Using Offline Data to Speed-up Reinforcement Learning in Procedurally Generated Environments

Alain Andres
University of the Basque Country, TECNALIA
San Sebastian, Spain
alain.andres@tecnalia.com

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

Esther Villar-Rodriguez
TECNALIA
Bilbao, Spain
esther.villar@tecnalia.com

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

Javier Del Ser
TECNALIA
Bilbao, Spain
javier.delser@tecnalia.com

ABSTRACT

One of the key challenges of *Reinforcement Learning* (RL) is the ability of agents to generalise their learned policy to unseen settings. Moreover, training RL agents requires large numbers of interactions with the environment. Motivated by the recent success of *Offline RL* and *Imitation Learning* (IL), we conduct a study to investigate whether agents can leverage offline data in the form of trajectories to improve the sample-efficiency in procedurally generated environments. We consider two settings of using IL from offline data for RL: (1) pre-training a policy before online RL training and (2) concurrently training a policy with online RL and IL from offline data. We analyse the impact of the quality (optimality of trajectories) and diversity (number of trajectories and covered level) of available offline trajectories on the effectiveness of both approaches. Across four well-known sparse reward tasks in the MiniGrid environment, we find that using IL for pre-training and concurrently during online RL training both consistently improve the sample-efficiency while converging to optimal policies. Furthermore, we show that pre-training a policy from as few as two trajectories can make the difference between learning an optimal policy at the end of online training and not learning at all. Our findings motivate the widespread adoption of IL for pre-training and concurrent IL in procedurally generated environments whenever offline trajectories are available or can be generated.

KEYWORDS

Imitation Learning, Reinforcement Learning, Generalisation, Diversity

1 INTRODUCTION

The *Reinforcement Learning* (RL) paradigm is widely used for sequential decision making in various fields, including healthcare [11], energy [10] and robotics [53]. Traditionally, RL algorithms are trained and evaluated in the same single task, with the goal of maximising the cumulative reward over time. However, the variability of real-world problems poses a challenge for these agents, as they may not generalise well to new (unseen) scenarios [41]. To address this issue, recent research in RL has shifted its focus towards the ability of agents to generalise to varying but similar tasks that can differ in either the state space, dynamics of the environment, agent's action space and even the reward function [21]. One way of evaluating the generalisation capability of agents is by training agents in procedurally-content-generated (PCG) environments. Any PCG task constitutes a set of levels over which the learned policy has to generalise. Completing the levels of a single task requires a common skill, but may, for example, vary in the agent's initial location, the layout of its environment, colours and locations of objects the agent can interact with. Such variability prevents the agent from memorising specific trajectories (overfitting) [4]; instead, PCG environments force the agent to learn relevant representations and policies which effectively generalise across all levels of a task.

However, PCG environments often require large amounts of interactions to train an effective policy [18]. In this work, we propose to use offline data to speed-up the learning of an agent in PCG environments. This is motivated by the availability of such data in real-world settings, where one of the main objectives is to decrease the number of agent-environment interactions due to economic, safety and time constraints. The main contribution of our work is to study the effectiveness of using offline data to improve the converged performance and sample-efficiency of RL agents in PCG environments. The contributions of our study are threefold. Firstly, we analyse how offline data can be used to pre-train a policy to kick-start the learning of an agent. Secondly, we study how offline data can be combined with the online collected experiences for online *Imitation Learning* (IL) to make the RL agent's training more sample efficient. Thirdly, we investigate how the quality and quantity of the provided offline data affects the learning process.

We collect a dataset of offline trajectories by training an agent with a self-imitation-learning approach specifically designed for PCG environments (RAPID [50]) and storing the best trajectories seen so far during training at three checkpoints during training. Each of these three datasets contains trajectories of varying quality as measured by the performance of the policy at that point during training. The offline data is used via IL, more concretely with *Behaviour Cloning* (BC) before and concurrently to the online RL training. Finally, we analyse the results in various MultiRoom and ObstructedMaze scenarios of the PCG MiniGrid [3] benchmark.

Our results show that using offline data in any of the analysed tasks significantly reduces the amount of interactions required to

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 9-10, 2023, Online, https://ala2023.github.io/. 2023.

learn an optimal policy. In fact, pre-training provides a good initialisation policy to kickstart learning. Moreover, training the agent with IL concurrently to online RL training further improves robustness and sample efficiency, obtaining the optimal policy with substantially less agent-environment interactions. Lastly, we empirically show how these results can be achieved even in a low data regime while emphasising the importance of the diversity of selected trajectories over their quality, which poses the potential of kickstarting an RL agent capable of generalising well with just a handful of trajectories.

2 RELATED WORK

Sample-efficiency in Procedurally Generated Environments. Off-policy algorithms are naturally suitable to make use of data collected by an arbitrary behaviour policy, and are more sample efficient in the number of agent-environment interactions due to the application of a replay buffer [6]. However, they exhibit larger instabilities and are more sensitive to hyperparameters than on-policy solutions [12]. These issues are further exacerbated in PCG environments [7], where comparably little research exists using off-policy (e.g. DQN [29], SAC [15]) algorithms in comparison with on-policy algorithms (e.g. PPO [43], IMPALA [8], PPG [5]). In fact, off-policy algorithms have only been applied to solve tasks that are comparably easily solved by on-policy algorithms [20, 32, 44]. Therefore, a large amount of algorithmic approaches have been focused on how to improve the sample-efficiency of on-policy algorithms by incentivising exploration with either intrinsic rewards that model the curiosity [9, 39, 42, 52] or using self-imitation-learning techniques [1, 50] which augment online RL training with BC from previously collected trajectories.

Offline Data for Reinforcement Learning. *Offline RL* is known as the paradigm of acquiring an effective policy by utilising only previously collected data (with no online interaction) [23]. In that context, the data collection can be supervised by a human in order to maximise the return [27]; or it can be done without supervision while trying to maximise the data coverage or the discovery of skills [25, 26]. However, such data can also be leveraged in the offline-to-online RL setting [48], in which offline data is used to pretrain a given policy and then finetune it during the online stage to reduce the number of required agent-environment interactions [28, 31, 45, 47, 51]. One of the simplest techniques to utilise offline data is through *Imitation Learning* (IL) [13, 27, 31, 40] which treats the learning process of a policy from the given data as a supervised learning problem. However, IL is sensitive to (1) the quality of the data as given by its optimality [23], and (2) the distribution shift between the provided offline data and the data encountered when deploying the trained policy in the environment [31, 51]. In particular the latter challenge of distribution shift is particularly prominent in PCG environments where the agent has to further generalise over a set of levels. Distribution shift can be minimised using prioritisation techniques [14, 49] or enforcing constraints on the learned policy [22, 24, 47] but these aforementioned works rely on off-policy RL solutions which have been shown to be ineffective in many PCG environments [4, 7, 30].

Our contribution. Although previous approaches combined *Imitation Learning* (offline data) with on-policy *Reinforcement Learning* (online data), we are not aware of any work that studies these techniques in PCG environments which require agents to generalise. Herein we empirically study the effectiveness of combining IL and RL when varying the offline demonstrations' quality, quantity, and diversity which are attributes that have strong influence on the expected benefits when using offline data[1], especially in PCG environments where generalisation is mandatory.

3 BACKGROUND

3.1 Partially Observable Markov Decision Process

We define a RL problem as a Markov Decision Process (MDP) given by a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow [0, 1]$ is the state-transition probability function, $\mathcal{R} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow \mathbb{R}$ represents the reward function, and $\gamma \in [0, 1)$ denotes the discount factor. At every time step t, the agent observes a state $s_t \in \mathcal{S}$ and selects an action a_t sampled from its policy $a_t \sim \pi(\cdot|s_t)$. Given the current state s_t and selected action a_t the environment transitions to a new state $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ and the agent receives a reward $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$. In partially observable environments where the agent might only observe a part of the state, the environment can be formalised as a Partially Observable Markov Decision Process (POMPD) [19]. A POMDP extends the MPD formalism to a 7-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{O}, \Omega\}$ where Ω represents the observation space and $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \rightarrow [0, 1]$ represents the observation function that maps a state and action to a distribution over observations. In a POMDP, the agent only receives observations $o_t \sim \mathcal{O}(s_t, a_t)$ based on the current state and selected action, and conditions its policy on the episodic history of observations.

3.2 Procedural Content Generation

In this work, we focus on procedurally generated environments which require agents to learn policies which generalise across a collection of levels that maximise (minimise) a given objective.

Formally, a task T is composed of a collection of different levels $l \in \mathcal{L}(T)$, where each level is considered a POMDP and $\mathcal{L}(T)$ represents the whole distribution of levels for task T. The levels are generated with a seed, ID or parameter vector that makes them differ from other levels with respect to their underlying $\mathcal{S}$ and Ω spaces [21]. The objective is to maximise the expected discounted returns over the whole level distribution $\mathbb{E}_{\mathcal{L}(T)}[\sum_{t=0}^{N} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})]$, where N is the episode length and $\mathcal{R}(s_t, a_t, s_{t+1})$ is the reward at time step t.

3.3 Imitation Learning

Imitation Learning can be applied in several ways. Herein we adopt *Behaviour Cloning* (BC) using the log loss surrogate function [35]:

$$L_{BC} = -\frac{1}{|B|} \sum_{(s,a)\sim B} ln(\pi(a|s)) \tag{1}$$

[1] Quantity and diversity are related with the distribution shift.

Table 1: Summary of the buffers collected for four MiniGrid tasks as stated in Section 4.1. We provide statistics of the quantity and diversity of the data as given by the total number of stored levels ($\#_{levels}$) and the mean number of trajectories per level ($\mu_{\tau/level}$) (the total number of stored experiences is given by the product of these quantities). The optimality of stored trajectories is given by their mean number of experiences ($\mu_{\{s,a\}}$) and returns ($\mu_{G(\tau)}$). The last rows correspond to the expected optimal returns ($\mathbb{E}^*[G(\tau)]$) and optimal number of steps ($\mathbb{E}^*[lenght(\tau)]$) required to solve these tasks where the expectation is over the entire level distribution of this task. Each of those buffers contain 10,000 experience tuples.

	O1Dlhb			O2Dlh			MN7S8			MN12S10		
	10%	60%	90%	10%	60%	90%	10%	60%	90%	10%	60%	90%
$\#_{levels}$: number of different levels	88	259	250	68	296	292	115	193	210	70	100	101
$\mu_{\tau/level}$: mean number of trajectories per level	1.01	1.03	2.18	1	1.07	2.39	1	1.02	1.13	1	1.04	1.26
$\mu_{\{s,a\}}$: mean number of experiences per trajectory	112.4	37.3	18.34	147.1	31.5	14.3	86.9	50.8	41.8	142.8	96.2	78.1
$\mu_{G(\tau)}$: mean return of trajectories	0.63	0.88	0.94	0.74	0.95	0.98	0.42	0.67	0.73	0.45	0.64	0.71
$\mathbb{E}^*[G(\tau)]$: expected optimal return for any level		0.92			0.95			0.67			0.65	
$\mathbb{E}^*[lenght(\tau)]$: expected optimal steps for any level		25.6			32			51.3			93.3	

where B is a batch of state action pairs (s, a) containing experiences to be imitated, and π denotes the policy that is being trained. Prior works combine this BC loss with the online RL loss for a single backpropagation and optimisation objective [16, 46], whereas we separate the optimisation for BC and RL [1, 50][2]. This allows us to control the number of optimisation steps and the learning frequency of each optimisation objective.

3.3.1 Self-Imitation Learning. When expert data is not available, the agent can be trained with self-imitation learning. This paradigm attempts to learn a policy based on past successful trajectories collected by the agent itself [1, 33, 50], so that the agent can improve its behaviour with actions that led to promising outcomes. RAPID [50] determines the success of a trajectory based on the following weighted score

$$S = w_0 \cdot S_{ext} + w_1 \cdot S_{local} + w_2 \cdot S_{global} \tag{2}$$

where S_{ext} refers to the returns of the episode, S_{local} represents the diversity of states within the episodes, and S_{global} represents the long-term exploration as given by state visitation counts [2, 36]. RAPID ranks trajectories based on their weighted score and stores the trajectories with highest scores in a replay buffer. Throughout training, a random batch of trajectories is sampled uniformly at random and the BC loss is minimised for the given samples.

4 METHODOLOGY

In this section, we outline our approach including the data collection and IL techniques applied for pre- and concurrent training.

4.1 Data Collection

Unlike in other IL works where expert demonstration are given to the agent, we initially train an agent until convergence[3] using RAPID [50], as outlined in Section 3.3.1. Each replay buffer, containing 10,000 experience tuples at maximum for a particular task, is stored as datasets of trajectories at three checkpoints throughout training. The three checkpoints for each scenario correspond to the first time the agent achieves a 0.1, 0.6 and 0.9 evaluation return in ObstructedMaze; and 0.06, 0.4 and 0.6 in MultiRoom. These returns correspond to approximately 10%, 60% and 90% of the expected optimal returns in those tasks. The replay buffers contain trajectories of varying quality, as given by the achieved evaluation returns, and diversity. We refer to Table 1 for further details about the quality, quantity and diversity of each of the datasets. Due to the total number of experience tuples being limited, the higher the number of steps per trajectory in a buffer, the lower the number of total trajectories stored in the buffer[4].

4.2 Learning from Offline Data

In this study, we consider two techniques to leverage offline data for the training of an RL agent: pre-training and concurrent IL.

Pre-training. For pre-training, prior to interacting with the environment, we sample uniformly at random from the selected dataset a batch of state-action tuples (independently of the trajectory that they belong to) and minimize the BC loss given in Eq 1. We complete a fixed number of such updates as given by the number of epochs. After pre-training, no more IL is applied and the agent is purely trained using standard RL while interacting with the environment.

Concurrent training. For concurrent training, both the IL and RL losses are utilised during online training. The RL agent's policy is randomly initialised and trained from online interactions with the environment as usual. In addition to the RL optimisation, we sample batches of experience from the selected replay buffer at regular intervals throughout training and minimise the BC loss for the given batch. In this work, we sample each of those batches right after every RL optimisation. Furthermore, during the online training phase, if an encountered trajectory has a higher score according to Eq 2 than other trajectories in the buffer, the buffer is updated with the new trajectory.

[2]This separation does not affect pre-training with IL because no RL updates are computed in that stage.

[3]The purpose of this agent is only to serve as a demonstrator to collect trajectories and is not leveraged in any other way for our study.

[4]For this reason, in environments where a decrease in the number of steps per trajectory implies a higher return, the 90% buffers that are expected to contain higher-quality trajectories (with less steps per trajectory) would have a larger number of trajectories in comparison to the 60% and 10% buffers.

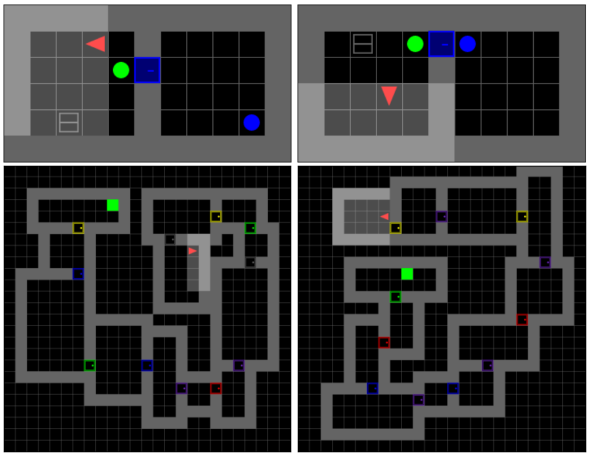

Figure 1: Two different levels of `O1Dlhb` (Top) and `MN12S10` (Bottom) tasks from the MiniGrid benchmark. In O1Dlhb, the agent (red triangle) has to move the ball, uncover the key under the box, pick up the key, open the door, discard the key and pick the blue ball; whereas in MN12S10, the agent has to go forward while opening the doors between rooms until reaching the green goal location. In all scenarios, the agent has only access to a partial observation of the environment as shown by the brighter area in front of the agent.

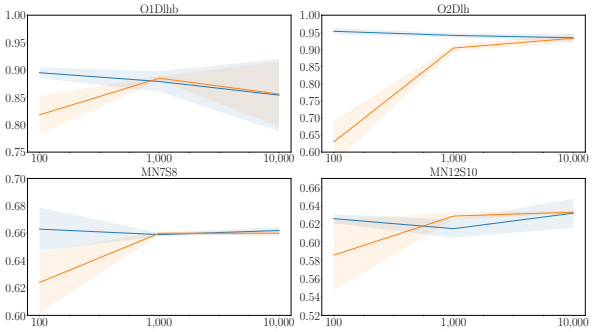

Figure 2: Illustration of the generalisation performance gap for evaluation in 1,000 held-out testing tasks of the level distribution (orange) and in a limited distribution of training levels (blue). For four MiniGrid environments, we show the final evaluation returns (y-axis) of an agent trained until convergence to the optimal policy using a state-of-the-art algorithm for PCG environments [1] as a function of the number of levels (x-axis) the agent is trained in. The mean and standard deviation is shown across 3 seeds. We can see that evaluation performance in 10,000 training levels accurately represents the expected testing performance.

5 EXPERIMENTAL SETUP

5.1 Research Questions

In order to understand how *Imitation Learning* impacts the described offline-to-online paradigm in PCG environments, we pose the following research questions:

(1) Does pre-training a RL agent with IL improve sample efficiency or converged performance? (RQ1)
(2) Can IL from offline trajectories be concurrently used to train an agent alongside online RL? (RQ2)
(3) How many levels and trajectories (correlated with the diversity of demonstrations) are needed for effective pre-training? How does the quality of demonstrations affect the pre-training? (RQ3)

5.2 Environment

We evaluate our results in multiple tasks of the MiniGrid environment [3] where the agent has to traverse a maze of varying layout to a goal location. The considered ObstructedMaze and MultiRoom tasks, as shown in Figure 1, require the agent to learn different skills. In these PCG environments the layout changes from level to level by modifying the agent's spawn location, orientation, the colour and objects with which the agent can interact with and also the final goal location. Hence, the optimal number of decisions/steps to solve each level is also different. Every level is generated with a seed and it can be modeled as a POMDP where the agent only perceives a fraction of the whole layout that surrounds it. Akin to other studies that provide information relating the generalisation performance gap for varying numbers of training levels [4], we analyse how many levels are sufficient to capture the agent's performance across the entire level distribution in MiniGrid tasks. Figure 2 shows that the performance across 10,000 levels accurately represents the performance of the agent across the entire level distribution. Hence, we train and evaluate the agent across 10,000 levels.

The considered tasks represent sparse reward problems (i.e., a non-zero reward is only provided if the agent reaches a goal location within the maze in a predefined number of steps) with a reward function that is defined as follows:

$$\mathcal{R}(s_t, a_t, s_{t+1}) = \begin{cases} 1 - 0.9 \cdot \frac{t}{t_{max}}, & \text{if}(t < t_{max} \& s_{t+1} \text{ is terminal}) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

with t_{max} being the maximum number of steps per episode that is dependent on the respective task (e.g., `O1Dlhb`: 288, `MN12S10`: 240).

5.3 Algorithms

Based on previous work demonstrating better performance of on-policy over off-policy algorithms in PCG environments [4, 7, 30], we use PPO [43] for the online RL training with hyperparameters specified in Table 2. We compare the results of agents with and without pre-training and concurrent IL with three baselines: pure online PPO, pure IL (train on the demonstrations provided in each buffer using BC) and RAPID [50](self-imitation-learning approach).

Regarding the neural network architecture, two independent actor and critic models are used. Both of them are composed by 2 fully connected layers with tanh activation functions. Note that the IL gradients are only applied through the actor network by forcing the agent to mimic the $\{s, a\}$ tuples provided in the demonstrations; the critic is not directly affected by IL. For every IL update, we uniformly sample a batch of 256 $\{s, a\}$-tuples from all the possible experiences at the buffer and updates the actor network for a total of 5 consecutive epochs with this batch.

Table 2: PPO Hyperparameters

Hyperparameter	Value
Optimiser	Adam
Learning Rate	10^{-4}
Adam epsilon	10^{-5}
Environment steps per update	2048
Discount γ	0.99
GAE λ	0.95
Entropy coefficient	0.01
Value loss coefficient	0.5
Number of epoch	4
Number of minibatches	4
PPO clipping constant	0.2
Max grad norm	0.5

6 EVALUATION RESULTS

All the provided plots report the mean and standard deviation of the average return over the past 100 episodes across 3 different seeds. As the results are evaluated in a PCG setting, the obtained training score is used to report the agent's performance.

6.1 RQ1. Pre-training with Imitation Learning

Figure 3 shows the performance of the agent when using IL for pre-training of the agent policy. We find that pre-training significantly improves sample efficiency and performance compared to pure RL (in yellow) and RAPID (in brown). Both baselines fail to learn at all or requires a larger number of interactions to attain the optimal policy. Moreover, an increase in quality of the demonstrations and number of imitation learning epochs increases the rate of convergence. Nevertheless, independently of the trajectories selected for imitation, the policy obtained by BC at the end of pre-training is unable to generalise well to the whole level distribution, as shown by the respective horizontal lines.

When using 3,000 epochs for pre-training, only the pre-training from the higher quality 60% and 90% buffers in `O1Dlhb` and with the 90% buffer in `MN12S10` was effective. By increasing the number of epochs to 10,000, also the agent trained from highly suboptimal buffers, given by the 10% and the 60% buffers in `O1Dlhb` and `MN12S10`, respectively, learn to solve the task. We note that the agent uses the same offline buffers and identical number of online interactions for this experiment; we only optimise the policy of the agent for a larger number of experience batches using IL.

Interestingly, the evaluation return obtained by the pre-trained policies are not a robust indicator for successful policies at the end of RL training. For example, in the MN12S10 task the agent pre-trained for 3k epochs from the 60% buffer achieves $\approx$ 0.2 returns after pre-training but quickly collapses to 0 returns during RL training. Moreover, in `O1Dlhb` the adoption of either the 60% or 90% buffers leads to almost the same kickstart in terms of performance to the agent (e.g., $\approx$0.4 return), yet the latter exhibits a faster convergence during the online phase. Notice that both buffers have similar quantity and diversity but they differ in the quality of the demonstrations, which explains why the 90% leverages better results. When the quality and the diversity of trajectories is decreased (i.e. 10% buffer, pink), the kickstart of the policy is less effective (e.g., $\approx$0.1 return) while still leading to significant benefits compared to the baselines.

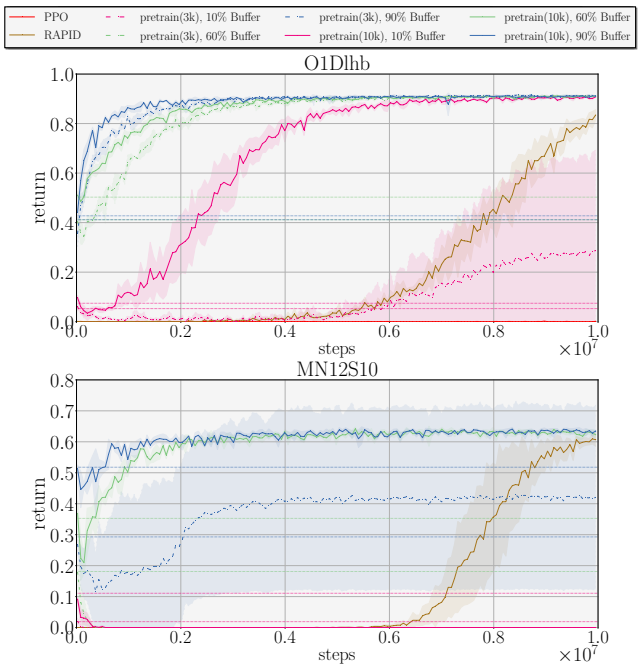

Figure 3: Performance of the agent when pre-training with IL before the RL training phase in `O1Dlhb` (Left) and `MN12S10` (Right). The horizontal dashed lines represent the pre-trained policies' evaluation score (trained solely with BC) that serve as initialisation point for the training phase. Depending the task and the demonstrations used, the employed number of pre-training updates (3k or 10k) affects more/less the performance. Notice the x-axis provides the number of interactions/steps of the agent (after the pre-training phase).

In contrast, in `MN12S10` all available buffers exhibit high diversity (see Table 1), but only the agent pre-trained from the 90% buffer (blue) is able to get non-failure results with 3,000 epochs. It appears that in this task, the quality of the trajectories plays a bigger role to attain a good initial policy than the diversity, failing when having a init policy that obtains less than $\approx$0.3 return ($\approx$ 186 steps), which is the case in the 60% buffer with 3,000 epochs. Nevertheless, when using both the 60% and 90% buffers with 10,000 epochs the agent has a initial return performance of 0.35 and 0.5 respectively, and it manages to learn the task consistently. In `O1Dlhb`, even when using the 10% buffer that corresponds with a $\approx$0.1 return ($\approx$ 272 steps) performance policy, the agent was able to learn an effective policy during RL training. Furthermore, all pre-training learning curves in `MN12S10` indicate a sharp drop in returns at the beginning of online RL training in comparison to the performance of the policy obtained through pre-training. We hypothesise that the reasons why this might happen is due to two phenomenons: (1) A strong shift between the level distribution of the demonstrations used in pre-training and the whole level distribution, and (2) misalignment between the IL and RL objectives (i.e., different gradient directions).

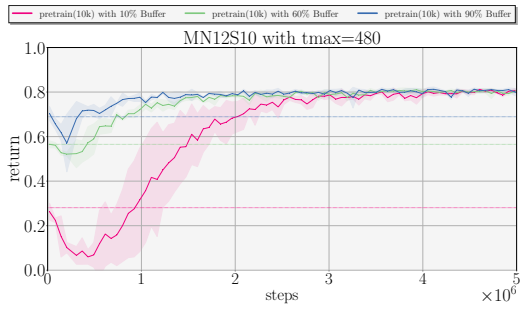

Figure 4: Performance of the agent in `MN12S10` when increasing t_{max} from 240 to 480 and using Imitation Learning during pre-training phase.

We highlight that the maximum number of steps for a single episode and expected optimal returns vary significantly across the `O1D1hb` and `MN12S10` tasks. In `O1D1hb` the agent is allowed to collect trajectories that are ×11.5 (i.e., 288/25 ≈ 11.5) longer (worse) than the trajectory of the optimal policy. In contrast, in `MN12S10` that gap is reduced to ×2.5 (i.e., 240/93 ≈ 2.5). Consequently, in the `MN12S10` task the agent has significantly less steps within an online episode to adapt its policy during the online training while still solving the task and thus receiving a positive reward signal. In order to evaluate our intuitions, we manually increment t_{max} in `MN12S10` from 240 steps to 480[5]. The curves in Figure 4 effectively show that an increase in maximum episode length allows the agent to act suboptimally while still receiving positive rewards and thereby avoids the previously observed failures for the 10% buffer (pink) while also preventing a drop in initial return when pre-trained with the 60% buffer (green).

6.2 RQ2. Concurrent Online Reinforcement Learning and Imitation Learning

In Figure 5 we compare the impact of concurrently training the agent with IL and RL during online training. We find that agents trained with concurrent IL manage to solve all the task and with all the analysed buffers even when pre-training exposed difficulties to learn (e.g. in `MN12S10` with the 10% buffer, pink). Thus, concurrent learning exhibits better robustness despite not having any prior knowledge at the beginning of the training phase. On the contrary, due to the significant jumpstart obtained by the pre-trained policies, the latter obtain a better sample-efficiency. These results are further improved by combining both pre-training with IL (in order to benefit from the kickstart and sample efficiency) and concurrent IL during online training (for robustness). Combining both of these approaches results in robust convergence to the optimal policies in both tasks in less number of online training steps.

6.3 RQ3. Sensitivity to number and diversity of demonstrations

Lastly, we analyse the sensitivity of using IL for pre-training in a low data regime where we train the policy with IL using a low number

[5]Note that this changes the expected optimal return from 0.65 shown in Table 1 to ≈ 0.82.

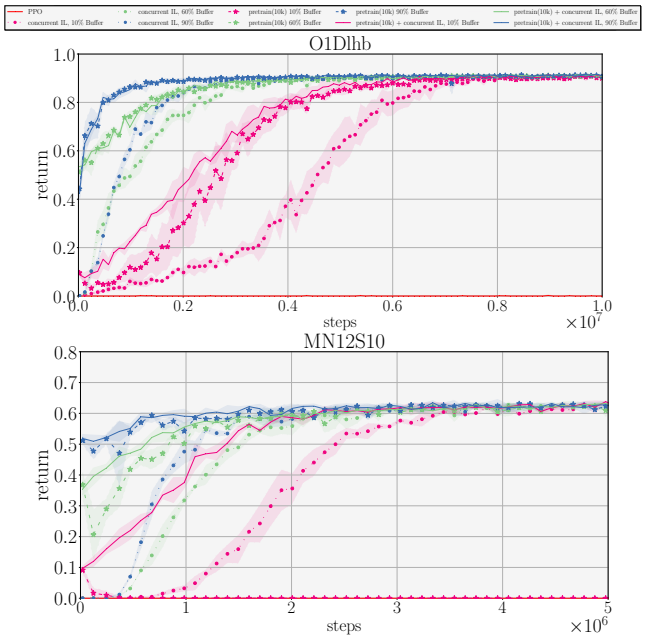

Figure 5: Performance of the agent when randomly initialising the policy and using both Imitation Learning and Reinforcement Learning losses online during the training in `O1D1hb`(Left) and `MN12S10`(Right). The obtained results are compared when IL is just used for pre-training (dashed lines). Within the worst demonstration setup (i.e., 10% Buffer), the best results are retrieved when IL is used at both pre-training and the main training phase.

of different levels (with a single trajectory per level). Moreover, we inspect how the trajectories belonging to optimal (90% buffer) or suboptimal (60% buffer) solutions significantly impact the IL. We show the following results of the RL training with varying pre-training in Figure 6.

Unexpectedly, the agent manages to effectively solve a large variety of environments when only provided with as low as 2 and up to 20 different trajectories. Pre-training with a larger number of trajectories positively impacts sample efficiency, but such benefits quickly diminish depending on the environment. For example for the `O1D1hb` task, very limited improvements in sample efficiency can be observed by having more than 5 levels. We note that online RL training with IL for pre-training in some cases fails to learn in `MN7S8` and `MN12S10`, which can potentially be addressed by increasing t_{max} as previously seen in Figure 4.

By vertically analysing the reported results in each scenario in Figure 6, it can be noticed that when using suboptimal demonstrations (60% Buffer) the agent is more robust to the reduction in the number of levels used; this is, the agent needs fewer numbers of trajectories during pre-training to learn an optimal policy at the end of the following online training. An example of that can be seen in MN12S10, where with the 90% buffer the agent only learns when using 10 or 20 levels, whereas with the 60% buffer the agent can learn with as few as 3 levels. We hypothesise that this occurs because of the specific levels stored (and consequently sampled) from

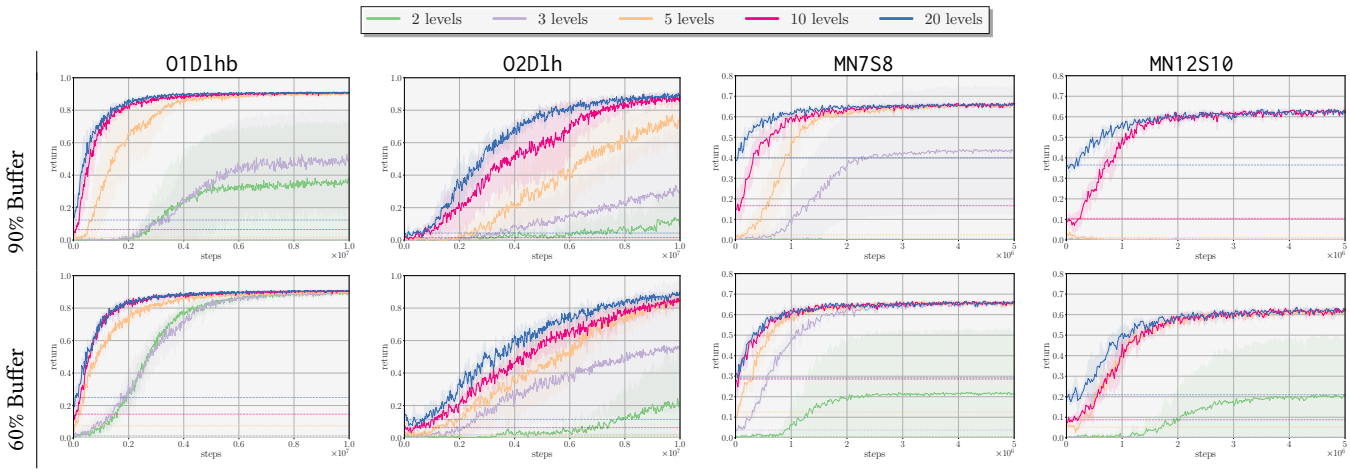

Figure 6: Agent performance when initialising the agent networks with the obtained policies during the IL pre-training phase with different fixed number of trajectories (one per level) that are considered optimal (top) or suboptimal (bottom). We provide the results, from left to right, for: `O1Dlhb`, `O2Dlh`, `MN7S8` and `MN12S10`. As in Figure 3, the dashed lines represent the BC pre-trained policies' evaluation score.

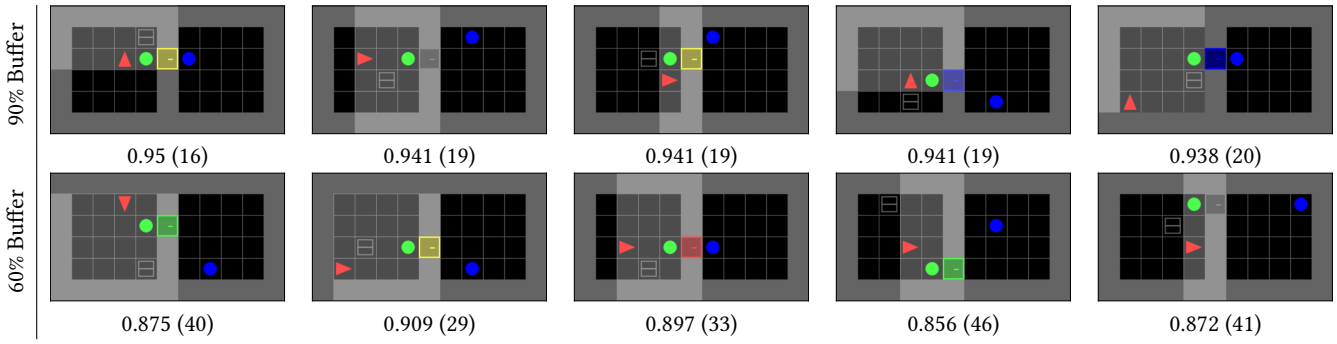

Figure 7: Levels corresponding to the trajectories collected in `O1Dlhb`. Below each level the return (and corresponding steps) of the trajectory/demonstration to be mimic is eased.

each buffer. In order to verify this, in Figure 7 we show the specific sampled levels in `O1Dlhb` together with the return and number of steps of the associated trajectory. The first 2 levels beginning from the left are used for the reported '2 levels' results in Figure 6; in the same way, the first 3 levels beginning from the left in Figure 7 are used for the reported '3 levels' in Figure 6; and all the provided 5 levels in Figure 7 for the '5 levels' results in Figure 6. Inspecting these levels demonstrates that the trajectories within the 60% buffer are notably suboptimal (the expected optimal number of steps required to solve levels in this environment are ≈ 26, see Table 1), whereas the levels within the 90% buffer contain trajectories with as few as 16-20 steps. The distribution of levels and thereby trajectories contained within the 90% buffer is therefore skewed towards easier levels which require shorter trajectories than expected for levels of this environment.

Therefore, there are two main possible reasons that might explain why the agent pre-trained with few trajectories from the 90% buffer exhibits worse results:

- **(1) The stored distribution of levels**. Each trajectory contained in the buffer belongs to a specific level, which at the same time requires a different number of steps to be solved optimally [38]. Thus, some levels can be considered easier due to them requiring fewer steps which leads to trajectories with higher returns. The RAPID prioritisation leads to trajectories of such easier levels to be prioritised over trajectories of other levels [1], causing a shift in the distribution of stored levels.
- **(2) The coverage and interactions represented by the trajectories within these levels.** Suboptimal trajectories in MiniGrid are longer than optimal trajectories, thereby covering a larger part of the state space and possible interactions with the environment which might be beneficial for learning skills required in the task.

Regarding the first hypothesis, we visualise the probability distribution of sampling trajectories depending on their number of steps in Figure 8. The distribution related to the steps needed to

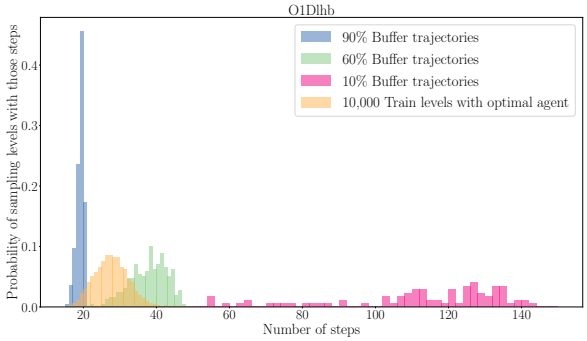

Figure 8: Probability distribution of sampling trajectories with variable number of steps from the 10% (pink), 60% (green) and 90% (blue) buffers. The same distribution is provided when doing it across the 10,000 train levels with an optimal agent (orange).

complete each task by an optimal agent across 10,000 train levels (orange) is not covered by any of the buffers. For the 90% buffer, the overlap with this distribution is fairly small, clearly indicating that the levels covered within this buffer are skewed towards levels with shorter optimal solutions between 15 and 21 steps. In contrast, the 10% buffer only contains highly suboptimal trajectories. Only the 60% buffer contains a notable number of levels which are representative of the data distribution generated by an optimal agent which might explain the results shown in Figure 6.

For our second hypothesis, we raise the following question: *What would happen if we explicitly select levels present in both the 90% and 60% buffers and train the agent with the respective trajectories of the buffers within these levels?* We reproduce the results of Figure 6 for `O1Dlhb` in Figure 9. When considering the same levels –yet different quality of trajectories– the results are very similar: agents trained from the 60% and 90% buffer exhibit robustness issues when using only 2 or 3 levels with instabilities being more severe for the 90% buffer. However, the agents pre-trained from the 90% buffer seem to converge slightly faster when having a larger amount of levels available.

In light of these results, we can state that the selection of levels (distribution shift of levels diversity) used for pre-training is perhaps surprisingly more important than the quality and quantity of the trajectories. This explains why in Figure 6 the 90% buffer reports worse results compared to the 60% buffer: the trajectories contained in the 90% buffer belong to levels that do not represent the whole level distribution, which can be seen in the mismatch between $\mu_{G(\tau)}$ and $\mathbb{E}^*[G(\tau)]$ in Table 1 for all the considered environments and also in the mismatch of probability distributions shown in Figure 8.

In summary, using IL to pre-train RL agents with only a handful of demonstrations can significantly speed-up the learning. Moreover, when using such low data regimes, it is more important to select trajectories belonging to the whole spectre of the level distribution (i.e., maximise the diversity of the levels) rather than providing optimal examples.

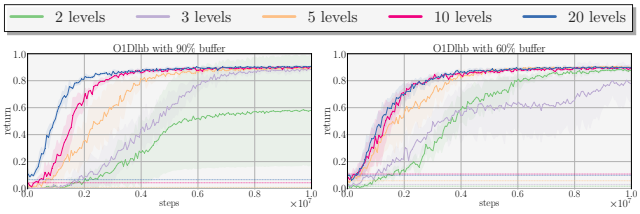

Figure 9: Interpretation as in Figure 6. Here, the same levels are used yet different quality of trajectories

7 CONCLUSIONS

In this paper, we studied the potential of *Imitation Learning* from offline data to improve the sample-efficiency and overall performance of on-policy RL algorithms in challenging PCG environments. We considered the setting of pre-training a policy using IL as well as concurrently optimising the policy with IL during online RL training. For this purpose, we collect demonstrations (buffers) with a variable quality, quantity and diversity belonging to different levels.

We show that pre-training on offline demonstrations leads to a significant jumpstart in the performance and consequently improved sample-efficiency in many tasks, even when provided demonstrations are far from optimal. To further improve the performance, the number of pre-training epochs can be increased. Besides, due to a possible misalignment between the offline trajectories and the actual level distributions, relaxing the maximum number of steps per episode is suggested. Concurrently training the agent with IL and RL during the online training exhibits robust performance, solving all considered tasks for demonstrations of various quality. Overall, the best strategy is to combine and use IL both for pre-training and during the online training concurrently with RL. Lastly, we provide empirical results showing that for all considered tasks pre-training with as few as 2 to 5 trajectories can make the agent learn an optimal solution, whereas RL without pre-training fails to solve the tasks. Interestingly, we find that the diversity of the distribution of trajectories used for pre-training is more important than the quality of these demonstrations. In particular, the pre-trained policy is more robust during following RL optimisation whenever the provided offline trajectories represent the full distribution of trajectories encountered for the PCG task.

Future work should consider extending the analysis of our work to other PCG environments (e.g., Procgen [4]) to study the effectiveness of proposed IL techniques in diverse settings. Moreover, the pre-training stage could make use of more advanced IL techniques such as adversarial IL [17, 34] and curriculum learning approaches [27] to further improve the performance of the agent. In line with our finding that diversity of the provided demonstrations is key for effective pre-training, techniques for unsupervised environment design to ensure diversity of levels [37] could be applied to ensure that demonstrations effectively cover the whole level distribution.

ACKNOWLEDGMENTS

Alain Andres and Javier Del Ser acknowledge for the received funding support from the Basque Government through the BIKAINTEK PhD and MATHMODE (ref. IT1456-22) respective programs.

REFERENCES

[1] Alain Andres, Esther Villar-Rodriguez, and Javier Del Ser. 2022. Towards Improving Exploration in Self-Imitation Learning using Intrinsic Motivation. https://doi.org/10.48550/arXiv.2211.16838 arXiv:2211.16838 [cs].

[2] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying Count-Based Exploration and Intrinsic Motivation. https://doi.org/10.48550/arXiv.1606.01868 arXiv:1606.01868 [cs, stat].

[3] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018. *Minimalistic Gridworld Environment for Gymnasium.* https://github.com/Farama-Foundation/Minigrid

[4] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. 2020. Leveraging Procedural Generation to Benchmark Reinforcement Learning. http://arxiv.org/abs/1912.01588 arXiv:1912.01588 [cs, stat].

[5] Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. 2020. Phasic Policy Gradient. https://doi.org/10.48550/arXiv.2009.04416 arXiv:2009.04416 [cs, stat].

[6] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. 2023. Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier. (2023).

[7] Andy Ehrenberg, Robert Kirk, Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. 2022. A Study of Off-Policy Learning in Environments with Procedural Content Generation. https://openreview.net/forum?id=rtWOANa41W5

[8] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. https://doi.org/10.48550/arXiv.1802.01561 arXiv:1802.01561 [cs].

[9] Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. 2021. Adversarially Guided Actor-Critic. https://doi.org/10.48550/arXiv.2102.04376 arXiv:2102.04376 [cs, stat].

[10] Qiming Fu, Zhicong Han, Jianping Chen, You Lu, Hongjie Wu, and Yunzhe Wang. 2022. Applications of reinforcement learning for building energy efficiency control: A review. *Journal of Building Engineering* 50 (June 2022), 104165. https://doi.org/10.1016/j.jobe.2022.104165

[11] Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. 2019. Guidelines for reinforcement learning in healthcare. *Nature Medicine* 25, 1 (Jan. 2019), 16–18. https://doi.org/10.1038/s41591-018-0310-5 Number: 1 Publisher: Nature Publishing Group.

[12] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. 2017. Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic. http://arxiv.org/abs/1611.02247 arXiv:1611.02247 [cs].

[13] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. 2019. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. https://doi.org/10.48550/arXiv.1910.11956 arXiv:1910.11956 [cs, stat].

[14] Gunshi Gupta, Tim G J Rudner, Rowan McAllister, Adrien Gaidon, and Yarin Gal. 2022. Can Active Sampling Reduce Causal Confusion in Offline Reinforcement Learning? (Dec. 2022).

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. https://doi.org/10.48550/arXiv.1801.01290 arXiv:1801.01290 [cs, stat].

[16] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. 2017. Deep Q-learning from Demonstrations. https://doi.org/10.48550/arXiv.1704.03732 arXiv:1704.03732 [cs].

[17] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*.

[18] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. 2021. Prioritized Level Replay. https://doi.org/10.48550/arXiv.2010.03934 arXiv:2010.03934 [cs].

[19] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (May 1998), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

[20] Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J. Roberts. 2022. Same State, Different Task: Continual Reinforcement Learning without Interference. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 7 (June 2022), 7143–7151. https://doi.org/10.1609/aaai.v36i7.20674 Number: 7.

[21] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. 2022. A Survey of Generalisation in Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.2111.09794 arXiv:2111.09794 [cs].

[22] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline Reinforcement Learning with Implicit Q-Learning. http://arxiv.org/abs/2110.06169 arXiv:2110.06169 [cs].

[23] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. 2022. When Should We Prefer Offline Reinforcement Learning Over Behavioral Cloning? http://arxiv.org/abs/2204.05618 arXiv:2204.05618 [cs].

[24] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-Learning for Offline Reinforcement Learning. http://arxiv.org/abs/2006.04779 arXiv:2006.04779 [cs, stat].

[25] Hao Liu and Pieter Abbeel. 2021. APS: Active Pretraining with Successor Features. http://arxiv.org/abs/2108.13956 arXiv:2108.13956 [cs].

[26] Hao Liu and Pieter Abbeel. 2021. Behavior From the Void: Unsupervised Active Pre-Training. http://arxiv.org/abs/2103.04551 arXiv:2103.04551 [cs].

[27] Minghuan Liu, Hanye Zhao, Zhengyu Yang, Jian Shen, Weinan Zhang, Li Zhao, and Tie-Yan Liu. 2022. Curriculum Offline Imitation Learning. http://arxiv.org/abs/2111.02056 arXiv:2111.02056 [cs].

[28] Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, and Sergey Levine. 2021. AW-Opt: Learning Robotic Skills with Imitation and Reinforcement at Scale. https://doi.org/10.48550/arXiv.2111.05424 arXiv:2111.05424 [cs].

[29] 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 (Feb. 2015), 529–533. https://doi.org/10.1038/nature14236 Number: 7540 Publisher: Nature Publishing Group.

[30] Sharada Mohanty, Jyotish Poonganam, Adrien Gaidon, Andrey Kolobov, Blake Wulfe, Dipam Chakraborty, Gražvydas Šemetulskis, João Schapke, Jonas Kubilius, Jurgis Pašukonis, Linas Klimas, Matthew Hausknecht, Patrick MacAlpine, Quang Nhat Tran, Thomas Tumiel, Xiaocheng Tang, Xinwei Chen, Christopher Hesse, Jacob Hilton, William Hebgen Guss, Sahika Genc, John Schulman, and Karl Cobbe. 2021. Measuring Sample Efficiency and Generalization in Reinforcement Learning Benchmarks: NeurIPS 2020 Procgen Benchmark. http://arxiv.org/abs/2103.15332 arXiv:2103.15332 [cs].

[31] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2021. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets. https://doi.org/10.48550/arXiv.2006.09359 arXiv:2006.09359 [cs, stat].

[32] Hai Nguyen, Andrea Baisero, Dian Wang, Christopher Amato, and Robert Platt. 2022. Leveraging Fully Observable Policies for Learning under Partial Observability. http://arxiv.org/abs/2211.01991 arXiv:2211.01991 [cs].

[33] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-Imitation Learning. https://doi.org/10.48550/arXiv.1806.05635 arXiv:1806.05635 [cs, stat].

[34] Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. 2021. What matters for adversarial imitation learning?. In *Advances in Neural Information Processing Systems*.

[35] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. 2018. An Algorithmic Perspective on Imitation Learning. *Foundations and Trends in Robotics* 7, 1-2 (2018), 1–179. https://doi.org/10.1561/2300000053 arXiv:1811.06711 [cs].

[36] Georg Ostrovski, Marc G. Bellemare, Aaron van den Oord, and Remi Munos. 2017. Count-Based Exploration with Neural Density Models. https://doi.org/10.48550/arXiv.1703.01310 arXiv:1703.01310 [cs].

[37] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. 2022. Evolving Curricula with Regret-Based Environment Design. https://doi.org/10.48550/ARXIV.2203.01302

[38] Roberta Raileanu and Rob Fergus. 2021. Decoupling Value and Policy for Generalization in Reinforcement Learning. http://arxiv.org/abs/2102.10330 arXiv:2102.10330 [cs].

[39] Roberta Raileanu and Tim Rocktäschel. 2020. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. https://doi.org/10.48550/arXiv.2002.12292 arXiv:2002.12292 [cs].

[40] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2018. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. https://doi.org/10.48550/arXiv.1709.10087 arXiv:1709.10087 [cs].

[41] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. 2022. A Generalist Agent. http://arxiv.org/abs/2205.06175 arXiv:2205.06175 [cs].

[42] Lukas Schäfer, Filippos Christianos, Josiah P. Hanna, and Stefano V. Albrecht. 2022. Decoupled Reinforcement Learning to Stabilise Intrinsically-Motivated Exploration. In *International Conference on Autonomous Agents and Multiagent Systems*.

[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. https://doi.org/10.48550/arXiv.

1707.06347 arXiv:1707.06347 [cs].

[44] Mathieu Seurin, Philippe Preux, and Olivier Pietquin. 2020. I'm sorry Dave, I'm afraid I can't do that, Deep Q-learning from forbidden action. http://arxiv.org/abs/1910.02078 arXiv:1910.02078 [cs, stat].

[45] Yuda Song, Yifei Zhou, Ayush Sekhari, J. Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. 2022. Hybrid RL: Using Both Offline and Online Data Can Make RL Efficient. http://arxiv.org/abs/2210.06718 arXiv:2210.06718 [cs].

[46] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2018. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. https://doi.org/10.48550/arXiv.1707.08817 arXiv:1707.08817 [cs].

[47] Benjamin Wexler, Elad Sarafian, and Sarit Kraus. 2022. Analyzing and Overcoming Degradation in Warm-Start Off-Policy Reinforcement Learning. (Sept. 2022).

[48] Zhihui Xie, Zichuan Lin, Junyou Li, Shuai Li, and Deheng Ye. 2022. Pretraining in Deep Reinforcement Learning: A Survey. https://doi.org/10.48550/arXiv.2211.03959 arXiv:2211.03959 [cs].

[49] Yang Yue, Bingyi Kang, Xiao Ma, Zhongwen Xu, Gao Huang, and Shuicheng Yan. 2022. Boosting Offline Reinforcement Learning via Data Rebalancing. http://arxiv.org/abs/2210.09241 arXiv:2210.09241 [cs].

[50] Daochen Zha, Wenye Ma, Lei Yuan, Xia Hu, and Ji Liu. 2021. Rank the Episodes: A Simple Approach for Exploration in Procedurally-Generated Environments. https://doi.org/10.48550/arXiv.2101.08152 arXiv:2101.08152 [cs].

[51] Haichao Zhang, We Xu, and Haonan Yu. 2023. Policy Expansion for Bridging Offline-to-Online Reinforcement Learning. https://doi.org/10.48550/arXiv.2302.00935 arXiv:2302.00935 [cs].

[52] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. 2020. BeBold: Exploration Beyond the Boundary of Explored Regions. https://doi.org/10.48550/arXiv.2012.08621 arXiv:2012.08621 [cs, stat].

[53] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. 2018. Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost. http://arxiv.org/abs/1810.06045 arXiv:1810.06045 [cs].