	PPO	Imitation with Exploration Strategy	Atari: Montezuma Revenge with distraction 74500 (Playing by youtube 41098)	Atari images	Ac	MF	D/ D

Legend: A — action space, Ac — action, R — reference, MB — model based, MF — model free, D — discrete, C — continuous, Q — Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]).

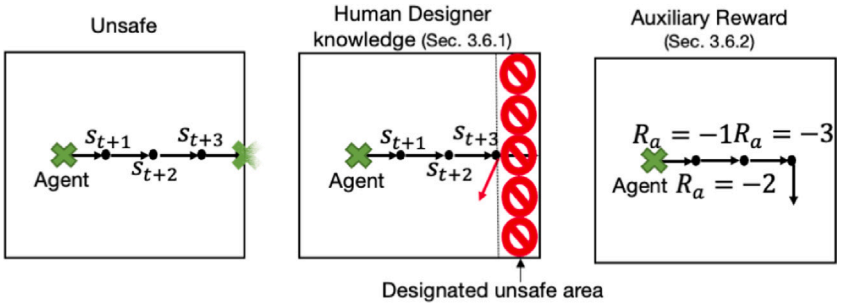

Fig. 11. Illustration of safe exploration methods. In safe exploration methods, attempts are made to prevent unsafe behaviours during exploration. Here, three techniques are highlighted: (i) human designer knowledge—the agent's behaviours are limited by human-designed boundaries; (ii) prediction models—the agent learns unsafe behaviours and how to avoid them; and (iii) auxiliary rewards—agents are punished in dangerous states.

Discussion. Using imitations as a starting point for exploration has shown impressive performance in difficult exploratory games. In particular, [26,131] scored highly in Montezuma's Revenge. This is the effect of overcoming the initial burden of exploration through demonstrations. The approach from [26] can score highly in Montezuma's revenge with just a single demonstration, making it very sample efficient. Meanwhile, the approach from [26] can combine data from multiple sources, making it more suitable for problems with many demonstrations.

3.5.3. Summary

A comparison of the imitation methods is presented in Table 5. Imitations in experience replay allow the agent to seamlessly and continuously learn from demonstration experiences. However, imitations with exploration strategies have the potential to find good novel strategies around existing ones. Imitations with exploration strategies have shown a great capability to overcome initial exploration difficulty. Imitations with exploration strategies achieve better performance compared with using imitations in experience replay only.

3.6. Safe exploration

In safe exploration, the problem of preventing agents from unsafe behaviours is considered. This is an important aspect of exploration research, as the agent's safety needs to be ensured. Safe exploration can be split into three categories: (i) human designer knowledge, (ii) prediction model, and (iii) auxiliary reward as illustrated in Fig. 11. For more details about safe exploration in reinforcement learning, the reader is invited to read [133].

3.6.1. Human designer knowledge methods

Human-designated safety boundaries are used in human designer knowledge methods. Knowledge from the human designer can be split into baseline behaviours, direct human intervention and prediction models.

Baseline behaviours impose an impassable safety baseline. Garcia et al. [134] proposed the addition of a risk function (which determines unsafe states) and baseline behaviour (which decides what to do in unsafe states). In [135], the agent was constrained by an additional pre-trained module to prevent unsafe actions as shown in Fig. 12, while in [136], agents are expected to perform no worse than the a priori known baseline. Classifying which object is dangerous and how to avoid them before the training of an agent was proposed in [137]. The agent learns how to avoid certain objects rather than states; thus, this approach can be generalised to new scenarios.

The human intervention approach was discussed in [138]. During the initial phases of exploration, humans in the loop stop disasters. Then, a supervised trained network of data collected from humans is used as a replacement for humans.

In the prediction model, the human designed safety model determines if the agent's next action leads to an unsafe position and

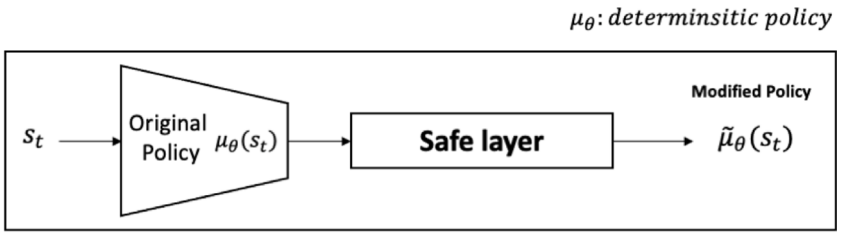

Fig. 12. Overview of safe exploration in continuous action spaces [135]. The additional model is modifying the actions of the original policy.

avoids it. In [139], a rover traversing a terrain of different heights was considered. The Gaussian process model provides estimates of the height at a given location. If the height is lower than the safe behaviour limit, the robot can explore safely. A heuristic safety model using a priori knowledge was proposed in [140]. To this end, they proposed an algorithm called action pruning, which uses the heuristics to prevent agent from committing to unsafe actions.

Discussion. In human designer knowledge methods, the barriers to unsafe behaviours are placed by a human designer. Baseline behaviours and human intervention methods guarantee certain performance in certain situations but they will only work in pre-defined situations. Prediction model methods require a model of the environment. This can be either in the form of a mathematical model [139] or heuristic rules [140]. Prediction models have a higher chance of working on previously unseen environments and have a higher chance of adaptability than baseline behaviours and human intervention methods.

3.6.2. Auxiliary reward methods

In auxiliary rewards, the agent is punished for putting itself into a dangerous situation. This approach requires the least human intervention, but it generates the weakest safety behaviours.

One of the methods is to find states in which an episode terminates and discourages an agent from approaching using an intrinsic fear [141]. The approach counts back a certain number of states from death and applies the distance-to-death penalty. Additionally, they made a simple environment in which a highest positive reward was next to the negative reward. The DQN eventually jumps to the negative rewards. The authors state "We might critically ask, in what real-world scenario, we could depend upon a system [DQN] that cannot solve [these kinds of problems]". A similar approach, but with more stochasticity, was later proposed in [142].

Allowing the agent to learn undesirable states from previous experiences autonomously was discussed in [143]. The states and their advantage values were stored in a common buffer. Then, frequently visited states with the lowest advantage have additional negative rewards associated with them.

Discussion. Auxiliary rewards can be an effective method of discouraging agents from unsafe behaviours. For example, in [141], half of the agent's death was prevented. Moreover, some approaches, such as Karimpanal et al. [143], have shown the ability to fully automatically determine undesirable states and avoid them. This, however, assumes that when the agent perishes, it has a low score; this may not always be the case.

3.6.3. Summary

An overview of the safety approaches is shown in Table 6. Safety is a vital aspect of reinforcement learning for practical applications in many domains. There are three general approaches: human designer knowledge, prediction models, and auxiliary rewards. Human designer knowledge guarantees safe behaviour in certain states. However, the agent struggles to learn new safe behaviours. Auxiliary reward approaches can adjust to new scenarios, but they require time to train and design of the negative reward.

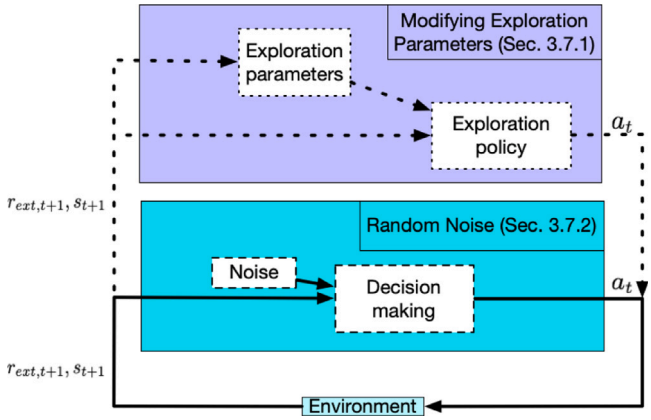

Fig. 13. Overview of random based methods. In random-based methods, simple random exploration is modified for improved efficiency. In modifying the states for exploration, the number of actions to be taken randomly is reduced. In modifying the exploration parameters, the exploration is automatically decided. In the network parameter noise, the noise is imposed on the policy parameters.

3.7. Random-based methods

In random-based approaches, improvements to simple random exploration are discussed. Random exploration tends to be inefficient as it often revisits the same states. To solve this problem, the following approaches are considered: (i) reduced states/actions for exploration methods, (ii) exploration parameters methods, and (iii) network parameter noise methods, as illustrated in Fig. 13.

3.7.1. Exploration parameters methods

In this section, exploration is parameterised (for example, ϵ in ϵ-greedy). Then, the parameters are modified according to the agent's learning progress.

One technique to adapt the exploration rate is by simply considering a reward and adjusting the random exploration parameter accordingly, as described in [144]. Using a pure reward can lead to problems with sparse rewards. To solve this problem, in [145], ϵ was made to depend on the error of the value-function estimates instead of the reward. It is also possible to determine the amount of random exploration using the environmental model entropy, as discussed in [146]. The learning rate [147] can also depend on exploration in which a parameter α that is functionally equivalent to the learning rate is introduced. If the agent is exploring a lot, the value of α slows down the learning to account for uncertainty. Khamassi et al. [148] used long-term and short-term reward averages to control exploration and exploitation. When the short-term average is below the long-term average, exploration should be increased.

Chang et al. [149] used multiple agents (ants) to adjust the exploration parameters. At each step, the ants chose their actions randomly, but were skewed by pheromone values left by other ants.

Another approach of this type could be reducing states for exploration based on some predefined metric. An approach using the

Table 6
Comparison of Safe approaches.

R	Prior Knowledge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
Garcelon et al. [136]	Baseline safe policy	Policy-based UCRL2	Human Designer Knowledge	stochastic inventory control: never breaching the safety baseline	amount of products in inventory	P	MB	D/ D
Garcia and Fernandez [134]	baseline behaviour		Human Designer Knowledge	car parking problem 6.5	angles and positions of respective controllable vehicles	Ac	MB	D/ D
Hunt et al. [137]	pretrained safety network	PPO	Human Designer Knowledge	Point mass environment: 0 unsafe actions (PPO 3000)	bird's eye view of the problem	Ac	MB	D/ D
Saunders et al. [138]	human intervention data	DQN	Human Designer Knowledge	Atari: Space Invaders 0 catastrophes (DQN 800000)	Atari images	Q	MB	D/ D
Dalal et al. [135]	pretrained safety model	DDPG	Human Designer Knowledge	spaceship: Arena 1000 (DDPG 300)	x–y position	Ac	MB	C/ C
Gao et al. [140]	environmental knowledge	PPO	Human Designer Knowledge	Pommerman: 0.8 (Baseline 0)	Agent, enemy agents and bombs positions	Ac	MB	D/ D
Turchetta et al. [139]		Bayesian optimisation	Human Designer Knowledge	Simulated rover: 80% exploration (Random 0.98%)	x–y position	Ac	MB	C/ C
Fatemi et al. [142]		DQN	Auxiliary Reward	Bridge: optimal after 14k episodes (ten times faster then competitor)	card types/ atari images	Q	MB	D/ D
Lipton et al. [141]		DQN	Auxiliary Reward	Atari: Asteroids total death 40,000 (DQN 80,000)	Atari images	Ac	MB	C/ C
Karimpanal et al. [143]		Q-learning and DDPG	Auxiliary Reward	Navigation environment: −3 (PQRL −3.5)	Enumerated state id	Q	MF	C/ C

Legend: A — action space, Ac — action, R — reference, MB — model based, MF — model free, D — discrete, C — continuous, Q — Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]).

adaptive resonance theorem (ART) [150] was presented in [151] and was later extended in [152]. In ART, knowledge about actions can be split into: (i) positive chunk which leads to positive rewards, (ii) negative chunk which leads to negative results, and (iii) empty chunk which is not yet taken. In this approach, the action is randomly chosen from positive and no chunks; thus, the agent is exploring either new things or ones with the positive reward. Wang et al. [152] extended this to include the probability of selecting the remaining actions based on how well they are known.

Discussion. Different parameters can be changed based on learning progress. Initially, approaches used learning progress, reward, or value of states to determine the rate of exploration. The challenge with these approaches is determining the parameters controlling the exploration. However, it is also possible to adjust the learning rate based on exploration [147]. The advantage is that the agent avoids learning uncertain information, but it slows down the training. Finally, reducing states for exploration can make exploration more sample efficient, but it struggles to account for unseen states that occurs after the eliminated states.

3.7.2. Random noise

In random noise approaches, random noise is used for exploration. The random noise can be either imposed on networks parameters or be produced based on states met during exploration.

The easiest method of including the noise is to include a fixed amount of noise [153]. This paper reviews the usage of small perturbations in the parameter space. In [154], chaotic networks were used to induce the noise in the network. It is also possible to adjust the noise strength using backpropagation, as described in [155] where the noise is created by a constant noise source multiplied by a gradient-adaptable parameter. Another way of the using the noise is by comparing the decision made by the noisy and noiseless policy [156]. Exploration is imposed, if decisions are sufficiently different.

In [157], the problem of assigning rewards when the same state is present multiple times is discussed. In such a problem, the agent will be likely to take different actions for the same state, making credit assignment difficult. To solve this problem, a random action generation function dependent on the input state was developed; if the state is the same, the random action is the same.

Discussion. Network parameter noise was first developed for evolutionary approaches, such as [153]. Recently, the noise of parameters has been used in policy-based methods. In particular, good performance was achieved in [155] which was able to achieve 50% improvement averaged over 52 Atari games.

3.7.3. Summary

A comparison of the random-based approach is presented in Table 7. The key advantage of reduced states for exploration methods is that the exploration can be very effective, but it needs to hold the memory of where it has been. Exploration parameter methods solves a trade-off between exploration and exploitation well; however, the agent can still get stuck in exploring unnecessary states. The random noise approaches are very simple to implement and show promising results, but they rely on careful tuning of parameters by designers.

Table 7
Comparison of Random based approaches.

R	Prior Knowledge	U	Method	Top score on a key benchmark	Input Types	O	MB/ MF	A/ S
Wang et al. [152]		ART	Exploration parameters	minefield navigation (successful rate): 91% (Baseline 91%)	Vehicles positions	Q	MB	C/ C
Shani et al. [147]		DDQN and DDPG	Exploration parameters	Atari: Frostbite 2686 (DDPG 1720); Mujoco: HalfCheetah 4579 (DDPG 2255)	Atari images, Mujoco joints angles	Ac/ Q	MF	C/ C
Patrascu and Stacey [144]		Fuzzy ART MAP architecture	Exploration parameters	Changing world environment (grid with two alternating paths to reward) 0.9	Enumerated state id	Ac	MB	D/ D
Usama and Chang [146]		DQN	Exploration parameters	VizDoom: Defend the centre 12.2 (ϵ-greedy 11.8)	Images	Q	MB	C/ C
Tokic [145]		V-learning	Exploration parameters	Multi-arm bandit: 1.42 (Softmax 1.38)	Enumerated state id	V	MF	D/ D
Khamassi et al. [148]		Q-learning	Exploration parameters	Nao simulator: Engagement 10 (Kalman-QL 5)	Robot joints angles	Q	MF	D/ D
Shibata and Sakashita [154]		Actor–critic	Random noise	area with randomly positioned obstacle: 0.6 out of 1	Enumerated state id	Ac	MF	D/ D
Plappert et al. [156]	measure of policies distance	DQN, DDPG and TRPO	Random noise	Atari: BeamRdier 9000 (ϵ-greedy 5000); Mujoco: Half cheetah 5000 (ϵ-greedy 1500)	Atari images/Mujoco joints angles	Ac/ Q	MF	C/ C
Shibata and Sakashita [154]			Random noise	Multi Arm Bandit Problem: 1 (Optimal)	Stateless	Ac	MB	C/ C
Fortunato et al. [155]			Random noise	Atari: 57 games 633 points (Duelling DQN 524)	Atari images	Ac	MF	C/ C

Legend: A — action space, Ac — action, R — reference, MB — model based, MF — model free, D — discrete, C — continuous, Q — Q values, V — values, P — policy, O — output, S — state space, U — underlying algorithm and Top score on a key benchmark explanation - [benchmark]:[scenario] [score] ([baseline approach] [score]).

4. Future challenges

In this section, we discuss the following future challenges on exploration in reinforcement learning: evaluation, scalability, exploration–exploitation dilemma, intrinsic reward, noisy TV problems, safety, and transferability.

Evaluation. Currently, evaluating and comparing different exploration algorithms is challenging. This issue arises from three reasons: lack of a common benchmark, lack of a common evaluation strategy, and lack of good metrics to measure exploration.

Currently, four major benchmarks used by the community are VizDoom [27], Minecraft [28], Atari Games [15] and Mujoco [29]. Each benchmark is characterised by different complexities in terms of state space, reward sparseness, and action space. Moreover, each benchmark offers several scenarios with various degrees of complexity. Such a wealth of benchmarks is desirable for exposure of agents to various complexities; however, the difference in complexity between different benchmarks is well-understood. This leads to difficulty in comparing algorithms using different benchmarks. There have been attempts to solve the evaluation issues using a common benchmark, for example, in [158]. However, this study is not commonly adopted yet.

Regarding the evaluation strategy, most algorithms use a reward after a certain number of steps. Note that in the context of this paragraph, steps could also mean episodes, iterations and epochs. This makes the reporting of results inconsistent in two aspects: (i) the number of steps in which the algorithm was tested and (ii) how the reward is reported. The first makes comparisons between algorithms difficult because performance can vary widely depending on when the comparison is made. The second concern is how rewards are reported. Most authors choose to report the average reward the agent has scored; however, sometimes comparison with the average human performance is used (without clear indication of what average human performance means exactly). Moreover, sometimes the distinction between the average reward or maximum reward is not clearly made.

Finally, it is arguable if a reward is an appropriate measure for evaluation [37]. One of the key issues is that it fails to account for the speed of learning, which should be higher if exploration is more efficient [37]. Attempts have been made to address this issue in [37], but as of the time of writing this review paper, this new metric is not widely adopted. Another issue with rewards is that it does not provide any information regarding the goodness of exploratory behaviour. This is even more difficult in continuous action space problems where computing novelty is considerably more challenging.

Scalability. Exploration in reinforcement learning does not scale well to real-world problems. This is caused by two limitations: training time and inefficient state representation. Currently, even the fastest training requires millions of samples in complex environments. Note that even the most complex environments currently used in reinforcement learning are still relatively simple compared to the real world. In the real world, collecting millions of samples for training is unrealistic owing to wear and tear of physical devices. To cope with the real world, either a sim-to-real gap needs to be reduced or exploration needs to become more sample efficient.

Another limitation is efficient state representation so that memorising states and actions is possible in large domains. For example, Go-Explore [31] does not scale up well if the environment is large. This problem was discussed in [159] by comparing how the brain stores memories and computes novelty. It states that the human brain is much faster at determining scene novelty and has a much larger capacity. To achieve this, the brain uses an agreement between multiple

neurons. The more neurons indicate that the given image is novel, the higher the novelty is. Thus, the brain does not need to remember full states; instead, it trains itself to recognise the novelty. This is currently unmatched in reinforcement learning in terms of the representation efficiency.

Exploration–exploitation dilemma. The exploration–exploitation dilemma is an ongoing research topic not only in reinforcement learning but also in a general problem. Most current exploration approaches have a built-in solution to exploration–exploitation, but not all methods do. This is particularly true in goal-based methods that rely on hand-designed solutions. Moreover, even in approaches that solve it automatically, the balance is still mostly decided by the designer-provided threshold. One potential way of solving this problem is to train a set of skills (policies) during exploration and combine skills in greater goal-oriented policies [160]. This is similar to how humans solve problems by learning smaller skills and then using them later to exploit them as a larger policy.

Intrinsic reward. Reward novel states and diverse behaviour approaches can be improved in two ways: (i) the agent should be more free to reward itself and (ii) better balance between long-term and short-term novelty should be achieved.

In most intrinsic reward approaches, the exact reward formulation is performed by an expert. Designing a reward that guarantees good exploration is a challenging and time-consuming task. Moreover, there might be ways of rewarding agents which were not conceived by designers. Thus, it could be beneficial if an agent is not only trained in the environment but is also trained on how to reward itself. This would be closer to human behaviour where the self-rewarding mechanism was developed through evolution.

Balancing the long-term novelty and short-term novelty is another challenge. In this problem, the agent tries to balance two factors: revisiting states often to find something new or abandoning states quickly to try to find something new. This is currently a hand-designed parameter, but its tuning is time-consuming. Recently, there has been a fix proposed in [15] where meta-learning decides the appropriate balance, but at the cost of computational complexity for training.

Noisy-TV problem. The noisy-TV (or couch potato problem) remains largely unsolved. While using memory can be used to solve it, they are limited by memory requirements. Thus, it can be envisioned that if the noisy sequence is very long and the state space is complex, memory approaches will also struggle to solve it. One method that has shown some promise is the use of clustering [107] to cluster noisy states and avoid that cluster. However, this requires the design of correct clusters.

Optimal exploration. One area which is rarely considered in the current exploration in reinforcement learning research is how to explore optimally. For optimal exploration, the agent does not revisit states unnecessarily and explores the most promising areas first. This problem and the proposed solution are described in detail in [161]. The solution uses a demand matrix, which is an m by n matrix of m states and n actions, indicating state–action exploration counts. It then defines the exploration cost for exploration policy, which is the number of steps each state–action pair needs to be explored. Note that the demand matrix does not need to be known a priori and can be updated online. This aspect needs further developments.

Safe exploration. Safe exploration is of paramount importance for real-world applications. However, so far, there have been very few approaches to cope with this issue. Most of them rely heavily on hand-designed rules to prevent catastrophes. Moreover, it has been shown in [141] that current reinforcement learning is struggling to prevent catastrophes even with carefully engineered rewards. Thus, there exists a need for the agent to recognise unsafe situations and act accordingly. Moreover, what constitutes an unsafe situation is not well defined beyond hand-designed rules. This leads to problems with regard to the scalability and transferability of safe exploration in reinforcement learning. A more rigorous definition of an unsafe situation would be beneficial to address this problem.

Transferability. Most exploratory approaches are currently limited to the domain on which they were trained. When faced with new environments (e.g., increased state space and different reward functions), exploration strategies do not seem to perform well [43,48]. Coping with this issue would be helpful in two scenarios. First, it would be beneficial to be able to teach the agent behaviours in smaller scenarios and then allow it to perform well on larger scenarios to alleviate computational issues. Second, in some domains, defining state spaces suitable for exploration is challenging and may vary in size significantly between tasks (e.g., search for a victim of an accident).

5. Conclusions

This paper presents a review of the exploration in reinforcement learning. The following methods were discussed: reward novel states, reward diverse behaviours, goal-based methods, uncertainty, imitation-based methods, safe exploration, and random methods.

In reward novel state methods, the agent is given a reward for discovering a novel or surprising state. This reward can be computed using prediction error, count, or memory. In prediction error methods, the reward is given based on the accuracy of the agent's internal environmental model. In count-based methods, the reward is given based on how often a given state is visited. In memory-based methods, the reward is computed based on how different a state is compared to other states in a buffer.

In reward diverse behaviour methods, the agent is rewarded for discovering as many diverse behaviours as possible. Note here that we use word behaviour loosely as a sequence of actions or a policy. Reward diverse behaviour methods can be divided into: evolutionary strategies and policy learning. In evolution strategies, diversity among the population of agents is encouraged. In policy learning, the diversity of policy parameters is encouraged.

In goal-based methods, the agent is given the goal of either exploring from or exploring while trying to reach the goal. In the first method, the agent chooses the goal to get to and then explore from it. This results in a very efficient exploration as the agent visits predominantly unknown areas. In the second method, called the exploratory goal, the agent is exploring while travelling towards a goal. The key idea of this method is to provide goals which are suitable for exploration.

In probabilistic methods, the agent holds an uncertainty model about the environment and uses it to make its next move. The uncertainty method has two subcategories: optimistic and uncertainty methods. In optimistic methods, the agent follows the *optimism under uncertainty* principle. This means that the agent will sample the most optimistic understanding of the reward. In uncertainty methods, the agent will sample from internal uncertainty to move towards the least known areas.

Imitation-based methods rely on using demonstrations to help exploration. In general, there are two methods: combining demonstrations with experience replay and combining them with an exploration strategy. In the first method, samples from demonstrations and collected by the agent are combined into one buffer for the agent to learn from. In the second method, the demonstrations are used as a starting point for other exploration techniques such as the reward novel state.

Safe exploration methods were devised to ensure the safe behaviour of the agents during exploration. In safe exploration, the most prevalent method is to use human designer knowledge to develop boundaries for the agent. Furthermore, it is possible to train a model that predicts and stops agents from making a disastrous move. Finally, the agent can be discouraged from visiting dangerous states with a negative reward.

Random exploration methods improve standard random exploration. These improvements include modifying the states for exploration, modifying exploration parameters, and putting the noise on network parameters. In modifying states for exploration, certain states and actions are removed from the random choice if they have been sufficiently explored. In modifying exploration parameter methods, the parameters

affecting when to randomly explore are automatically chosen based on the agent's learning progress. Lastly, in the network parameter noise approach, random noise is applied to the parameters to induce exploration before the weight convergence.

Finally, the best approaches in terms of ease of implementation, computational cost and overall performance are highlighted. The easiest methods to implement are reward novel states, reward diverse behaviours and random-based approaches. Basic implementation of those approaches can be used with almost any other existing reinforcement learning algorithms; they might require a few additions and tuning to work. In terms of computational efficiency, random-based, reward novel states and reward divers behaviours generally require the least resources. Particularly, random-based approaches are computationally efficient as the additional components are lightweight. Currently, best-performing methods are goal-based and reward novel states methods where goal-based methods have achieved high scores in difficult exploratory problems such as Montezuma's revenge. However, goal-based methods tend to be the most complex in terms of implementation. Overall, reward novel states methods seem like a good compromise between ease of implementation and performance.

CRediT authorship contribution statement

Pawel Ladosz: Conceptualization, Investigation, Visualization, Data curation, Writing – original draft, Writing – review & editing. **Lilian Weng:** Conceptualization, Validation, Writing – review & editing. **Minwoo Kim:** Visualization, Writing – review & editing, Learning. **Hyondong Oh:** Conceptualization, Validation, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, A Bradford Book, Cambridge, MA, USA, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533, http://dx.doi.org/10.1038/nature14236, URL: http://dx.doi.org/10.1038/nature14236.

[3] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: 4th International Conference on Learning Representations, ICLR 2016, 2016, arXiv:1509.02971.

[4] S. Lee, H. Bang, Automatic gain tuning method of a quad-rotor geometric attitude controller using A3C, Int. J. Aeronaut. Space Sci. 21 (2) (2020) 469–478, http://dx.doi.org/10.1007/s42405-019-00233-x.

[5] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, A. Cangelosi, Autonomous quadrotor landing using deep reinforcement learning, 2017, arXiv:1709.03339.

[6] B.R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A. Sallab, S. Yogamani, P. Perez, Deep reinforcement learning for autonomous driving: A survey, IEEE Trans. Intell. Transp. Syst. (2021) 1–18, http://dx.doi.org/10.1109/TITS.2021.3054625, arXiv:2002.00444.

[7] C. Yu, J. Liu, S. Nemati, Reinforcement learning in healthcare: A survey, 2019, arXiv:1908.08796.

[8] A. Irpan, Deep reinforcement learning doesn't work yet, 2018, https://www.alexirpan.com/2018/02/14/rl-hard.html.

[9] J. Clark, D. Amodei, Faulty reward functions in the wild, 2016, https://openai.com/blog/faulty-reward-functions/.

[10] J. Schmidhuber, Curious model-building control systems, in: 1991 IEEE International Joint Conference on Neural Networks, IJCNN 1991, 1991, pp. 1458–1463, http://dx.doi.org/10.1109/ijcnn.1991.170605.

[11] J. Schmidhuber, A possibility for implementing curiosity and boredom in model-building neural controllers, in: Proceedings of the First International Conference on Simulation of Adaptive Behavior, 1, 1991, pp. 5–10, URL: ftp://ftp.idsia.ch/pub/juergen/curiositysab.pdf.

[12] B. Eysenbach, A. Gupta, J. Ibarz, S. Levine, Diversity is all you need, in: 7th International Conference on Learning Representations, ICLR 2019, 2019, URL: https://openreview.net/pdf?id=SJx63jRqFm.

[13] Y. Burda, H. Edwards, A. Storkey, O. Klimov, Exploration by random network distillation, in: 7th International Conference on Learning Representations, ICLR 2019, 2018, arXiv:1810.12894.

[14] M.G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, R. Munos, Unifying count-based exploration and intrinsic motivation, in: Conference on Neural Information Processing Systems, NeurIPS 2016, 2016, http://dx.doi.org/10.1002/pola.10609, arXiv:1606.01868.

[15] A.P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, C. Blundell, Agent57: Outperforming the atari human benchmark, 2020, arXiv:2003.13350.

[16] A. Aubret, L. Matignon, S. Hassas, A survey on intrinsic motivation in reinforcement learning, 2019, ArXiv, Im, arXiv:1908.06976.

[17] Y. Li, Deep reinforcement learning, 2018, http://dx.doi.org/10.18653/v1/p18-5007, arXiv:1911.10107.

[18] T.T. Nguyen, N.D. Nguyen, S. Nahavandi, Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications, 2018, pp. 3826–3839, ArXiv, 50 (9).

[19] S. Levine, Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018, ArXiv, arXiv:1805.00909.

[20] A. Lazaridis, A. Fachantidis, I. Vlahavas, Deep reinforcement learning: A state-of-the-art walkthrough, J. Artificial Intelligence Res. 69 (2020).

[21] R. Mcfarlane, A survey of exploration strategies in reinforcement learning, 1999, pp. 1–10, URL: https://pdfs.semanticscholar.org/0276/1533d794ed9ed5dfd0295f2577e1e98c4fe2.pdf.

[22] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3–4) (1992) 229–256, http://dx.doi.org/10.1007/bf00992696.

[23] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38, http://dx.doi.org/10.1109/MSP.2017.2743240, arXiv:arXiv:1708.05866v2.

[24] Exploration, 2020, https://dictionary.cambridge.org/dictionary/english/exploration. (Accessed: 09 April 2020).

[25] M.G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: An evaluation platform for general agents, J. Artificial Intelligence Res. 47 (2013) 253–279, http://dx.doi.org/10.1613/jair.3912, arXiv:1207.4708.

[26] Y. Aytar, T. Pfaff, D. Budden, T. Le Paine, Z. Wang, N. De Freitas, Playing hard exploration games by watching youtube, in: Conference on Neural Information Processing Systems, NeurIPS 2018, 2018, pp. 2930–2941, arXiv:1805.11592.

[27] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jaskowski, ViZDoom: A Doom-based AI research platform for visual reinforcement learning, in: IEEE Conference on Computatonal Intelligence and Games, CIG, 2016, http://dx.doi.org/10.1109/CIG.2016.7860433, arXiv:1605.02097.

[28] M. Johnson, K. Hofmann, T. Hutton, D. Bignell, The malmo platform for artificial intelligence experimentation, in: IJCAI International Joint Conference on Artificial Intelligence, Vol. 2016-Janua, 2016, pp. 4246–4247.

[29] E. Todorov, T. Erez, Y. Tassa, MuJoCo: A physics engine for model-based control, in: IEEE International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033, http://dx.doi.org/10.1109/IROS.2012.6386109.

[30] J. Schmidhuber, Formal theory of creativity, fun, and intrinsic motivation (1990–2010), IEEE Trans. Auton. Ment. Dev. 2 (3) (2010) 230–247, http://dx.doi.org/10.1109/TAMD.2010.2056368, arXiv:1510.05840.

[31] A. Ecoffet, J. Huizinga, J. Lehman, K.O. Stanley, J. Clune, Go-explore: A new approach for hard-exploration problems, 2019, pp. 1–37, arXiv:1901.10995.

[32] P.-Y. Oudeyer, F. Kaplan, What is intrinsic motivation? A typology of computational approaches, Front. Neurorobot. 1 (6) (2007) 1184–1191, http://dx.doi.org/10.3389/neuro.12.006.2007, arXiv:arXiv:1410.5401v2, URL: http://journal.frontiersin.org/article/10.3389/neuro.12.006.2007/abstract.

[33] J. Achiam, S. Sastry, Surprise-based intrinsic motivation for deep reinforcement learning, 2017, pp. 1–13, arXiv:1703.01732.

[34] B. Li, T. Lu, J. Li, N. Lu, Y. Cai, S. Wang, Curiosity-driven exploration for off-policy reinforcement learning methods, in: IEEE International Conference on Robotics and Biomimetics, ROBIO 2019, December, 2019, pp. 1109–1114, http://dx.doi.org/10.1109/ROBIO49542.2019.8961529.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K.Q. Weinberger (Eds.), Conference on Neural Information Processing Systems, NeurIPS 2014, Vol. 27, 2014, URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[36] W. Hong, M. Zhu, M. Liu, W. Zhang, M. Zhou, Y. Yu, P. Sun, Generative adversarial exploration for reinforcement learning, in: ACM International Conference Proceeding Series, 2019, http://dx.doi.org/10.1145/3356464.3357706.

[37] B.C. Stadie, S. Levine, P. Abbeel, Incentivizing exploration in reinforcement learning with deep predictive models, 2015, pp. 1–11, arXiv:1507.00814.

[38] N. Bougie, R. Ichise, Fast and slow curiosity for high-level exploration in reinforcement learning, Appl. Intell. (2020) http://dx.doi.org/10.1007/s10489-020-01849-3.

[39] N. Bougie, R. Ichise, Towards high-level intrinsic exploration in reinforcement learning, in: International Joint Conference on Artificial Intelligence, IJCAI-20, 2020, arXiv:1810.12894.

[40] I. Osband, J. Aslanides, A. Cassirer, Randomized prior functions for deep reinforcement learning, in: Conference on Neural Information Processing Systems, NeurIPS 2018, 2018.

[41] D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 488–489, http://dx.doi.org/10.1109/CVPRW.2017.70, arXiv:1705.05363.

[42] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, A.A. Efros, Large-scale study of curiosity-driven learning, in: 7th International Conference on Learning Representations, ICLR 2019, 2019, arXiv:1808.04355.

[43] R. Raileanu, T. Rocktäschel, RIDE: REwarding impact-driven exploration for procedurally-generated environments, in: 8th International Conference on Learning Representations, ICLR 2020, 2020, arXiv:2002.12292.

[44] J. Li, X. Shi, J. Li, X. Zhang, J. Wang, Random curiosity-driven exploration in deep reinforcement learning, Neurocomputing 418 (2020) 139–147, http://dx.doi.org/10.1016/j.neucom.2020.08.024.

[45] H. Kim, J. Kim, Y. Jeong, S. Levine, H.O. Song, EMI: EXploration with mutual information, in: 36th International Conference on Machine Learning, ICML 2019, 2019, pp. 5837–5851, arXiv:1810.01176.

[46] Y. Kim, W. Nam, H. Kim, J.H. Kim, G. Kim, Curiosity-bottleneck: Exploration by distilling task-specific novelty, in: 36th International Conference on Machine Learning, ICML 2019, 2019, pp. 5861–5874.

[47] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A.J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, R. Hadsell, Learning to navigate in complex environments, in: 5th International Conference on Learning Representations, ICLR 2017, 2017, arXiv:1611.03673.

[48] V. Dhiman, S. Banerjee, B. Griffin, J.M. Siskind, J.J. Corso, A critical investigation of deep reinforcement learning for navigation, 2018, arXiv:1802.02274.

[49] B. Li, T. Lu, J. Li, N. Lu, Y. Cai, S. Wang, ACDER: AUgmented curiosity-driven experience replay, in: IEEE International Conference on Robotics and Automation, ICRA 2020, 2020, pp. 4218–4224, http://dx.doi.org/10.1109/ICRA40945.2020.9197421.

[50] C. Stanton, J. Clune, Deep curiosity search: Intra-life exploration can improve performance on challenging deep reinforcement learning problems, 2018, ArXiv, arXiv:1806.00553.

[51] V. Dean, S. Tulsiani, A. Gupta, See, hear, explore: Curiosity via audio-visual association, 2020, ArXiv, arXiv:2007.03669.

[52] J.Z. Kolter, A.Y. Ng, Near-Bayesian exploration in polynomial time, 2009, pp. 513–520.

[53] D. Pathak, D. Gandhi, A. Gupta, Self-supervised exploration via disagreement, in: Proceedings of the 36th International Conference on Machine Learning, 2019.

[54] S. Still, D. Precup, An information-theoretic approach to curiosity-driven reinforcement learning, Theory Biosci. 131 (3) (2012) 139–148, http://dx.doi.org/10.1007/s12064-011-0142-z.

[55] S. Still, Information theoretic approach to interactive learning, 2009, pp. 1–6, Arxiv.

[56] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, P. Abbeel, VIME: VAriational information maximizing exploration, in: Conference on Neural Information Processing Systems, NeurIPS 2016, 2016, pp. 1117–1125, arXiv:1605.09674.

[57] S. Mohamed, D.J. Rezende, Variational information maximisation for intrinsically motivated reinforcement learning, in: Conference on Neural Information Processing Systems, NeurIPS 2015, 2015, pp. 2125–2133, arXiv:1509.08731.

[58] I.M. De Abril, R. Kanai, Curiosity-driven reinforcement learning with homeostatic regulation, in: Proceedings of the International Joint Conference on Neural Networks, Vol. 2018-July, no. 1, IEEE, 2018, http://dx.doi.org/10.1109/IJCNN.2018.8489075, arXiv:1801.07440.

[59] J.-T. Chien, P.-C. Hsu, Stochastic curiosity maximizing exploration, in: 2020 International Joint Conference on Neural Networks, IJCNN, 2020, http://dx.doi.org/10.1109/ijcnn48605.2020.9207295.

[60] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, S. Gelly, Episodic curiosity through reachability, in: 7th International Conference on Learning Representations, ICLR 2019, 2019, pp. 1–20, arXiv:1810.02274.

[61] P. Ménard, O.D. Domingues, A. Jonsson, E. Kaufmann, E. Leurent, M. Valko, Fast active learning for pure exploration in reinforcement learning, 2020, pp. 1–36, arXiv:2007.13442.

[62] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. De Turck, P. Abbeel, Exploration: A study of count-based exploration for deep reinforcement learning, in: Conference on Neural Information Processing Systems, NeurIPS 2017, 2017, pp. 2754–2763, arXiv:1611.04717.

[63] M.S. Charikar, Similarity estimation techniques from rounding algorithms, in: Conference Proceedings of the Annual ACM Symposium on Theory of Computing, 2002, pp. 380–388, http://dx.doi.org/10.1145/509907.509965.

[64] J. Choi, Y. Guo, M. Moczulski, J. Oh, N. Wu, M. Norouzi, H. Lee, Contingency-aware exploration in reinforcement learning, in: 7th International Conference on Learning Representations, ICLR 2019, 2019, pp. 1–19, arXiv:1811.01483.

[65] M.C. Machado, M.G. Bellemare, M. Bowling, Count-based exploration with the successor representation, in: AAAI Conference on Artificial Intelligence, 2020, http://dx.doi.org/10.1609/aaai.v34i04.5955, arXiv:1807.11622.

[66] R. Zhao, V. Tresp, Curiosity-driven experience prioritization via density estimation, 2019, ArXiv, arXiv:1902.08039.

[67] G. Ostrovski, M.G. Bellemare, A. Van Den Oord, R. Munos, Count-based exploration with neural density models, in: 34th International Conference on Machine Learning, Vol. 6, ICML 2017, 2017, pp. 4161–4175, arXiv:1703.01310.

[68] J. Martin, S.S. Narayanan, T. Everitt, M. Hutter, Count-based exploration in feature space for reinforcement learning, in: IJCAI International Joint Conference on Artificial Intelligence, 2017, pp. 2471–2478, http://dx.doi.org/10.24963/ijcai.2017/344, arXiv:arXiv:1706.08090v1.

[69] T. Malisiewicz, A. Gupta, A.A. Efros, Ensemble of exemplar-SVMs for object detection and beyond, in: Proceedings of the IEEE International Conference on Computer Vision, IEEE, 2011, pp. 89–96, http://dx.doi.org/10.1109/ICCV.2011.6126229.

[70] J. Fu, J.D. Co-Reyes, S. Levine, Ex2: Exploration with exemplar models for deep reinforcement learning, in: Conference on Neural Information Processing Systems, NeurIPS 2017, 2017, pp. 2578–2588, arXiv:1703.01260.

[71] A.P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, C. Blundell, Never give up: Learning directed exploration strategies, in: 8th International Conference on Learning Representations, ICLR 2020, 2020, arXiv:2002.06038.

[72] F.P. Such, V. Madhavan, E. Conti, J. Lehman, K.O. Stanley, J. Clune, Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, 2017, http://dx.doi.org/10.48550/ARXIV.1712.06567, arXiv:1712.06567.

[73] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning, 2017, pp. 476–485, http://dx.doi.org/10.1109/ICSTW.2011.58, ArXiv, arXiv:1703.03864v2.

[74] J. Lehman, K.O. Stanley, Abandoning objectives: Evolution through the search for novelty alone, Evol. Comput. 19 (2) (2011) 189–222, http://dx.doi.org/10.1162/EVCO_a_00025.

[75] S. Risi, S.D. Vanderbleek, C.E. Hughes, K.O. Stanley, How novelty search escapes the deceptive trap of learning to learn, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation - GECCO '09, 2009, p. 153, http://dx.doi.org/10.1145/1569901.1569923, URL: http://portal.acm.org/citation.cfm?doid=1569901.1569923.

[76] E. Conti, V. Madhavan, F.P. Such, J. Lehman, K.O. Stanley, J. Clune, Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents, in: Conference on Neural Information Processing Systems, NeurIPS 2018, 2018, pp. 5027–5038, arXiv:1712.06560.

[77] D. Gravina, A. Liapis, G.N. Yannakakis, Quality diversity through surprise, 2018, pp. 1–14, http://dx.doi.org/10.1109/TEVC.2018.2877215, arXiv:1807.02397,

[78] J.B. Mouret, S. Doncieux, Encouraging behavioral diversity in evolutionary robotics: An empirical study, Evol. Comput. 20 (1) (2012) 91–133, http://dx.doi.org/10.1162/EVCO_a_00048.

[79] J.K. Pugh, L.B. Soros, K.O. Stanley, Quality diversity: A new frontier for evolutionary computation, Front. Robot. AI 3 (July) (2016) http://dx.doi.org/10.3389/frobt.2016.00040, URL: http://journal.frontiersin.org/Article/10.3389/frobt.2016.00040/abstract.

[80] Z.W. Hong, T.Y. Shann, S.Y. Su, Y.H. Chang, C.Y. Lee, Diversity-driven exploration strategy for deep reinforcement learning, in: 6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings, 2018.

[81] A. Cohen, L. Yu, X. Qiao, X. Tong, Maximum entropy diverse exploration: Disentangling maximum entropy reinforcement learning, 2019, arXiv:1911.00828.

[82] V.H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, S. Levine, Skew-fit: State-covering self-supervised reinforcement learning, 2019, arXiv:1903.03698.

[83] T. Gangwani, Q. Liu, J. Peng, Learning self-imitating diverse policies, in: 7th International Conference on Learning Representations, ICLR 2019, 2019.

[84] A. Ecoffet, J. Huizinga, J. Lehman, K.O. Stanley, J. Clune, First return then explore, 2020, pp. 1–46, arXiv:2004.12919.

[85] G. Matheron, N. Perrin, O. Sigaud, PBCS: EFficient exploration and exploitation using a synergy between reinforcement learning and motion planning, in: ICANN 2020, Vol. 12397 LNCS, Springer International Publishing, 2020, pp. 295–307, http://dx.doi.org/10.1007/978-3-030-61616-8_24, arXiv:2004.11667.

[86] Z.D. Guo, E. Brunskill, Directed exploration for reinforcement learning, 2019, arXiv:1906.07805.

[87] Y. Guo, J. Choi, M. Moczulski, S. Bengio, M. Norouzi, H. Lee, Self-imitation learning via trajectory-conditioned policy for hard-exploration tasks, 2019, pp. 1–22, arXiv:1907.10247.

[88] J. Oh, Y. Guo, S. Singh, H. Lee, Self-imitation learning, in: Proceedings of the 35th International Conference on Machine Learning, 2018.

[89] Y. Guo, J. Choi, M. Moczulski, S. Feng, S. Bengio, M. Norouzi, H. Lee, Memory based trajectory-conditioned policies for learning from sparse rewards, in: Conference on Neural Information Processing Systems, NeurIPS 2020, 2020.

[90] E.Z. Liu, R. Keramati, S. Seshadri, K. Guu, P. Pasupat, E. Brunskill, P. Liang, Learning abstract models for strategic exploration and fast reward transfer, 2020, arXiv:2007.05896.

[91] A.D. Edwards, L. Downs, J.C. Davidson, Forward-backward reinforcement learning, 2018, arXiv:1803.10227.

[92] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, P. Abbeel, Reverse curriculum generation for reinforcement learning, 2017, CoRL, arXiv:1707.05300.

[93] S. Forestier, Y. Mollard, P.Y. Oudeyer, Intrinsically motivated goal exploration processes with automatic curriculum learning, 2017, pp. 1–33, ArXiv, arXiv:1708.02190.

[94] C. Colas, P. Founder, O. Sigaud, M. Chetouani, P.Y. Oudeyer, CURIOUS: INtrinsically motivated modular multi-goal reinforcement learning, in: 36th International Conference on Machine Learning, ICML 2019, 2019, pp. 2372–2387, arXiv:1810.06284.

[95] A. Péré, S. Forestier, O. Sigaud, P.Y. Oudeyer, Unsupervised learning of goal spaces for intrinsically motivated goal exploration, in: 6th International Conference on Learning Representations, ICLR 2018, 2018, pp. 1–26, arXiv:1803.00781.

[96] A.S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, K. Kavukcuoglu, Feudal networks for hierarchical reinforcement learning, in: 34th International Conference on Machine Learning, vol. 7, ICML 2017, 2017, pp. 5409–5418, arXiv:1703.01161.

[97] T. Hester, P. Stone, Learning exploration strategies in model-based reinforcement learning, in: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAAI, 2013.

[98] T.D. Kulkarni, K.R. Narasimhan, A. Saeedi, J.B. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, in: Conference on Neural Information Processing Systems, NeurIPS 2016, 2016, http://dx.doi.org/10.1162/NECO, arXiv:NIHMS150003.

[99] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. van de Wiele, V. Mnih, N. Heess, T. Springenberg, Learning by playing - solving sparse reward tasks from scratch, in: Proceedings of the 35th International Conference on Machine Learning, 2018.

[100] M. Ghafoorian, N. Taghizadeh, H. Beigy, Automatic abstraction in reinforcement learning using ant system algorithm, in: AAAI Spring Symposium - Technical Report, Vol. SS-13-05, 2013, pp. 9–14.

[101] M.C. Machado, M.G. Bellemare, M. Bowling, A Laplacian framework for option discovery in reinforcement learning, in: 34th International Conference on Machine Learning, Vol. 5, ICML 2017, 2017, pp. 3567–3582, arXiv:1703.00956.

[102] M.J. Zaki, W. Meira Jr., Data Mining and Analysis: Fundamental Concepts and Algorithms, Cambridge University Press, 2014, http://dx.doi.org/10.1017/CBO9780511810114.

[103] M.C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, M. Campbell, Eigenoption discovery through the deep successor representation, in: 6th International Conference on Learning Representations, ICLR 2018, 2018, arXiv:arXiv:1710.11089v3.

[104] K. Fang, Y. Zhu, S. Savarese, L. Fei-Fei, Adaptive procedural task generation for hard-exploration problems, in: ICLR 2021, 2020, arXiv:2007.00350, submitted for publication.

[105] C. Guestrin, R. Patrascu, D. Schuurmans, Algorithm-directed exploration for model-based reinforcement learning in factored mdps, in: Machine Learning International Workshop, 2002, pp. 235–242, URL: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Algorithm-Directed+Exploration+for+Model-Based+Reinforcement+Learning+in+Factored+MDPs{#}0.

[106] D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, R.E. Schapire, Exploratory gradient boosting for reinforcement learning in complex domains, 2016, arXiv:1603.04119.

[107] G. Kovač, A. Laversanne-Finot, P.-Y. Oudeyer, GRIMGEP: LEarning progress for robust goal sampling in visual deep reinforcement learning, 2020, pp. 1–15, CoRL, arXiv:2008.04388v1.

[108] I. Osband, B. Van Roy, Why is posterior sampling better than optimism for reinforcement learning? in: 34th International Conference on Machine Learning, ICML 2017, 2017, pp. 4133–4148, arXiv:1607.00215.

[109] T. Jung, P. Stone, Gaussian Processes for sample efficient reinforcement learning with rmax-like exploration, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6321 LNAI, (PART 1) 2010, pp. 601–616, http://dx.doi.org/10.1007/978-3-642-15880-3_44, arXiv:1201.6604.

[110] C. Xie, S. Patil, T. Moldovan, S. Levine, P. Abbeel, Model-based reinforcement learning with parametrized physical models and optimism-driven exploration, in: IEEE International Conference on Robotics and Automation, ICRA 2016, IEEE, 2016, pp. 504–511, http://dx.doi.org/10.1109/ICRA.2016.7487172, arXiv:1509.06824.

[111] C. D'Eramo, A. Cini, M. Restelli, Exploiting action-value uncertainty to drive exploration in reinforcement learning, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2019, IEEE, 2019, http://dx.doi.org/10.1109/IJCNN.2019.8852326.

[112] I. Osband, B. Van Roy, Z. Wen, Generalization and exploration via randomized value functions, in: 33rd International Conference on Machine Learning, ICML 2016, 2016, pp. 3540–3561, arXiv:1402.0635.

[113] I. Osband, B. Van Roy, D.J. Russo, Z. Wen, Deep exploration via randomized value functions, J. Mach. Learn. Res. 20 (2019) 1–62, arXiv:1703.07608.

[114] Y. Tang, S. Agrawal, Exploration by distributional reinforcement learning, in: IJCAI International Joint Conference on Artificial Intelligence, Vol. 2018-July, 2018, pp. 2710–2716, http://dx.doi.org/10.24963/ijcai.2018/376, arXiv:1805.01907.

[115] C. Colas, O. Sigaud, P.Y. Oudeyer, GEP-PG: DEcoupling exploration and exploitation in deep reinforcement learning algorithms, in: Proceedings of the 35th International Conference on Machine Learning, ICML 2018, 2018.

[116] D. Janz, J. Hron, P. Mazur, K. Hofmann, J.M. Hernández-Lobato, S. Tschiatschek, Successor uncertainties: Exploration and uncertainty in temporal difference learning, in: Conference on Neural Information Processing Systems, Vol. 33, NeurIPS 2019, 2019, arXiv:1810.06530.

[117] F. Stulp, Adaptive exploration for continual reinforcement learning, in: IEEE International Conference on Intelligent Robots and Systems, IROS 2012, IEEE, 2012, pp. 1631–1636, http://dx.doi.org/10.1109/IROS.2012.6385818.

[118] T. Akiyama, H. Hachiya, M. Sugiyama, Efficient exploration through active learning for value function approximation in reinforcement learning, Neural Netw. 23 (5) (2010) 639–648, http://dx.doi.org/10.1016/j.neunet.2009.12.010, URL: http://dx.doi.org/10.1016/j.neunet.2009.12.010.

[119] M. Strens, A Bayesian framework for reinforcement learning, in: Proc of the 17th International Conference on Machine Learning, 2000, pp. 943–950, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.1701{&}rep=rep1{&}type=pdf.

[120] A. Guez, D. Silver, P. Dayan, Efficient Bayes-adaptive reinforcement learning using sample-based search, in: Conference on Neural Information Processing Systems, NeurIPS 2012, 2012, pp. 1025–1033, arXiv:1205.3109.

[121] B. O'Donoghue, I. Osband, R. Munos, V. Mnih, The uncertainty Bellman equation and exploration, in: 35th International Conference on Machine Learning Vol. 9, ICML 2018, 2018, pp. 6154–6173, arXiv:1709.05380.

[122] N. Nikolov, J. Kirschner, F. Berkenkamp, A. Krause, Information-directed exploration for deep reinforcement learning, in: 7th International Conference on Learning Representations, ICLR 2019, 2019, arXiv:1812.07544.

[123] I. Osband, C. Blundell, A. Pritzel, B.V. Roy, Deep exploration via bootstrapped DQN, in: Conference on Neural Information Processing Systems, NeurIPS 2016, 2016.

[124] T. Pearce, N. Anastassacos, M. Zaki, A. Neely, BayesIan inference with anchored ensembles of neural networks, and application to exploration in reinforcement learning, in: Exploration in Reinforcement Learning Work- Shop At the 35th International Conference on Machine Learning, 2018, arXiv:1805.11324.

[125] P. Shyam, W. Jaskowski, F. Gomez, Model-based active exploration, in: 36th International Conference on Machine Learning, ICML 2019, 2019, pp. 10136–10152, arXiv:1810.12162.

[126] M. Henaff, Explicit explore-exploit algorithms in continuous state spaces, in: Conference on Neural Information Processing Systems, NeurIPS 2019, 2019, arXiv:1911.00617.

[127] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, M. Riedmiller, Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards, 2017, arXiv:1707.08817.

[128] T. Hester, T. Schaul, A. Sendonaris, M. Vecerik, B. Piot, I. Osband, O. Pietquin, D. Horgan, G. Dulac-Arnold, M. Lanctot, J. Quan, J. Agapiou, J.Z. Leibo, A. Gruslys, Deep q-learning from demonstrations, in: 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, 2018, pp. 3223–3230, arXiv:1704.03732.

[129] C. Gulcehr, T.L. Paine, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams, G. Barth-Maron, Z. Wang, N. de Freitas, Making efficient use of demonstrations to solve hard exploration problems, in: 8th International Conference on Learning Representations, ICLR 2020, 2020.

[130] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, P. Abbeel, Overcoming exploration in reinforcement learning with demonstrations, in: Proceedings - IEEE International Conference on Robotics and Automation, IEEE, 2018, pp. 6292–6299, http://dx.doi.org/10.1109/ICRA.2018.8463162, arXiv:1709.10089.

[131] T. Salimans, R. Chen, Learning Montezuma's revenge from a single demonstration, in: Conference on Neural Information Processing Systems, NeurIPS 2018, 2018, arXiv:1812.03381.

[132] K. Subramanian, C.L. Isbell, A.L. Thomaz, Exploration from demonstration for interactive reinforcement learning, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2016, pp. 447–456.

[133] J. Garcıa, F. Fernandez, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (2015).

[134] J. Garcia, F. Fernandez, Safe exploration of state and action spaces in reinforcement learning, J. Artificial Intelligence Res. 45 (2012) 515–564, http://dx.doi.org/10.1613/jair.3761.

[135] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, Y. Tassa, Safe exploration in continuous action spaces, 2018, arXiv:1801.08757.

[136] E. Garcelon, M. Ghavamzadeh, A. Lazaric, M. Pirotta, Conservative exploration in reinforcement learning, in: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, 2020, arXiv:2002.03218.

[137] N. Hunt, N. Fulton, S. Magliacane, N. Hoang, S. Das, A. Solar-Lezama, Verifiably safe exploration for end-to-end reinforcement learning, 2020, arXiv:2007.01223.

[138] W. Saunders, A. Stuhlmüller, G. Sastry, O. Evans, Trial without error: Towards safe reinforcement learning via human intervention, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2018, pp. 2067–2069, arXiv:1707.05173.

[139] M. Turchetta, F. Berkenkamp, A. Krause, Safe exploration in finite Markov decision processes with Gaussian processes, in: Conference on Neural Information Processing Systems, NeurIPS 2016, 2016, pp. 4312–4320, arXiv:1606.04753.

[140] C. Gao, B. Kartal, P. Hernandez-Leal, M.E. Taylor, On hard exploration for reinforcement learning: a case study in pommerman, in: Fifteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2019, arXiv:1907.11788.

[141] Z.C. Lipton, K. Azizzadenesheli, A. Kumar, L. Li, J. Gao, L. Deng, Combating reinforcement learning's sisyphean curse with intrinsic fear, 2016, arXiv:1611.01211.

[142] M. Fatemi, S. Sharma, H. van Seijen, S.E. Kahou, Dead-ends and secure exploration in reinforcement learning, in: 36th International Conference on Machine Learning, ICML 2019, 2019, pp. 3315–3323.

[143] T.G. Karimpanal, S. Rana, S. Gupta, T. Tran, S. Venkatesh, Learning transferable domain priors for safe exploration in reinforcement learning, 2020, pp. 1–10, http://dx.doi.org/10.1109/ijcnn48605.2020.9207344, arXiv:1909.04307.

[144] R. Patrascu, D. Stacey, Adaptive exploration in reinforcement learning, in: Proceedings of the International Joint Conference on Neural Networks, Vol. 4, 1999, pp. 2276–2281, http://dx.doi.org/10.1109/ijcnn.1999.833417.

[145] M. Tokic, Adaptive ϵ-greedy exploration in reinforcement learning based on value differences, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 6359 LNAI, 2010, pp. 203–210, http://dx.doi.org/10.1007/978-3-642-16111-7_23.

[146] M. Usama, D.E. Chang, Learning-driven exploration for reinforcement learning, 2019, arXiv:1906.06890.

[147] L. Shani, Y. Efroni, S. Mannor, Exploration conscious reinforcement learning revisited, in: 36th International Conference on Machine Learning, ICML 2019, 2019, pp. 9986–10012, arXiv:1812.05551.

[148] M. Khamassi, G. Velentzas, T. Tsitsimis, C. Tzafestas, Active exploration and parameterized reinforcement learning applied to a simulated human-robot interaction task, in: 2017 1st IEEE International Conference on Robotic Computing, IRC 2017, IEEE, 2017, pp. 28–35, http://dx.doi.org/10.1109/IRC.2017.33.

[149] H.S. Chang, An ant system based exploration-exploitation for reinforcement learning, in: Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, Vol. 4, 2004, pp. 3805–3810, http://dx.doi.org/10.1109/ICSMC.2004.1400937.

[150] S. Grossberg, Competitive learning: From interactive activation to adaptive resonance, Cogn. Sci. 11 (1) (1987) 23–63, http://dx.doi.org/10.1016/S0364-0213(87)80025-3.

[151] T.H. Teng, A.H. Tan, Knowledge-based exploration for reinforcement learning in self-organizing neural networks, in: Proceedings - 2012 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2012, IEEE, 2012, pp. 332–339, http://dx.doi.org/10.1109/WI-IAT.2012.154.

[152] P. Wang, W.J. Zhou, D. Wang, A.H. Tan, Probabilistic guided exploration for reinforcement learning in self-organizing neural networks, in: Proceedings - 2018 IEEE International Conference on Agents, ICA 2018, IEEE, 2018, pp. 109–112, http://dx.doi.org/10.1109/AGENTS.2018.8460067.

[153] T. Rückstieß, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, J. Schmidhuber, Exploring parameter space in reinforcement learning, J. Behav. Robot. 1 (1) (2010) 14–24, http://dx.doi.org/10.2478/s13230-010-0002-4.

[154] K. Shibata, Y. Sakashita, Reinforcement learning with internal-dynamics-based exploration using a chaotic neural network, in: Proceedings of the International Joint Conference on Neural Networks, IJCNN 2015, IEEE, 2015, http://dx.doi.org/10.1109/IJCNN.2015.7280430.

[155] M. Fortunato, M.G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, S. Legg, Noisy networks for exploration, in: 6th International Conference on Learning Representations, ICLR 2018, 2018, pp. 1–21, arXiv:1706.10295.

[156] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R.Y. Chen, X. Chen, T. Asfour, P. Abbeel, M. Andrychowicz, Parameter space noise for exploration, in: 6th International Conference on Learning Representations, ICLR 2018, 2018, pp. 1–18, arXiv:1706.01905.

[157] T. Rückstieß, M. Felder, J. Schmidhuber, State-dependent exploration for policy gradient methods, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5212 LNAI, (PART 2) 2008, pp. 234–249, http://dx.doi.org/10.1007/978-3-540-87481-2_16.

[158] I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepezvari, S. Singh, B. van Roy, R. Sutton, D. Silver, H. van Hasselt, Behaviour suite for reinforcement learning, in: 8th International Conference on Learning Representations, ICLR 2020, 2020, arXiv:1908.03568.

[159] A. Jaegle, V. Mehrpour, N. Rust, Visual novelty, curiosity, and intrinsic reward in machine learning and the brain, Curr. Opin. Neurobiol. 58 (2019) 167–174, http://dx.doi.org/10.1016/j.conb.2019.08.004, arXiv:1901.02478.

[160] OpenAI, Asymmetric self-play for automatic goal discovery in robotic manipulation, 2021, arXiv:2101.04882.

[161] L. Zhang, K. Tang, X. Yao, Explicit planning for efficient exploration in reinforcement learning, in: Conference on Neural Information Processing Systems, Vol. 32, NeurIPS 2019, 2019.