2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Enhanced Generalization through Prioritization and Diversity in Self-Imitation Reinforcement Learning over Procedural Environments with Sparse Rewards

Alain Andres*†‡, Daochen Zha§, and Javier Del Ser†‡
† TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain
‡ University of the Basque Country (UPV/EHU), 48013 Bilbao, Bizkaia, Spain
§ Rice University, Houston, Texas, United States
* Corresponding author: alain.andres@tecnalia.com

***Abstract*—Exploration poses a fundamental challenge in Reinforcement Learning (RL) with sparse rewards, limiting an agent's ability to learn optimal decision-making due to a lack of informative feedback signals. Self-Imitation Learning (self-IL) has emerged as a promising approach for exploration, leveraging a replay buffer to store and reproduce successful behaviors. However, traditional self-IL methods, which rely on high-return transitions and assume singleton environments, face challenges in generalization, especially in procedurally-generated (PCG) environments. Therefore, new self-IL methods have been proposed to rank which experiences to persist, but they replay transitions uniformly regardless of their significance, and do not address the diversity of the stored demonstrations. In this work, we propose tailored self-IL sampling strategies by prioritizing transitions in different ways and extending prioritization techniques to PCG environments. We also address diversity loss through modifications to counteract the impact of generalization requirements and bias introduced by prioritization techniques. Our experimental analysis, conducted over three PCG sparse reward environments, including MiniGrid and ProcGen, highlights the benefits of our proposed modifications, achieving a new state-of-the-art performance in the MiniGrid-MultiRoom-N12-S10 environment.**

***Index Terms*—Reinforcement Learning, Self-Imitation Learning, Experience Replay Buffer, Generalization, Diversity**

I. Introduction

Exploration is a fundamental challenge in Reinforcement Learning (RL), especially in scenarios with sparse rewards where the agent may struggle to learn optimal decision-making due to a lack of informative feedback signals [1]–[5].

One promising approach to improve exploration is self-Imitation Learning (self-IL), which uses a replay buffer to store past successful behaviors. In this way, the agent can reproduce and exploit rare instances of good exploration. This idea has traditionally been achieved by selecting high-return transitions that provide positive advantage [3]–[8]. However, they often assume that agents operate in singleton environments, using the same environment for training and testing. Yet, recent studies have unveiled that this approach's susceptibility to overfitting hampers its generalization capabilities [9], [10]. To address this issue, the adoption of procedurally-generated (PCG) environments has been proposed [11], where a different environment is generated in each episode. Unfortunately, self-IL approaches often yield poor results in PCG environments, as the agent may not be able to encounter a high-return transition more than once [12].

In previous work we proposed **Ra**nking the E**pi**so**d**es (RAPID) [12], which takes into account the whole episode rather than isolated transitions, effectively distinguishing good exploration behaviors that handle state space changes affecting the pursued generalization capacity. We discovered that episode-level selection can significantly boost the sample efficiency in PCG environments [12]. However, RAPID solely focuses on which experiences to store, and replays data uniformly from the buffer, indirectly implying that all experiences of such self-collected demonstrations are equally valuable. Furthermore, it does not guarantee diversity of behaviors between the stored episodes [13], [14], potentially resulting in over-fitted policies incapable of generalizing.

To address the above limitations, in this work, we propose methods to specifically tailor self-IL sampling strategies (Section III-A). Instead of treating all transitions as equally significant, we prioritize them in different ways. Moreover, we extend prioritization techniques that have been previously evaluated with off-policy methods in singleton environments to PCG environments. Furthermore, we propose modifications to counteract the diversity loss stemming from the tasks' generalization requirements and the bias introduced by prioritization techniques (Section III-B). The resulting analysis demonstrates the performance benefits of our modifications through experiments conducted in three PCG sparse reward environments: `MultiRoom` and `ObstructedMaze` from MiniGrid [11] and `Ninja` from ProcGen [15] (Section IV). In particular, our approach establishes a new state-of-the-art in the `MiniGrid-MultiRoom-N12-S10` environment.

II. Preliminaries

This section begins with a background on self-Imitation Learning (Section II-A), followed by a discussion regarding the Experience Replay Buffer adoption (Section II-A). Then we formally describe our research goal (Section II-C).

A. Self-Imitation Learning

Self-imitation learning (self-IL) is a promising approach to foster exploration by leveraging good behaviors exhibited in the past. Initially introduced in the so-called SIL [7], this method employs a replay buffer to retain historical state-action pairs, imitating only those pairs that yielded greater returns than the agent's value estimate in previous episodes. By leveraging good past behaviors, the agent achieves enhanced exploration capabilities, ultimately leading to improved sample efficiency [7].

However, SIL [7] assumes that agents operate solely within singleton environments, meaning that the same environment is employed for both training and testing purposes. Recent studies have revealed that training an agent in such a manner renders it susceptible to overfitting, hindering its ability to generalize [9], [10]. Thus, the utilization of PCG environments has been advocated [11]. By generating distinct environments for each episode, the agent is encouraged to learn generalizable skills.

To enable self-IL in PCG environments, we proposed RAPID in our previous work [12]. The idea behind RAPID is to facilitate agents to replicate good episode-level exploration behaviors. This is accomplished by assigning a score to the entire trajectory and rank all the past trajectories in a small replay buffer, based on the following formulation:

$$S = w_0 \cdot S_{ext} + w_1 \cdot S_{local} + w_2 \cdot S_{global}, \qquad (1)$$

where S_{ext} determines the extrinsic Monte Carlo return, S_{local} promotes diversity of states within the episode, S_{global} fosters the lifelong training exploration, and all the w_* are used to balance the weight given to each score. Building upon RAPID, our previous work further introduced Intrinsic Motivation (IM) to enable better exploration [13].

While RAPID [12] and RAPID+IM [13] have achieved promising performance, their utilization of a strict ranking-based buffer and uniform replay strategy may result in certain state-action pairs overpowering the learning process. This can potentially lead to learning divergence, as exemplified in the `MultiRoom-N12-S10` environment [12].

B. Experience Replay Buffer

One pivotal component of self-IL is the Experience Replay Buffer [16], [17], a data structure designed to retain past experiences. This buffer empowers the agent to effectively reuse previous experiences to enhance learning efficiency. In the past, numerous strategies have been proposed to enhance experience replay. These strategies can be broadly categorized into two groups: 1) defining how to replay experiences from the buffer, and 2) determining which experiences to store.

1) How to replay: Various non-uniform reply strategies have emerged, such as using the TD-error as proxy [6], applying importance sampling [6], [18], minimizing meaningless updates with episode-level sampling [19], adopting hierarchical experience replay [20], sampling frequently visited transitions [21], and optimizing the use of the buffer with meta-learning approaches [22], [23].

2) Which experiences to store: The most fundamental approach is the First-In-First-Out (FIFO) strategy, which employs a fixed-size memory to sequentially store data [17]. The works in [18], [24], [25] uncovered the significant impact of different sizes of Experience Replay Buffers on performance. Subsequently, a range of strategies has been proposed to enhance its utility. These strategies include increasing the diversity of experiences employing short-term and long-term buffers [26]–[28], prioritizing the storage of experiences with high rewards in a greedy manner [29], continuously refreshing the buffer according to the current policy [30], utilizing multiple buffers representing different event types [31], [32], as well as employing Hindsight Experience Replay [33].

C. Research Objective

Despite the aforementioned efforts, none of the strategies to manage the Experience Replay Buffer were specifically designed for self-IL, nor have they been tested in PCG environments. There lies the research goal pursued in this work: to examine the impact of various experience replay strategies for self-IL approaches within PCG environments. Our objectives to accomplish this goal are threefold: 1) to assess the efficacy of prioritization strategies in replay mechanisms, 2) to empirically evaluate the effectiveness of filtering strategies to avoid meaningless updates, and 3) to explore whether improving data diversity in the buffer could contribute positively to the learning process.

III. Designed strategies for Efficient Experience Replay in Self-Imitation Learning

We now introduce methods to prioritize the sampling from the buffer (Section III-A) and modifications to promote the diversity of transitions therein stored (Section III-B).

A. Prioritization & Filtering

Uniform sampling strategy has been largely adopted in experience replay due to its simplicity. However, an agent could learn more effectively from some transitions than from others. Motivated by this, we designed several prioritization and filtering methods for self-IL.

1) Prioritization: The idea is to replay some experiences with more frequency due to their significance in learning. We extend the idea of PER [6] considering three different proxies for *prioritization*:

- **TD-error** $\rightarrow \delta = r_t + \gamma \cdot V(s_{t+1}) - V(s_t)$, where r_t is the reward, $V(\cdot)$ is the value function, γ is the discount factor. It represents how *surprising* or unexpected a given transition is with respect to the knowledge retained at the agent. It has been shown to work well in practice with algorithms that have already computed the TD-error for updating its parameters (e.g., Q-learning or SARSA [17], [34], [35]). However, it can be a poor estimate under some circumstances, such as partial observability, sparse rewards, and stochastic transitions [6], which are common in PCG environments.

- **Log-Likelihood** $\rightarrow \sum_{(s_t,a_t)\sim B} ln(\pi(a_t|s_t))$, where $\pi(a_t|s_t)$ is the probability of selecting action a_t given the current state s_t. It suggests how likely an action may occur in a given state. The log-likelihood prioritization promotes frequent actions.
- **Novelty** $\rightarrow 1/\sqrt{N(s_t)}$, where $N(s_t)$ stands for the state visitation counts throughout the whole training. It aims to promote transitions that are more novel, fostering the exploration in those states in which the agent is uncertain about its captured knowledge.

Once the score p_i of each transition has been calculated according to any of the above proxies, the probability $P(i)$ of sampling a given transition i is defined as:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \tag{2}$$

where α is a hyperparameter used to determine how much prioritization is applied[1].

2) Filtering: Instead of prioritizing the experiences based on different proxies/scores like those explained above, an alternative strategy is to sample uniformly from the buffer, but apply some *filters* to avoid undesired updates. We consider the following filtering objectives:

- **Non-zero Return Trajectories** $\rightarrow G_t > 0$, where the discounted return is given by $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. It grants priority to those trajectories that represent a valid/success example[2] to complete the task. When no success trajectories are available in the buffer, the agent samples them uniformly. However, when a valid demonstration exists, the agent will imitate those experiences greedily.
- **Positive Advantage** $\rightarrow [G_t - V(s_t)]_+$, where G_t is the discounted return, $V(\cdot)$ is the value function, and $[\cdot]_+ = \max(\cdot, 0)$. Akin to SIL [7], this option only considers experiences that are expected to have a positive impact on the agent's learning process. Therefore, if a $\{s, a\}$ tuple has a worse return (G_t) than the one expected by the agent ($V(s_t)$), that transition is not considered for replay.
- **Unique states** $\rightarrow \mathbb{I}\{N_e(s_{t+1})\}$, where $\mathbb{I}\{\cdot\}$ takes value 1 if its argument is 1 (0 otherwise), and $N_e(s_{t+1})$ stands for the state visitation counts within an episode. Motivated by episodic exploration success approaches [36]–[38], with this filter w foster exploration by preventing the replay of transitions in the same trajectory that lead to the same s_{t+1}.

B. Data Diversity

RAPID focused exclusively on episode-level scores, without considering whether the stored demonstrations are closely related and provide a comprehensive representation of the required diversity. In previous studies conducted in PCG environments, it was observed that optimal solutions – and $V(s_t)$ – can significantly vary from one level to another, even within the same task [13], [14]. As a result, certain levels may yield a superior extrinsic return, G_t, which can inadvertently bias the diversity of the trajectories considered by RAPID, and lead to behavioral overfitting. Furthermore, the data replay strategies introduced earlier can induce an additional bias toward the selected proxy. We propose two strategies to alleviate such issues:

- **On-policy Novelty** (*Intrinsic Motivation*): The on-policy updates are decoupled from the off-policy ones. Therefore, novelty-seeking techniques such as Intrinsic Motivation [1], [2] can be used not only to promote the exploration, but also as a tool to prevent the agent from getting stuck in a local optimum due to low-diversity off-policy updates.
- **Forced Diversity**. Instead of considering any episode belonging to any level, we constrain the buffer so that it always contains a fixed number of episodes per level, *forcing* diversity among the levels represented in the buffer.

IV. Experiments & Results

This section describes the experimental setup used to assess the performance of the aforementioned strategies. Specifically, the environments and selected hyperparameter values are given in Section IV-A, whereas results are analyzed and discussed in Section IV-B.

A. Experimental Setup

a) Environments: Performance evaluations are carried out over MiniGrid and Procgen PCG environments, where the agent position, background, and even configuration of objects randomly change from episode to episode. We only consider sparse reward tasks that constitute an exploration challenge:

- **MiniGrid** [11]: Within this benchmark, we evaluate the solutions over a MultiRoom environment with 12 rooms and a maximum room size of 10 –`MN12S10`– where the agent has to open doors and move forward until reaching a far green square goal. In addition, we also consider an ObstructedMaze scenario – `O1Dlhb` – in which the agent must find out where a hidden key is, move a ball that obstructs the opening of the locked door, and move forward to another room where the goal is accomplished when picking up the ball placed in it. The agent is fed with a partially observable state of dimensions $7 \times 7 \times 3$ that represents the surroundings in a compact manner. The agent is capable of executing up to 7 possible discrete actions.
- **ProcGen** [15]: Among the 16 possible environments, we opt for `Ninja` as the agent is only rewarded with either a +0 or +10 reward depending on whether the agent succeeds in the completion of the task. The agent has to move forward while avoiding bombs that can kill itself, so it has to jump and move through multiple elevated platforms carefully. The input observation consists of a $64 \times 64 \times 3$ image, while the action space consists of 15 possible discrete values.

b) Hyperparameters: For MiniGrid, we use the hyperparameters and neural network architectures proposed in [13]. That is, we use Proximal Policy Optimization (PPO) [39] with an actor-critic framework, using 64-64 Multi-Layer Perceptron. Besides, we select the best value resulting from a grid search for $\alpha \in [0.2, 0.4, 0.6, 0.8, 1.0]$ in Equation (2)

[1]In Expression (2), $\alpha = 0$ refers to the uniform sampling case, whereas $\alpha = 1$ stands for maximum prioritization.

[2]We refer as valid success examples to any trajectory with $G_t \neq 0$.

when using any of the prioritizing strategies explained in Section III-A. That is, we use$\alpha = 0.6$ in `MN12S10` and $\alpha = 1.0$ in `O1Dlhb` for *Novelty* prioritization; for the rest of cases, we opt for $\alpha = 0.2$. As for ProcGen, we adopt the hyperparameters and architectures used in [15]. Furthermore, when IM is employed, we use BeBold [2] as in [13].

B. Results and Discussion

We report the mean and standard deviation of the average return calculated over the last 100 episodes for each experiment. Such statistics are computed over 3 different runs to account for the statistical variability of the results. In addition, we consider two batch sizes ($\mathcal{B}_{IL}$) for self-IL; unless otherwise stated, solid curves represent results obtained with $\mathcal{B}_{IL} = 256$, whereas dash-dotted lines indicate that the batch size in use is $\mathcal{B}_{IL} = 2048$.

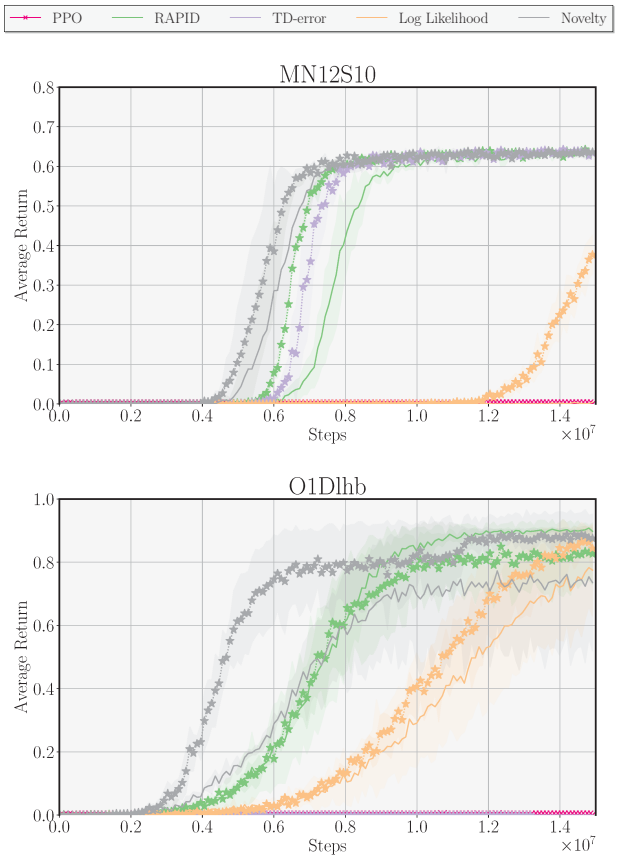

Fig. 1: Performance of the agent when adopting prioritization strategies in `MN12S10` and `O1Dlhb` tasks.

1) Prioritization & Filtering: According to the results in Figures 1 and 2, the following observations can be made:

Firstly, **prioritization** (Figure 1) based on *Novelty* (gray) consistently outperforms uniform sampling (green) in terms of sample efficiency for learning optimal policies. By contrast, using *TD-error* (magenta) as a proxy for prioritization yields poor results, only enabling learning over `MN12S10` with large batch sizes and failing in other cases. Moreover, the proposed *Log-Likelihood* prioritization strategy (yellow) manages to learn a good policy, but requires more interactions than uniform sampling.

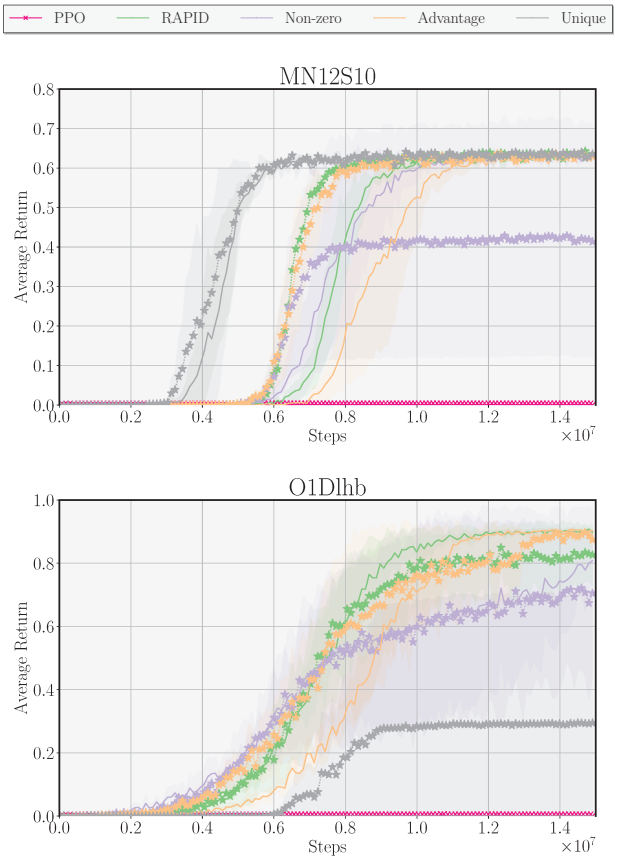

Fig. 2: Performance of the agent when adopting filtering strategies in `MN12S10` and `O1Dlhb` tasks.

When it comes to **filtering** strategies, Figure 2 reveals that these methods render faster learning compared to uniform sampling (green). In fact, the best result on `MN12S10` is achieved with the *Unique* filtering strategy. However, it is worth noting that adopting these filtering methods can potentially lead to a loss of diversity, as the agent tends to overfit to a subset of the entire experience distribution (e.g., *Non-zero*, magenta color, gets stuck in local optima solutions despite beginning to learn earlier on training). In contrast, uniform sampling (green) maintains diversity but at the cost of a slower learning and a lower sample efficiency.

Last but not least, increasing the **batch size** ($\mathcal{B}_{IL} = 2048$) improves sample efficiency and the likelihood of obtaining a successful policy compared to a smaller batch size ($\mathcal{B}_{IL} = 256$). This trend is consistent across all scenarios and cases, where the agent converges faster and/or achieves a valid policy. Increasing the batch size ensures that larger amounts of information are considered within each update, maximizing the probability of receiving valuable feedback and minimizing variance. This aspect is particularly critical in PCG environments, where generalization is essential for effective

learning and exploration [13]. However, it is important to note that while increasing the batch size reduces learning variance, it may also introduce a trade-off by potentially increasing bias. This trade-off is observed in scenarios such as `MN12S10` with the *Non-zero* filtering strategy and in `O1Dlhb` across all algorithmic approaches, where a larger batch size leads to faster learning, but poses challenges in attaining the optimal expected performance.

Based on these observations, it is hypothesized that by ensuring diversity while employing the aforementioned prioritization and filtering techniques, both sample efficiency and the agent's learning capacity can be improved.

2) Analysis of Data Diversity:

a) Intrinsic Motivation: In prior work [13], the combination of self-IL and IM in PCG environments was demonstrated to be a successful framework for improving sample efficiency. However, the influence of IM on diversity remained unexplored. Figure 3 exposes that **prioritization and filtering methods that previously resulted in suboptimal performance** (e.g., *non-zero*) **can now converge to the optimal policy when utilizing BeBold**.

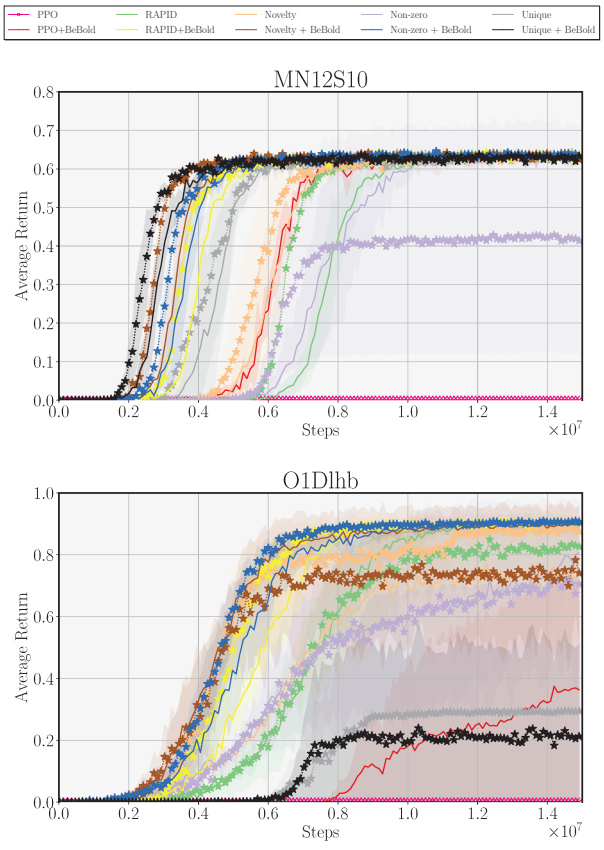

Fig. 3: Performance of the agent when fostering the diversity of self-IL with IM while adopting prioritization and filtering strategies in `MN12S10` and `O1Dlhb` tasks.

More importantly, the proposed modifications poses a new landmark performance level in terms of sample efficiency [13]. This is evident when comparing the number of samples required to reach a specific performance level. For instance, in `MN12S10`, the *Novelty* prioritization (brown) and *Unique* filtering (black) methods converge to the expected optimal policy in approximately $3.8M$ steps, while uniform sampling (yellow) requires around $5M$ steps[3]. That is, **the same optimal solution is achieved with 24% fewer interactions**. Similarly, the *Non-zero* filtering (blue) approach improves sample efficiency by approximately 10%. In the case of `O1Dlhb`, sample efficiency gaps are less noticeable. However, the *Non-zero* filtering approach (blue) still dominates by **reducing the number of required steps by about 11%**.

b) Forced Diversity: We recall that in Section III-B we indicate the potential need for addressing the lack of diversity in the stored trajectories. In light of the results reported in Section IV-B1, the diversity should be addressed. One way to do this is by imposing a fixed number of episodes per level (i.e., 1 episode per level, 1_ep, or 4 episodes per level, 4_ep), so that the representation of all levels in the buffer is always guaranteed. We evaluate this idea in Figure 4, with 10 training levels for task `O1Dlhb`. Neither 1_ep nor 4_ep achieve better results with respect to the *Default* setup executed with RAPID regardless of diversity issues.

Furthermore, we also consider two more ablation studies related to the difference of optimal solutions between levels [13], [14]: *Normalized* and *NormalizedFlex*. The first normalizes the extrinsic return obtained from the environment according to the optimal number of steps of each level, preventing the agent from repleting the buffer content with levels requiring shorter optimal paths. The second approach –*NormalizedFlex*– is an extension of *Normalized* that assigns a maximum return score of 1.0 to any collected trajectory that requires 0 to 20 steps more than the actual shortest solution. Based on these modifications, the agent manages to learn with fewer interactions. However, **all the analyzed approaches are incapable of performing well in all the 10 selected training levels**. Concretely, they fail into learning 1 out of 10 levels.

For completeness, we extend the aforementioned results to another environment with different state space and complexity (`Ninja`). Such results are summarized in Figure 5. Indeed, since the input is an image, RAPID cannot compute the local and global exploration scores, as it was originally designed for discrete state spaces [12]. Therefore, we applied RAPID considering only the extrinsic return, i.e., $S = w_0 \cdot S_{ext} = \sum_k \gamma^k r_{t+k}$ with respect to the original Equation (1). As opposed to previous outcomes, we observe that **in `Ninja`, storing at least one episode per level has a positive impact** on the agent's learning, making the difference between learning an almost optimal policy or a policy that barely surpasses a random agent's performance.

V. Conclusions and Future Work

The pivotal idea we wish readers to take away from this research is the threefold necessity of applying self-IL,

[3]Measures taken for curves with $\mathcal{B}_{IL} = 2048$. Improvements are also consistent for $\mathcal{B}_{IL} = 256$.

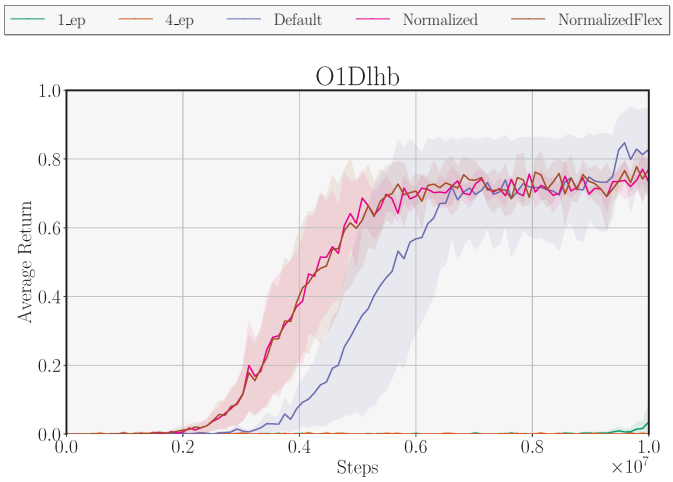

Fig. 4: Average train return over 10 levels in `O1Dlhb`. The expected optimal performance is approximately 0.9. All the shown curves use RAPID+IM (i.e., BeBold) and $\mathcal{B}_{IL} = 256$.

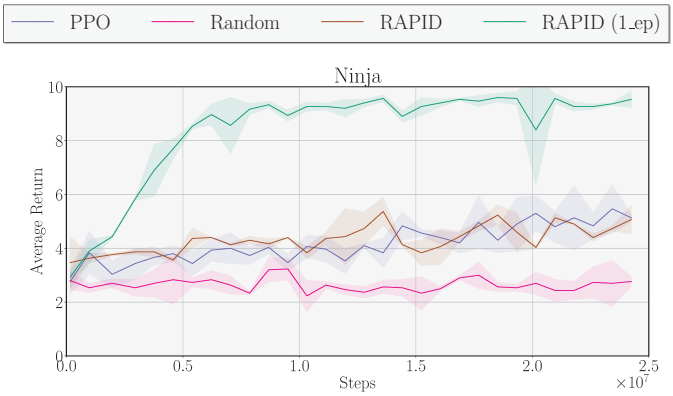

Fig. 5: Average train return over 200 levels in ProcGen's `Ninja` task. RAPID is used with $w_1 = w_2 = 0$. The expected optimal performance is equal to 10.

particularly for generalization over PCG environments.

Firstly, it is crucial to manage which demonstrations to store in the buffer, showing that episode-level exploration scores are suitable for this purpose [12]. Secondly, uniform sampling replay strategies assume all the stored experiences to be equally valuable, despite not being necessarily true. The prioritizing and filtering strategies proposed in this work have shown improvements in terms of sample efficiency, with *Novelty* prioritization and *Unique* filtering the ones yielding better outcomes. Notably, the latter has established a new state-of-the-art performance in `MN12S10`. Moreover, we have shown that increasing the imitation batch size increases the probability of sampling valuable experiences, reducing the variance of the updates and the required interactions to learn.

Last but not least, ensuring diversity is necessary for generalization. We have concluded that Intrinsic Motivation is an effective tool not only to foster on-policy exploration, but also to avoid the overfitting derived from prioritization and filtering techniques. Furthermore, in some scenarios like MiniGrid, having a diverse set of episodes to be imitated is not of help. However, in other tasks like `Ninja` from ProcGen, forcing such diversity between episodes can boost performance.

In the future, this research work can be extended by using alternative techniques that guarantee the diversity between the stored episodes. Besides Behavior Cloning, it would be interesting to see how other techniques related to Imitation Learning (e.g., GAIL, GASIL [8], [40]) or Inverse Reinforcement Learning can perform in these PCG environments. Finally, finding out effective exploration scores suitable for continuous state spaces would allow the adoption of RAPID to learn over wider problems.

Acknowledgments

A. Andres and J. Del Ser receive support from the FaRADAI project (ref. 101103386) funded by the European Commission under the European Defence Fund (EDF-2021-DIGIT-R). J. Del Ser also acknowledges funding from the Basque Government (MATHMODE, IT1456-22).

References

[1] T. Zhang, P. Rashidinejad, J. Jiao, T. Yuandong, J. E. Gonzalez, and S. Russell, "MADE: Exploration via Maximizing Deviation from Explored Regions," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021, pp. 9663–9680.

[2] T. Zhang, H. Xu, X. Wang, Y. Wu, K. Keutzer, J. E. Gonzalez, and Y. Tian, "NovelD: A Simple yet Effective Exploration Criterion," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021, pp. 25 217–25 230.

[3] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, "Deep Q-learning from Demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

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

[5] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6292–6299.

[6] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," in *International Conference on Learning Representations (ICLR)*, 2016.

[7] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-Imitation Learning," in *International Conference on Machine Learning (ICML)*, 2018, pp. 3878–3887.

[8] Y. Guo, J. Oh, S. Singh, and H. Lee, "Generative adversarial self-imitation learning," *arXiv preprint arXiv:1812.00950*, 2018.

[9] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade, "Towards generalization and simplicity in continuous control," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6553–6564.

[10] A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," *arXiv preprint arXiv:1806.07937*, 2018.

[11] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic Gridworld Environment for Gymnasium," https://github.com/Farama-Foundation/Minigrid, 2018.

[12] D. Zha, W. Ma, L. Yuan, X. Hu, and J. Liu, "Rank the Episodes: A Simple Approach for Exploration in Procedurally-Generated Environments," in *International Conference on Learning Representations (ICLR)*, 2021.

[13] A. Andres, E. Villar-Rodriguez, and J. Del Ser, "Towards Improving Exploration in Self-Imitation Learning using Intrinsic Motivation," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Singapore, 2022, arXiv:2211.16838 [cs].

[14] R. Raileanu and R. Fergus, "Decoupling value and policy for generalization in reinforcement learning," in *International Conference on Machine Learning*, 2021, pp. 8787–8798.

[15] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2020, pp. 2048–2056.

[16] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3, pp. 293–321, 1992.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] T. de Bruin, J. Kober, K. Tuyls, and R. Babu˘ska, "Experience Selection in Deep Reinforcement Learning for Control," *Journal of Machine Learning Research*, vol. 19, no. 1, pp. 347–402, 2018.

[19] S. Y. Lee, C. Sungik, and S.-Y. Chung, "Sample-efficient deep reinforcement learning via episodic backward update," *Advances in Neural Information Processing Systems (NIPS)*, vol. 32, 2019.

[20] H. Yin and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017, pp. 1640–1646.

[21] P. Sun, W. Zhou, and H. Li, "Attentive Experience Replay," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5900–5907, 2020.

[22] D. Zha, K.-H. Lai, K. Zhou, and X. Hu, "Experience replay optimization," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 4243–4249.

[23] Y. Oh, L. Kimin, J. Shin, E. Yang, and S. J. Hwang, "Learning to sample with local and global contexts in experience replay buffers," in *International Conference on Learning Representations (ICLR)*, 2021.

[24] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv preprint arXiv:1712.01275*, 2017.

[25] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney, "Revisiting fundamentals of experience replay," in *International Conference on Machine Learning (ICML)*, 2020, pp. 3061–3071.

[26] T. De Bruin, J. Kober, K. Tuyls, and R. Babuška, "The importance of experience replay database composition in deep reinforcement learning," in *Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[27] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "Improved deep reinforcement learning for robotics through distribution-based experience retention," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3947–3952.

[28] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[29] T. G. Karimpanal and R. Bouffanais, "Experience Replay Using Transition Sequences," *Frontiers in Neurorobotics*, vol. 12, p. 32, 2018.

[30] Y. Du, G. Warnell, A. Gebremedhin, P. Stone, and M. E. Taylor, "Lucid dreaming for experience replay: refreshing past states with the current policy," *Neural Computing and Applications*, vol. 34, no. 3, pp. 1687–1712, 2022.

[31] V. Kompella, T. Walsh, S. Barrett, P. Wurman, and P. Stone, "Event tables for efficient experience replay," *Transactions on Machine Learning Research (TMLR)*, 2023.

[32] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano, "Outracing champion Gran Turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[33] Z.-W. Hong, T. Chen, Y.-C. Lin, J. Pajarinen, and P. Agrawal, "Topological experience replay," in *International Conference on Learning Representations (ICLR)*, 2021.

[34] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[35] G. A. Rummery and M. Niranjan, "On-line Q-learning using Connectionist Systems," *University of Cambridge*, vol. 37, p. 21, 1994.

[36] A. Andres, E. Villar-Rodriguez, and J. Del Ser, "An Evaluation Study of Intrinsic Motivation Techniques Applied to Reinforcement Learning over Hard Exploration Environments," in *Machine Learning and Knowledge Extraction*, 2022, pp. 201–220.

[37] M. Henaff, M. Jiang, and R. Raileanu, "A study of global and episodic bonuses for exploration in contextual MDPs," *arXiv preprint arXiv:2306.03236*, 2023.

[38] K. Wang, K. Zhou, B. Kang, J. Feng, and S. Yan, "Revisiting Intrinsic Reward for Exploration in Procedurally Generated Environments," in *International Conference on Learning Representations (ICLR)*, 2023.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[40] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.