Decoupled Reinforcement Learning to Stabilise Intrinsically-Motivated Exploration

Lukas Schäfer
University of Edinburgh
Edinburgh, United Kingdom
l.schaefer@ed.ac.uk

Filippos Christianos
University of Edinburgh
Edinburgh, United Kingdom
f.christianos@ed.ac.uk

Josiah P. Hanna
University of Wisconsin–Madison
Madison, United States of America
jphanna@cs.wisc.edu

Stefano V. Albrecht
University of Edinburgh
Edinburgh, United Kingdom
s.albrecht@ed.ac.uk

ABSTRACT

Intrinsic rewards can improve exploration in reinforcement learning, but the exploration process may suffer from instability caused by non-stationary reward shaping and strong dependency on hyperparameters. In this work, we introduce Decoupled RL (DeRL) as a general framework which trains separate policies for intrinsically-motivated exploration and exploitation. Such decoupling allows DeRL to leverage the benefits of intrinsic rewards for exploration while demonstrating improved robustness and sample efficiency. We evaluate DeRL algorithms in two sparse-reward environments with multiple types of intrinsic rewards. Our results show that DeRL is more robust to varying scale and rate of decay of intrinsic rewards and converges to the same evaluation returns than intrinsically-motivated baselines in fewer interactions. Lastly, we discuss the challenge of distribution shift and show that divergence constraint regularisers can successfully minimise instability caused by divergence of exploration and exploitation policies.

KEYWORDS

Reinforcement Learning; Exploration; Intrinsic Rewards

ACM Reference Format:
Lukas Schäfer, Filippos Christianos, Josiah P. Hanna, and Stefano V. Albrecht. 2022. Decoupled Reinforcement Learning to Stabilise Intrinsically-Motivated Exploration. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 37 pages.

1 INTRODUCTION

Exploration is one of the essential challenges in reinforcement learning (RL). RL algorithms often use simple randomised methods, e.g. applying ϵ-greedy policies [41] or adding random noise to continuous actions [17]. Such exploration techniques may be inefficient on tasks where rewards are sparse. One category of exploration techniques which has been found to be particularly effective in sparse-reward environments are intrinsic rewards [6, 7, 11, 28, 30, 31]. These additional rewards r^i are computed by the agent and added to the extrinsic reward r^e provided by the environment for a combined reward signal $r = r^e + \lambda r^i$ with some weighting factor λ. Intrinsic rewards incentivise the exploration of novel or underexplored parts of the environment commonly using self-supervised predictions in the environment [6, 30, 31, 33] or (pseudo-) counts of states [25, 36, 39].

Unfortunately, optimising for this combined feedback introduces three challenges. (1) **Intrinsic rewards lead to non-stationary rewards** as they are designed to diminish with more completed exploration. Such non-stationary reward shaping violates the Markov assumption and can cause the learning progress to be inconsistent. (2) **Intrinsically-motivated exploration is sensitive to the scale** λ. If intrinsic rewards are too large, they might heavily distort training and introduce non-stationary noise to the optimisation. On the other hand, we show that small intrinsic rewards have no sufficient impact and do not incentivise exploration as intended (Figure 4). (3) **Intrinsically-motivated exploration is sensitive to the rate of decay** which intrinsic rewards rely on throughout training. Similar to their scale, we show that slowly decaying intrinsic rewards disrupt training whereas quickly vanishing intrinsic rewards have insufficient impact on exploration (Figure 5).

These challenges lead to a significant dependency of intrinsic rewards on hyperparameters. Additionally, determining these hyperparameters for scale and rate of decay is task-dependent due to the scale of extrinsic rewards and required exploration in the respective task. Current approaches usually address the difficulties caused by sensitivity to hyperparameters using a large hyperparameter search to find effective parameterisation of a method. However, such a search can be considered an exploration by itself, and introduces bias in reported results focusing only on runs with best-identified hyperparameters and disregarding the considerable computational cost involved [29]. We argue that this bias is particularly harmful in approaches focusing on exploration as best-identified hyperparameters may steer exploration towards the solution of an environment, effectively shifting the achieved exploration from the proposed method to the hyperparameter search. All these properties make the practical application of such methods difficult [38] and motivate the need for more robust approaches.

Motivated by these challenges and success in off-policy RL [9, 12, 13, 35, 44], we propose to separate the RL training into two separate policies. We train an *exploration policy* π_β with the combined signal of extrinsic and intrinsic rewards. Simultaneously, we train an *exploitation policy* π_e using only extrinsic rewards on the data

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

collected by the exploration policy. We refer to this approach as **Decoupled RL** (DeRL)*. Using such decoupling addresses challenges (1)–(3) of previous application of intrinsic rewards. The exploration policy is optimised using the combined objective of extrinsic and intrinsic rewards as in typical intrinsically-motivated RL, but it is only trained to generate data for the training of the exploitation policy. The exploitation policy is thereby decoupled from the challenges of training with intrinsic rewards and optimised to be an effective policy in the given environment. Our experiments show that DeRL leverages the benefits of intrinsically-motivated exploration while stabilising its inherent sensitivity to scale and rate of decay of intrinsic rewards.

We implement and evaluate two versions of DeRL built upon on-policy actor-critic and off-policy Q-learning with five types of intrinsic rewards [6, 30, 31, 39] in two learning environments that focus on exploration. We analyse the sensitivity of DeRL and RL baselines to the scale and the rate of decay of intrinsic rewards to verify the general dependency of these methods on the hyperparameters of intrinsic rewards and show that DeRL is more robust to varying hyperparameters. Additionally, the exploitation policy of several DeRL algorithms is able to converge to higher evaluation returns using up to $\sim$ 40% fewer interactions and reaches higher returns in some tasks compared to intrinsically-motivated RL baselines. Such improved robustness and sample efficiency can justify the additional cost of training a second policy. However, we also observe that DeRL still suffers from variability in the off-policy optimisation of the exploitation policy π_e in several tasks. We hypothesise that distribution shift caused by the divergence of π_e and π_β leads to these instabilities, and show that regularisers can be applied to restrict divergence of both policies [43], reducing deviations in returns of exploration and exploitation policies and further improving robustness to hyperparameters of intrinsic rewards.

2 BACKGROUND

2.1 Markov Decision Process

We formulate an environment as a Markov Decision Process (MDP) [14] defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ denote the sets of states and actions, respectively, and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ represents the transition function defining a probability distribution over the next state given current state and applied action. The agent receives rewards for a given transition following $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. The objective is to learn a policy $\pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$ which maximises the expected discounted returns $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) \mid a_t \sim \pi(s_t)\right]$ with discount factor $\gamma \in [0, 1)$.

2.2 Intrinsically-Motivated Exploration

A variety of methods have been proposed to replicate the exploration incentive of curiosity in RL [3, 26, 27] guided by the idea that agents should be incentivised to explore novel or poorly understood parts of the environment. Therefore, intrinsic rewards are defined which reward the agent for such exploration. Over time, the agent should become less "curious" and exploitation will gradually take over. There are two common branches of intrinsic rewards for exploration: (1) count-based and (2) prediction-based intrinsic rewards.

2.2.1 *Count-based Intrinsic Rewards.*

Count-based intrinsic rewards are inverse proportional to the visitations of encountered states.

$$r_t^i := \frac{1}{\sqrt{N(s_t)}} \tag{1}$$

Thereby, agents are incentivised to visit states within the environment which are less frequently encountered. Likewise, agents are discouraged from visiting frequently encountered states which are deemed less valuable for exploration. While this approach is easily applicable in small, discrete state spaces, pseudo-counts have to be computed for large or continuous state spaces where encountering any state multiple times is rare. These pseudo-counts can be computed using density models predicting visitations of states [4, 25] or using locality-sensitive [2] hash functions [39].

2.2.2 *Prediction-based Intrinsic Rewards.*

A separate approach defines intrinsic rewards using predictions in the environment. Schmidhuber [33] proposed an intrinsic reward defined as the error of predicting the next state given the current state and action. However, stochastic and thereby unpredictable dynamics within the environment lead to the so-called "noisy TV problem" [5], i.e. the exploration signal remains high in the presence of unpredictability, which remains a major challenge of these approaches.

Intrinsic curiosity module (ICM): Pathak et al. [30] propose to learn efficient state representations $\phi(s)$ and assign an intrinsic reward for the prediction error of the next state

$$r_t^i := \left(\widehat{\phi}(s_{t+1}) - \phi(s_{t+1})\right)^2 \tag{2}$$

where ϕ is a learned self-supervised state-representation trained using an inverse-dynamics objective: given a representation of the current state $\phi(s_t)$ and next state $\phi(s_{t+1})$ predict the applied action a_t. Through this representation, the model learns to encode information which can be affected by the agent's actions. The intrinsic reward is given by the error of the prediction $\widehat{\phi}(s_{t+1})$ of the next state given current state s_t and applied action a_t.

Rewarding impact-driven exploration (RIDE): Raileanu and Rocktäschel [31] propose to reward the agent for applying actions which lead to significant change in the environment. Such change is defined as the difference between embeddings of consecutive states, where the embedding function ϕ is trained using an inverse dynamics model identical to ICM [30]. In order to avoid the agent going back and forth between a group of states, an episodic state-count N_{ep} is added to the objective.

$$r_t^i := \frac{(\phi(s_{t+1}) - \phi(s_t))^2}{\sqrt{N_{ep}(s_{t+1})}} \tag{3}$$

Random network distillation (RND): Burda et al. [6] propose a simplified prediction-based intrinsic reward which optimises a state representation function $\widehat{\phi}$ to mimic a randomly initialised, fixed target representation ϕ.

$$r_t^i := \left(\widehat{\phi}(s_t) - \phi(s_t)\right)^2 \tag{4}$$

[*] We provide an implementation of DeRL and the Hallway environment at https://github.com/uoe-agents/derl

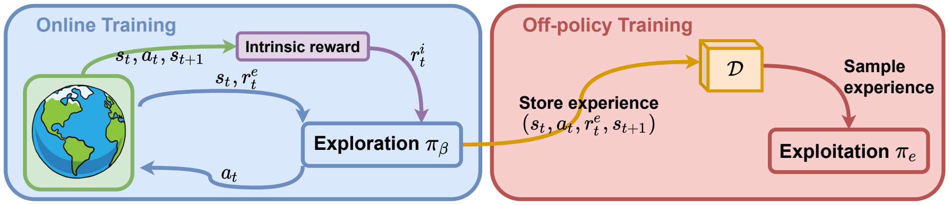


Figure 1: Visualisation of Decoupled Reinforcement Learning (DeRL) training loop.

3 RELATED WORK

Liu et al. [19] propose a new objective for Meta-RL optimisation to decouple exploration and exploitation. Using such an objective, they train separate exploration and exploitation policies guided by task-specific information for fast adaptation to novel scenarios. For multi-agent RL, Liu et al. [20] learn separate exploration and exploitation policies using off-policy RL to focus coordinated exploration across multiple agents towards underexplored parts within the state space. However, a mixture of both policies is applied to explore whereas our work fully decouples both policies and their training. Furthermore, both of these approaches consider the meta-learning and multi-agent settings, respectively, and do not address the challenge of single-agent exploration we focus on.

Independently from our work, Whitney et al. [42] propose to concurrently train an exploration policy using only intrinsic rewards and train a task policy using off-policy soft Double-DQN [40]. They apply a factored policy of both the task and exploration policies with optimisations focused on fast adaptation of the exploration policy. In contrast, we fully decouple both trained policies and find that training of the task policy using on-policy actor-critic algorithms with off-policy correction leads to higher returns and less sensitivity to hyperparameters in several tasks. Furthermore, we evaluate DeRL with several intrinsic rewards whereas Whitney et al. [42] use a single count-based intrinsic reward which is also used for optimisitic initialisation [32].

4 DECOUPLED REINFORCEMENT LEARNING

In this work, we propose to decouple exploration and exploitation into two separate policies to improve sample efficiency and reduce sensitivity to hyperparameters of intrinsic rewards. We train an exploration policy π_β with the intent to explore the environment. Using the data collected by the exploration policy, we train a separate exploitation policy π_e, as visualised in Figure 1. Separating exploration and exploitation in this way enables training of the exploration policy with intrinsic rewards without modifying the training objective of the exploitation policy.

Formally, an agent trains an exploration policy π_β to maximise the sum of intrinsic and extrinsic rewards,

$$\pi_\beta \in \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(r_t^e + \lambda r_t^i\right) \mid a_t \sim \pi(s_t)\right] \tag{5}$$

$$= \arg\max_\pi \mathbb{E}\left[G_t^{e+i} \mid a_t \sim \pi(s_t)\right] \tag{6}$$

with G^{e+i} denoting the discounted returns computed using the combination of extrinsic and intrinsic rewards with scaling factor λ and discount factor $\gamma \in [0, 1)$. During training of π_β, experience samples $(s_t, a_t, r_t^e, s_{t+1})$ with extrinsic rewards are collected in $\mathcal{D}$.

In addition to this typical intrinsically-motivated RL, we train a separate exploitation policy π_e to maximise only expected cumulative extrinsic rewards using experience accumulated in $\mathcal{D}$ with G_t^e denoting discounted extrinsic returns.

$$\pi_e \in \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^e \mid (s_t, a_t, r_t^e, s_{t+1}) \sim \mathcal{D}\right] \tag{7}$$

$$= \arg\max_\pi \mathbb{E}\left[G_t^e \mid (s_t, a_t, r_t^e, s_{t+1}) \sim \mathcal{D}\right] \tag{8}$$

Both exploration policy π_β and exploitation policy π_e can be trained using any RL algorithm given the defined objectives. We optimise the exploration policy π_β as RL with intrinsic rewards [3, 26, 27], whereas we train π_e every T_{Dec} timesteps on experience from $\mathcal{D}$ which is generated by π_β's interaction in the environment. Note that $\mathcal{D}$ only contains extrinsic rewards and is off-policy data for the optimisation of π_e as it was generated by following π_β. Therefore, training the exploitation policy using experience generated by the exploration policy requires off-policy RL. Off-policy RL is concerned with the optimisation of a policy using experience generated within the environment by following a separate behaviour policy. Below, we propose two methods to apply such decoupled RL using an actor-critic and Q-learning framework.

4.1 Decoupled Actor-Critic

In order to use on-policy RL algorithms to train π_e using $\mathcal{D}$, off-policy correction must be applied to account for differences in trajectory distributions of both π_e and π_β.

One technique for off-policy correction is *importance sampling* (IS). In the following, we train π_e using an on-policy actor-critic RL algorithm with state value function V, parameterised by θ, and policy π_e, parameterised by ϕ. We optimise the latter by minimising the actor loss given in Equation (9)

$$\mathcal{L}(\phi) = \mathbb{E}\left[-\rho(a_t|s_t) \log \pi_e(a_t|s_t;\phi)\, A^e(s_t) \mid (s_t, a_t, r_t^e, s_{t+1}) \sim \mathcal{D}\right] \tag{9}$$

with bootstrapped advantage estimates $A^e(s_t)$ and IS weights $\rho(a_t|s_t)$.

$$A^e(s_t) = \left(r_t^e + \gamma V(s_{t+1};\theta) - V(s_t;\theta)\right) \tag{10}$$

$$\rho(a_t|s_t) = \frac{\pi_e(a_t|s_t;\phi)}{\pi_\beta(a_t|s_t)} \tag{11}$$

Similarly, the value loss for π_e using IS weights can be defined as follows:

Algorithm 1 Decoupled Actor-Critic

```
Initialise: parameters $\phi$, $\theta$ and $\pi_\beta$
$\mathcal{D} \leftarrow \emptyset$
$i \leftarrow 0$
for ep = $0, \dots, N_{ep}$ do
    $a_t \sim \pi_\beta(s_t)$
    $s_{t+1}, r^e_t \leftarrow$ environment step with $a_t$
    Update $\pi_\beta$ using RL on intrinsic rewards (Equation (6))
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r^e_t, s_{t+1})$
    $i \leftarrow i + 1$
    if $i \mod T_{Dec} = 0$ then
        Update $\phi$ with Equation (9) and $\mathcal{D}$
        Update $\theta$ with Equation (12) and $\mathcal{D}$
        $\mathcal{D} \leftarrow \emptyset$
    end if
end for
```

$$\mathcal{L}(\theta) = \mathbb{E}\left[\rho(a_t|s_t)\ \left(V(s_t;\theta) - \left(r^e_t + \gamma V(s_{t+1};\theta)\right)\right)^2 \mid (s_t, a_t, r^e_t, s_{t+1}) \sim \mathcal{D}\right] \quad (12)$$

The IS weights, ρ, can cause inconsistent returns during off-policy training through exploding weights when $\pi_e(a_t|s_t;\phi) \gg \pi_\beta(a_t|s_t)$ or vanishing weights for $\pi_e(a_t|s_t;\phi) \ll \pi_\beta(a_t|s_t)$ [8]. In particular, such instabilities occur when one policy assigns approximately zero probability for some action. Various techniques have been proposed to address such exploding weights, including clipping of importance weights [10, 23] to minimise vanishing or exploding gradients. The pseudocode for Decoupled Actor-Critic optimisation of π_e can be found in Algorithm 1.

4.2 Decoupled Deep Q-Networks

Instead of optimising π_e using actor-critic algorithms with off-policy corrections, we can also apply off-policy algorithms such as Q-learning without the need for any correction. In this work, we consider optimising π_e using Deep Q-Networks (DQN) [22]. For DQN optimisation, the following loss is minimised

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left(Q(s_t, a_t;\theta) - \bar{Q}(s_t)\right)^2 \mid (s_t, a_t, r^e_t, s_{t+1}) \sim \mathcal{D}\right] \quad (13)$$

with target Q-values $\bar{Q}(s_t)$ and $\bar{\theta}$ denoting the parameters of the periodically updated target network.

$$\bar{Q}(s_t) = (r^e_t + \gamma \max_{a'} Q(s_{t+1}, a';\bar{\theta})) \quad (14)$$

Pseudocode for Decoupled Deep Q-Networks of π_e can be found in Algorithm 2. Note that $\mathcal{D}$ is only used for a single update in Decoupled Actor-Critic, whereas in Decoupled Deep Q-Networks $\mathcal{D}$ represents a replay buffer [18] which is continually filled with experience.

5 EVALUATION

We evaluate DeRL in two learning environments with a variety of RL algorithms and intrinsic rewards. In particular, we investigate the following three hypotheses: (1) Intrinsically-motivated RL is sensitive to varying scale λ and rate of decay of intrinsic rewards, (2) DeRL is more robust than intrinsically-motivated RL baselines to varying scale and rate of decay, and (3) DeRL leads to similar or improved returns and sample efficiency compared to intrinsically-motivated RL baselines.

Algorithm 2 Decoupled Deep Q-Networks

```
Initialise: parameters $\theta$ and $\pi_\beta$
$\mathcal{D} \leftarrow \emptyset$
$i \leftarrow 0$
for ep = $0, \dots, N_{ep}$ do
    $a_t \sim \pi_\beta(s_t)$
    $s_{t+1}, r^e_t \leftarrow$ environment step with $a_t$
    Update $\pi_\beta$ using RL on intrinsic rewards (Equation (6))
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r^e_t, s_{t+1})$
    $i \leftarrow i + 1$
    if $i \mod T_{Dec} = 0$ then
        Update $\theta$ with Equation (13) and $\mathcal{D}$
    end if
end for
```

5.1 Algorithms

Baselines: As baselines, we consider on-policy RL algorithms Advantage Actor-Critic (A2C) [21] and Proximal Policy Optimisation (PPO) [34]. Both algorithms are trained using the combined reward $r_t = r^e_t + \lambda r^i_t$ with some weighting factor λ and various intrinsic reward definitions as stated below.

DeRL: For our decoupled RL optimisation, we consistently train π_β using A2C as we found it to be more robust than PPO. As intrinsic rewards, we use Count and ICM to train π_β. For the optimisation of π_e, we consider A2C and PPO for Decoupled Actor-Critic and Decoupled Deep Q-Networks based on DQN. We refer to these algorithms as DeA2C, DePPO and DeDQN.

5.2 Intrinsic Rewards

Count-based: We consider two count-based intrinsic rewards computing intrinsic rewards following Equation (1). **Count** directly stores and increments state occurrences in a table. **Hash-Count** first groups states using the SimHash function [39].

Prediction-based: Besides count-based intrinsic exploration definitions, we consider **ICM** [30], **RND** [6], and **RIDE** [31] as prediction-based approaches. For details on these intrinsic rewards, see Section 2.2.2.

5.3 Environments

DeepSea is an environment proposed as part of the Behaviour Suite (Bsuite) for RL [24], visualised in Figure 2. The environment targets the challenge of exploration and represents a $N \times N$ grid where the agent starts in the top left and has to reach a goal in the bottom right location. At each timestep, the agent moves one row down and can choose one out of two actions. For each row, both actions are randomly assigned to left and right movement. The agent observes the current location as a 2D one-hot encoding and receives a small negative reward of $\frac{-0.01}{N}$ for moving right and 0 reward for moving left. Additionally, the agent receives a reward of +1 for reaching the goal and the episode ends after N timesteps.

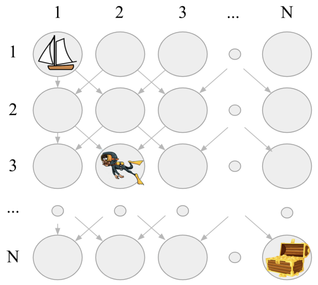


Figure 2: DeepSea environment, from Osband et al. [24].

The difficulty of the exploration in DeepSea can be adjusted using N: the larger N, the harder it becomes for the agent to reach the goal location for optimal returns of 0.99. We evaluate all algorithms in the DeepSea task for $N \in \{10, 14, 20, 24, 30\}$.

Hallway is a new environment proposed as part of this work to represent domains where exploration and exploitation are misaligned, visualised in Figure 3. In DeepSea, agents receive reward by reaching states at the end of the environment, so intrinsic rewards for exploration strongly align with extrinsic rewards from the environment. We hypothesise that tasks in which intrinsic and extrinsic rewards are not well aligned require carefully balanced exploration through intrinsic rewards. Motivated by this hypothesis, we design the Hallway environment in which an agent is located in a hallway starting on the left. A goal can be reached by moving N_l cells to the right. In contrast to DeepSea, the goal is not necessarily located at the right end of the hallway, but there might be further N_r empty cells to the right of the goal location. At each timestep, the agent can choose between three actions: move left, stay or move right. The agent receives a reward of +1 for reaching the goal for the first time and every time it stays at the goal location for 10 steps. Therefore, the agent needs to learn to move to the goal and stay there for the remaining timesteps of the episode to collect further reward. Episodes end after $2N_l$ steps and small negative reward of -0.01 is assigned for moving right or stay. Hallway tasks, in particular with $N_r > 0$, require exploration through intrinsic rewards to be carefully balanced because staying at the goal for optimal returns and exploration are not aligned. We evaluate all algorithms in the Hallway environment with $N_l \in \{10, 20, 30\}$ and N_r either being 0 or equal to N_l.

5.4 Implementation Details

We compute n-step returns [37] to reduce the bias of value estimates in all algorithms. On-policy training uses four parallel, synchronous environments and an additional entropy regularisation term in the policy loss [21]. Double-DQN [40] targets are computed for DQN. For details on the conducted hyperparameter search as well as all values used throughout experiments, see Appendix A[†].

We train all algorithms for 100,000 episodes and evaluate every 1,000 episodes for a total of 100 evaluations by applying the greedy (evaluation) policy in the respective task for 8 episodes. Following recent suggestions for evaluation in deep RL [1], we report averaged evaluation returns and stratified bootstrap 95% confidence intervals using 5,000 samples for the bootstrapping across five random seeds. Optimal returns are indicated using a dashed horizontal line. A weighting factor of $\lambda = 1$ is used for the combined reward signal unless stated otherwise.

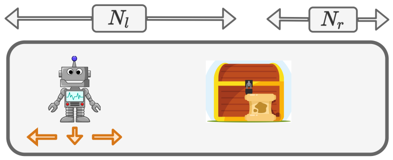


Figure 3: Hallway environment.

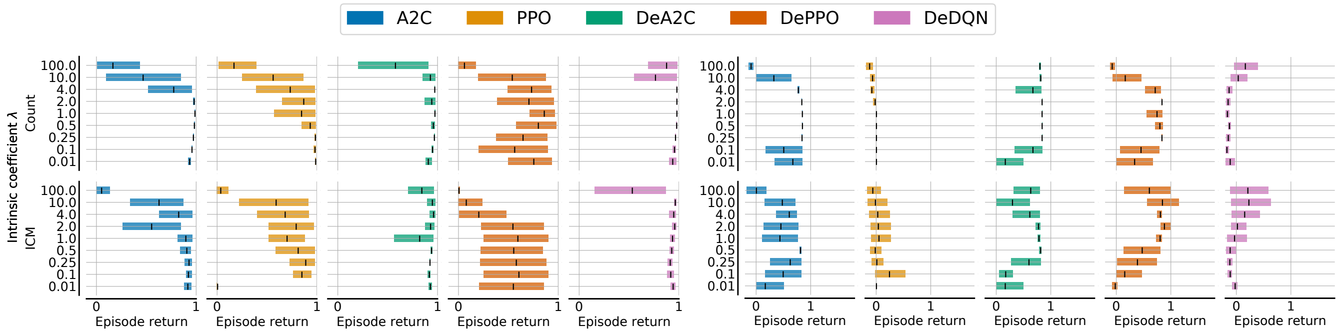
5.5 Hyperparameter Sensitivity

To investigate our first two hypotheses, we train all baselines and DeRL algorithms on the combined reward $r = r^e + \lambda r^i$ in DeepSea $N = 10$ and Hallway $N_l = N_r = 10$ with varying λ and rates of decay. We confirm our first hypothesis that intrinsically-motivated RL is indeed highly sensitive to scale and decay rate of intrinsic rewards, not learning at all or reaching significantly lower evaluation returns for many hyperparameter values. In particular in the Hallway environment where exploration and extrinsic rewards are misaligned, all algorithms exhibit significant dependency on carefully tuned scale and rate of decay. We further confirm our second hypothesis that DeRL algorithms are more robust to varying scale and decay rate of intrinsic rewards, reaching higher returns across a wider range of hyperparameters. Figures 4 and 5 show sensitivity of all baselines and DeRL algorithms with Count and ICM. Sensitivity analysis for all remaining intrinsic rewards can be found in Appendix C.

Scale of intrinsic rewards: We consider $\lambda \in \{0.01, 0.1, 0.25, 0.5, 1.0, 2.0, 4.0, 10.0, 100.0\}$ to analyse the sensitivity to varying scale of intrinsic rewards. Figure 4 shows average evaluation returns for all values of λ for baselines and DeRL. Average returns and bootstrap confidence intervals are computed across five seeds before the average over all 100 evaluations is computed. In DeepSea $N = 10$, DeA2C and DeDQN exhibit improved robustness by reaching close to optimal returns for almost all values of λ. In contrast, DePPO and the baselines are found to be more sensitive in particular to large values of λ. In Hallway, all algorithms exhibit larger variance for varying λ compared to DeepSea with no significant learning being observed for large or small values of λ, with DeA2C and DePPO demonstrating slightly more robustness. These results indicate the sensitivity to values of λ. Even small deviations can make the difference between learning and not learning at all.

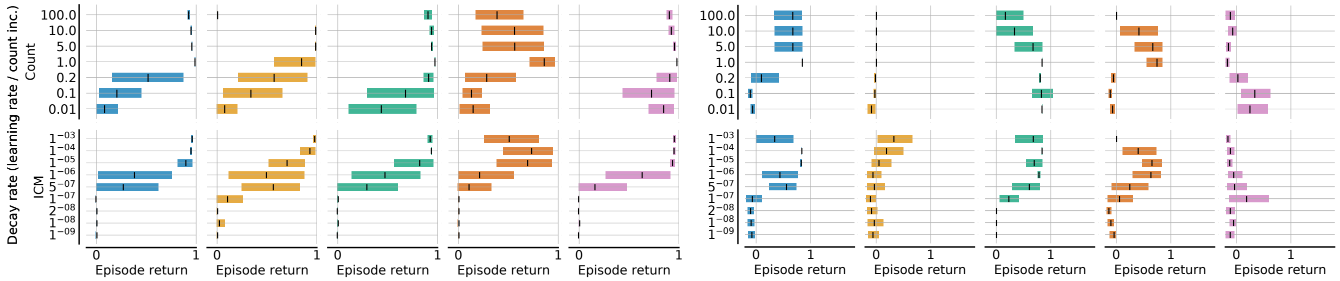
Decay of intrinsic rewards: We also investigate the sensitivity of intrinsically-motivated baselines and DeRL algorithms to the rate of decay of intrinsic rewards. For count-based intrinsic rewards, the rate of decay can be determined by the increment of the state count $N(s)$. For a sensitivity analysis, we consider increments $\{0.01, 0.1, 0.2, 1.0, 5.0, 10.0, 100.0\}$. For deep prediction-based intrinsic rewards, we consider learning rates $\{1e^{-9}, 1e^{-8}, 2e^{-8}, 1e^{-7}, 5e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}\}$ determining the rate of decay. Figure 5 shows average evaluation returns of baselines and DeRL with varying rates of decay in both DeepSea $N = 10$ and Hallway $N_l = N_r = 10$. A2C

[†] All appendices are available online at https://arxiv.org/pdf/2107.08966.pdf.



(a) DeepSea 10 **(b) Hallway $N_l = N_r = 10$**

Figure 4: Average evaluation returns in DeepSea 10 and Hallway $N_l = N_r = 10$ with $\lambda \in \{0.01, 0.1, 0.25, 0.5, 1.0, 2.0, 4.0, 10.0, 100.0\}$. Shading indicates 95% confidence intervals. A method that is insentive to hyperparameters will have final average episodic return concentrated to the right for all hyperparameter values.



(a) DeepSea 10 **(b) Hallway $N_l = N_r = 10$**

Figure 5: Average evaluation returns in DeepSea 10 and Hallway $N_l = N_r = 10$ with varying rates of decay. Shading indicates 95% confidence intervals.

is shown to be more robust to varying rates of decay in both environments compared to PPO. DePPO demonstrates larger sensitivity compared to A2C, but DeA2C and DeDQN are again shown to be the most robust algorithms, especially with Count intrinsic rewards, exhibiting high evaluation returns for most considered values in DeepSea $N = 10$. Similar to λ sensitivity, we observe very significant dependency on the rate of decay in the Hallway task with DeA2C exhibiting improved robustness to varying values.

5.6 Evaluation Returns

Lastly, we report evaluation returns of all algorithms across all DeepSea and Hallway tasks in Table 1. Average returns and standard deviations are computed across all 100 evaluations after being averaged across five seeds to indicate achieved returns as well as sample efficiency. Additionally, we present normalised returns with 95% confidence intervals across both environments in Figure 6, and tables with maximum achieved evaluation returns at any evaluation and learning curves for each individual task in Appendix B.

In DeepSea, DeDQN performs best out of all algorithms (Figure 6). DeDQN converges to returns comparable to or higher than the best performing baselines exhibiting highest average evaluation returns in all tasks but DeepSea 20. DeA2C and DePPO demonstrate similar returns and sample efficiency in some of these tasks (see Figures 9a and 9b). In DeepSea 24 (Figure 9d) and harder Hallway tasks with $N_l = 20, N_r = 0$ and $N_l = N_r = 30$ (Figures 11b and 11f), the exploitation policies of DeA2C and DePPO converge to the highest returns and are shown to be more sample efficient reaching high returns after up to 40% fewer episodes of training compared to the best performing baselines. Generally, we can see that DeA2C learns the optimal policy in the majority of Hallway tasks for some of the five executed runs, but fails to converge to such behaviour consistently. Instead, the majority of baselines and some DeRL runs learn to reach the goal but move back and forth between the goal and its left neighboured cell. Presumably, consistently staying at the goal is rarely discovered due to the small negative reward of staying at a cell.

However, we also observe some failure cases for DeRL algorithms. DeDQN achieves low returns in the Hallway environment compared to both on-policy DeA2C and DePPO. Also, significant variance can be observed for baselines and DeRL algorithms in harder DeepSea and most Hallway tasks. Off-policy optimisation is theoretically independent of the policy generating training samples, and in DeA2C and DePPO IS weight correction is applied to correct for the off-policy training data. However, we believe distribution shift [12] is causing inconsistent returns when optimising the exploitation policy from data generated by π_β. Figure 7 visualises

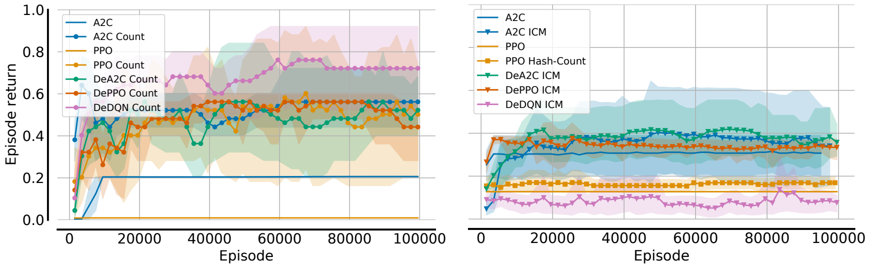


Figure 6: Normalised evaluation returns for DeepSea (left) and Hallway (right). Returns for each task are normalised to be within [0, 1] before averaged returns and 95% confidence intervals are computed across all tasks and five random seeds.

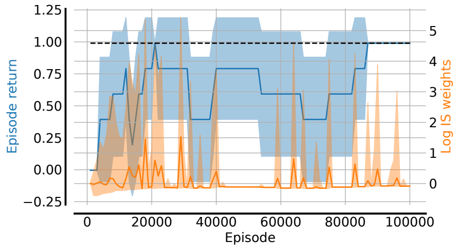


Figure 7: Evaluation returns and log IS weights for DeA2C Count in DeepSea $N = 14$.

Table 1: Average evaluation returns and a single standard deviation in all DeepSea and Hallway tasks over 100,000 episodes. The highest achieved returns in each task are highlighted in bold together with all returns within a single standard deviation. For DeRL algorithms, evaluations are executed using the exploitation policy.

Alg	DeepSea 10	DeepSea 14	DeepSea 20	DeepSea 24	DeepSea 30	Hallway 10-0	Hallway 10-10	Hallway 20-0	Hallway 20-20	Hallway 30-0	Hallway 30-30
A2C	**0.93 ± 0.22**	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.67 ± 0.05	0.49 ± 0.09	0.42 ± 0.02	0.50 ± 0.03	0.28 ± 0.08	0.42 ± 0.08
A2C Count	**0.98 ± 0.07**	**0.94 ± 0.16**	**0.74 ± 0.10**	0.11 ± 0.15	−0.01 ± 0.00	**0.85 ± 0.01**	**0.85 ± 0.02**	0.61 ± 0.03	**0.55 ± 0.06**	−0.33 ± 0.15	−0.06 ± 0.07
A2C Hash-Count	**0.98 ± 0.07**	**0.96 ± 0.15**	0.39 ± 0.14	**0.53 ± 0.12**	−0.01 ± 0.00	**0.85 ± 0.01**	**0.85 ± 0.03**	0.56 ± 0.03	**0.55 ± 0.06**	−0.34 ± 0.15	−0.13 ± 0.11
A2C ICM	0.87 ± 0.20	0.69 ± 0.31	0.54 ± 0.23	**0.46 ± 0.30**	**0.08 ± 0.12**	0.62 ± 0.17	0.57 ± 0.17	0.27 ± 0.12	**0.78 ± 0.27**	**1.16 ± 0.47**	**0.64 ± 0.38**
A2C RND	0.06 ± 0.01	0.19 ± 0.02	−0.01 ± 0.00	−0.01 ± 0.00	−0.01 ± 0.00	−0.12 ± 0.02	−0.07 ± 0.00	−0.20 ± 0.01	−0.24 ± 0.00	−0.24 ± 0.01	−0.12 ± 0.00
A2C RIDE	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	**0.85 ± 0.04**	**0.85 ± 0.02**	**0.70 ± 0.00**	**0.62 ± 0.00**	0.37 ± 0.04	0.28 ± 0.08
PPO	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
PPO Count	0.84 ± 0.10	0.70 ± 0.17	0.46 ± 0.19	0.17 ± 0.18	**0.20 ± 0.15**	0.00 ± 0.01	0.00 ± 0.00	0.00 ± 0.01	0.00 ± 0.01	0.00 ± 0.00	0.00 ± 0.01
PPO Hash-Count	0.86 ± 0.08	0.77 ± 0.13	0.34 ± 0.14	0.28 ± 0.20	**0.12 ± 0.13**	0.39 ± 0.11	0.10 ± 0.07	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
PPO ICM	0.84 ± 0.17	0.28 ± 0.17	0.00 ± 0.03	0.12 ± 0.17	0.00 ± 0.03	0.05 ± 0.15	0.11 ± 0.15	0.02 ± 0.16	0.08 ± 0.19	−0.04 ± 0.08	−0.02 ± 0.14
PPO RND	0.26 ± 0.12	0.15 ± 0.08	−0.01 ± 0.00	0.00 ± 0.00	−0.01 ± 0.00	−0.04 ± 0.04	−0.04 ± 0.11	−0.21 ± 0.06	−0.17 ± 0.09	−0.27 ± 0.10	−0.27 ± 0.11
PPO RIDE	0.73 ± 0.08	0.00 ± 0.00	0.00 ± 0.02	−0.01 ± 0.00	−0.01 ± 0.00	−0.10 ± 0.03	0.02 ± 0.08	−0.21 ± 0.03	−0.08 ± 0.08	−0.32 ± 0.04	−0.29 ± 0.08
DeA2C Count	**0.98 ± 0.10**	0.65 ± 0.23	0.42 ± 0.16	0.07 ± 0.10	**0.09 ± 0.08**	**0.84 ± 0.07**	**0.84 ± 0.09**	0.42 ± 0.02	**0.70 ± 0.01**	0.55 ± 0.00	0.22 ± 0.02
DeA2C ICM	0.86 ± 0.19	0.52 ± 0.28	0.27 ± 0.24	0.08 ± 0.14	**0.05 ± 0.11**	0.77 ± 0.18	0.80 ± 0.17	0.44 ± 0.15	**0.53 ± 0.20**	0.52 ± 0.34	**0.97 ± 0.51**
DePPO Count	0.61 ± 0.20	**0.92 ± 0.18**	−0.01 ± 0.01	**0.63 ± 0.27**	−0.01 ± 0.00	0.73 ± 0.10	0.80 ± 0.08	0.56 ± 0.01	**0.55 ± 0.04**	−0.20 ± 0.17	−0.06 ± 0.07
DePPO ICM	0.61 ± 0.18	0.37 ± 0.17	0.00 ± 0.01	−0.01 ± 0.00	0.00 ± 0.00	0.82 ± 0.11	0.81 ± 0.11	0.64 ± 0.16	**0.57 ± 0.07**	−0.01 ± 0.25	0.26 ± 0.06
DeDQN Count	**0.98 ± 0.09**	**0.95 ± 0.17**	0.40 ± 0.08	**0.53 ± 0.27**	**0.10 ± 0.10**	−0.13 ± 0.04	−0.15 ± 0.04	−0.05 ± 0.05	−0.12 ± 0.08	−0.17 ± 0.07	−0.10 ± 0.06
DeDQN ICM	**0.94 ± 0.20**	0.59 ± 0.40	0.16 ± 0.12	0.24 ± 0.25	**0.05 ± 0.12**	−0.09 ± 0.09	0.02 ± 0.16	−0.11 ± 0.09	−0.19 ± 0.08	−0.26 ± 0.08	−0.19 ± 0.08

unstable IS weights for DeA2C in the DeepSea task with $N = 14$ averaged over five seeds. These appear to correlate with some of the noticeable drops in returns throughout training, indicating the negative impact of divergence of exploration and exploitation policies on RL training of π_e. Even when applying Retrace(λ) [23] to clip IS weights, similar results are observed.

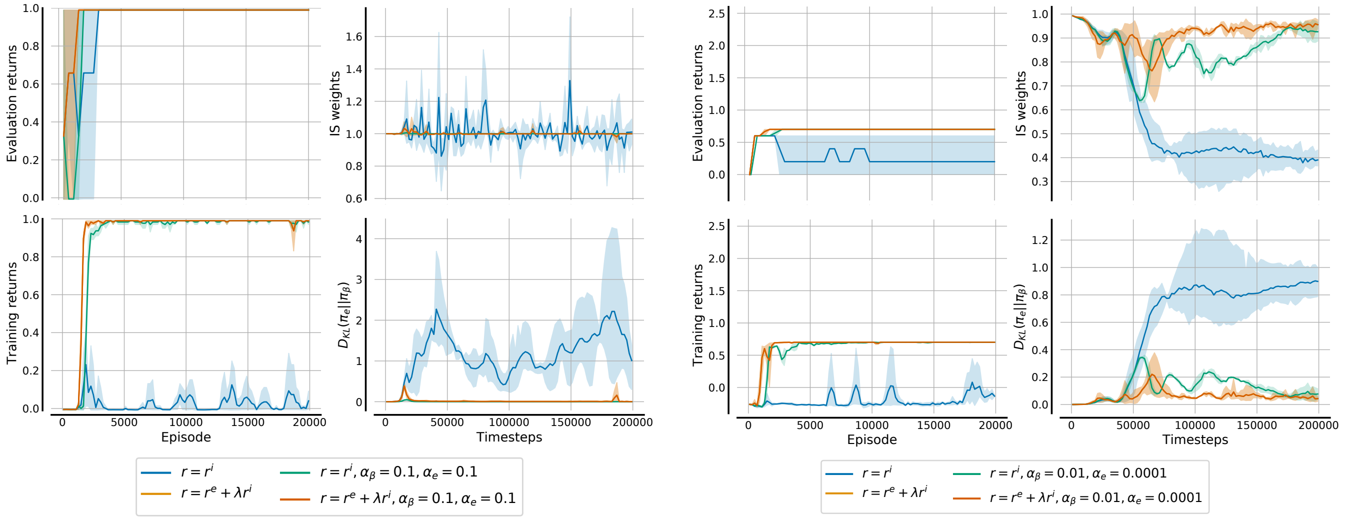
6 EXPLORATION USING ONLY INTRINSIC REWARDS

Prior work on intrinsic rewards for exploration investigated the effectiveness of training using only intrinsic rewards and no extrinsic rewards from the environment [6, 11, 30, 31]. Motivated by these experiments, we also investigate the possibility of optimising the exploration policy π_β using only intrinsic rewards. Such optimisation would likely lead to increased robustness to hyperparameters of intrinsic rewards as they would not be combined with extrinsic rewards of the environment. However, we also find that the optimisation of π_β without extrinsic rewards causes further divergence of π_β and π_e. It should be noted that the evaluation policy is still trained using extrinsic rewards.

We conduct experiments in the DeepSea 10 and Hallway $N_l = N_r = 20$ tasks training DeA2C with Count intrinsic rewards for 20,000 episodes. Results are averaged across three seeds and we directly compare optimising π_β using either the sum of extrinsic and intrinsic rewards (orange) or only using intrinsic rewards (blue) in Figure 8. We find that training the exploration policy with only intrinsic rewards does lead to increased divergence of both policies seen in IS weights (top right) and the KL divergence (bottom right). Training of the exploitation policy appears to suffer from such differences in the more challenging Hallway task, but in DeepSea the exploitation policy was successfully trained to solve the task despite the exploration policy never reaching high returns. Our results show the feasibility of training π_β using only intrinsic rewards but also the challenge of increased distribution shift.

7 DIVERGENCE CONSTRAINTS

In order to address distribution shift caused by diverging exploration and exploitation policies, we investigate the application of divergence constraints proposed in the literature of offline RL [16]. These auxiliary objectives are introduced to the optimisation and enforce π_e and π_β to not diverge significantly by introducing a term $\alpha D(\pi_e, \pi_\beta)$ to the optimisation loss. This term is based on a distance



(a) DeepSea $N = 10$

(b) Hallway $N_l = N_r = 20$

Figure 8: Evaluation and training returns, IS weights and KL-divergence of exploration and exploitation policy for training of DeA2C with Count in (a) DeepSea 10 and (b) Hallway $N_l = N_r = 20$ with intrinsic and extrinsic rewards or only intrinsic rewards as training signal for π_β, and with KL-divergence constraints with coefficients α_β and α_e. Evaluation and training returns are achieved using the exploitation and exploration policies, respectively. Shading indicates 95% confidence intervals.

measure D between the distribution of policies π_β and π_e and some weighting hyperparameter α. A common distance measure is the Kullback-Leibler (KL) divergence, which has been applied in offline RL [15], which can be written as following.

$$D_{KL}(\pi_e(s_t), \pi_\beta(s_t)) = \mathbb{E}_{a \sim \pi_e(\cdot|S)} \left[\log \pi_e(a|s_t) - \log \pi_\beta(a|s_t) \right] \quad (15)$$

For more distance measure candidates between two policies, see Wu et al. [43] which found these metrics to perform comparably.

In DeRL, divergence constraints can be directly applied to the optimisation of either the exploration π_β or exploitation policy π_e, i.e. can choose to keep π_β close to π_e and likewise can enforce π_e to stay close to π_β. We consider either of these directions as well as a combination of both constraints.

We evaluate the application of KL constraints as regularisers in the policy loss of the exploration policy, $\alpha_\beta D_{KL}(\pi_\beta, \pi_e)$, and exploitation policy, $\alpha_e D_e(\pi_e, \pi_\beta)$, with varying weights α_β and α_e, respectively. These constraints are applied in both settings introduced in Section 6 with π_β being optimised using only intrinsic (green) or intrinsic and extrinsic rewards (red) with results shown in Figure 8 for selected constraint coefficients. We find KL divergence constraints successfully address distribution shift and thereby keep both policies close to each other, even if π_β is only trained using intrinsic rewards. Such minimised divergence also leads to reduced variability of returns in both tasks. These results indicate the feasibility of training π_β using only intrinsic rewards and the effectiveness of divergence constraints to minimise distribution shift. We further evaluate the sensitivity of DeA2C with KL divergence constraints and show that such regularisation can improve robustness. For figures showing distribution shift, training and evaluation returns for a range of KL constraint coefficients, α_β and α_e as well the conducted sensitivity analysis, see Appendix D.

8 CONCLUSION

In this work, we proposed Decoupled RL (DeRL) which decouples exploration and exploitation into two separate policies. DeRL optimises the exploration policy with additional intrinsic rewards to incentivise exploration and trains the exploitation policy using only extrinsic rewards from data generated by the exploration policy. Based on this general framework, we formulate Decoupled Actor-Critic and Decoupled Deep Q-Networks and evaluated in two sparse-reward environments. Our results demonstrate that intrinsically-motivated RL is highly dependent on careful hyperparameter tuning of intrinsic rewards, indicating the need for more robust solutions. We show that decoupling exploration and exploitation is possible and does lead to significant benefits in robustness to varying scale and rate of decay of intrinsic rewards. Furthermore, we identify distribution shift as a challenge in separating the RL optimisation into two policies with separate optimisation objectives and investigate the application of divergence constraints to minimise such divergence of both policies. Our results demonstrate the effectiveness of divergence constraint regularisation and indicate improved sample efficiency of DeRL in some tasks by reaching high returns in fewer interactions in the environment. Lastly, we demonstrated the feasibility of training the exploration policy using only intrinsic rewards. Alongside divergence constraints, such a training setting seems a promising directions for further research into decoupled exploitation and exploration.

REFERENCES

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. 2021. Deep reinforcement learning at the edge of the statistical precipice. In *Advances on Neural Information Processing Systems*.

[2] Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51, 1 (2008),

117.

[3] Andrew G Barto. 2013. Intrinsic motivation and reinforcement learning. In *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer, 17–47.

[4] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances on Neural Information Processing Systems*. 1471–1479.

[5] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. 2018. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355* (2018).

[6] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*.

[7] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. 2005. Intrinsically motivated reinforcement learning. In *Advances on Neural Information Processing Systems*. 1281–1288.

[8] Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 33. 10707–10717.

[9] Thomas Degris, Martha White, and Richard S Sutton. 2012. Off-policy actor-critic. In *International Conference on Machine Learning*.

[10] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*. PMLR, 1407–1416.

[11] Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. 2021. Adversarially Guided Actor-Critic. In *International Conference on Learning Representations*.

[12] Scott Fujimoto, Herke Van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*. PMLR, 1587–1596.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, 1861–1870.

[14] Ronald A Howard. 1964. Dynamic programming and Markov processes. (1964).

[15] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456* (2019).

[16] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

[18] Long-Ji Lin. 1992. *Reinforcement learning for robots using neural networks*. Carnegie Mellon University.

[19] Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. 2021. Decoupling Exploration and Exploitation for Meta-Reinforcement Learning without Sacrifices. In *International Conference on Machine Learning*. PMLR, 6925–6935.

[20] Iou-Jen Liu, Unnat Jain, Raymond A Yeh, and Alexander Schwing. 2021. Cooperative exploration for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 6826–6836.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. 1928–1937.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[23] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G Bellemare. 2016. Safe and efficient off-policy reinforcement learning. In *Advances on Neural Information Processing Systems*, Vol. 29.

[24] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, et al. 2020. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*.

[25] Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. 2017. Count-based exploration with neural density models. In *International Conference on Machine Learning*. JMLR, 2721–2730.

[26] Pierre-Yves Oudeyer and Frederic Kaplan. 2009. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics* 1 (2009).

[27] Pierre-Yves Oudeyer, Frederic Kaplan, et al. 2008. How can we define intrinsic motivation. In *Conference on Epigenetic Robotics*, Vol. 5. 29–31.

[28] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. 2007. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on Evolutionary Computation* 11 (2007), 265–286.

[29] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Advances in Neural Information Processing Systems, Track on Datasets and Benchmarks*.

[30] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*.

[31] Roberta Raileanu and Tim Rocktäschel. 2020. RIDE: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*.

[32] Tabish Rashid, Bei Peng, Wendelin Böhmer, and Shimon Whiteson. 2020. Optimistic Exploration even with a Pessimistic Initialisation. In *International Conference on Learning Representations*.

[33] Jürgen Schmidhuber. 1991. Curious model-building control systems. *IEEE International Joint Conference on Neural Networks* 2 (1991), 1458–1463.

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[35] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*. PMLR, 387–395.

[36] Alexander L Strehl and Michael L Littman. 2008. An analysis of model-based interval estimation for Markov decision processes. *J. Comput. System Sci.* 74 (2008), 1309–1331.

[37] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[38] Adrien Ali Taïga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. 2019. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388* (2019).

[39] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. # Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances on Neural Information Processing Systems*. 2753–2762.

[40] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI Conference on Artificial Intelligence*. AAAI Press, 2094–2100.

[41] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. King's College, Cambridge.

[42] William F Whitney, Michael Bloesch, Jost Tobias Springenberg, Abbas Abdolmaleki, and Martin Riedmiller. 2021. Decoupled Exploration and Exploitation Policies for Sample-Efficient Reinforcement Learning. *arXiv preprint arXiv:2101.09458* (2021).

[43] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[44] Rujie Zhong, Josiah P. Hanna, Lukas Schäfer, and Stefano V. Albrecht. 2021. Robust On-Policy Data Collection for Data-Efficient Policy Evaluation. In *Offline Reinforcement Learning Workshop at Neural Information Processing Systems Conference*.