Using Hippocampal Replay to Consolidate Experiences in Memory-Augmented Reinforcement Learning

John Tan Chong Min
Department of Electrical and Computer Engineering
National University of Singapore
johntancm@u.nus.edu.sg

Mehul Motani
Department of Electrical and Computer Engineering
National University of Singapore
motani@nus.edu.sg

Abstract

Reinforcement Learning (RL) agents traditionally face difficulties to learn in sparse reward settings. Go-Explore is a state-of-the-art algorithm that learns well in spite of sparse reward, largely due to storing experiences in external memory and updating this memory with better trajectories. We improve upon this method and introduce a more efficient count-based approach for both state selection (“Go") and exploration (“Explore") phases, as well as perform a novel form of hippocampal replay inspired from sharp-wave ripples (SWR) during hippocampal replay in mice to consolidate successful trajectories and enable consistent performance.

1 Introduction

Reinforcement Learning (RL) agents traditionally face difficulties to learn in sparse reward settings, as rewards are the main signal to motivate an agent’s behavior, and sparse rewards mean that many experiences do not enable the agent to learn. In order to promote learning in these settings, Andrychowicz et al. (2017) proposes Hindsight Experience Replay (HER) in order to randomly sample goals from learnt trajectories, learning from any experience. While this can eventually learn to reach the goal state through an implicit curriculum learning of learning more pseudo-goal states, this method can be time consuming to learn how to solve the task. Another approach to learning is to use a memory-based augmentation in order to store the desired information of each state. This is done in Go-Explore (Ecoffet et al., 2019, 2021), which stores the memory for each state, including the action trajectory to reach there and the total reward gained till that state. This memory is used for the “Go" phase in order to choose a promising state to return to, before exploring more from that state in the “Explore" phase. By only updating the memory if there is an improvement, it can eventually lead to finding the optimal path with enough exploration.

Unfortunately, such a pursuit for optimality may necessitate continual exploration in the environment, as even after solving the environment, we will still need to explore in order to ensure that the optimal path is discovered at least once. In contrast to existing work in RL, we do not seek to find the best possible solution for a given environment. We seek instead, to find a *satisficing* solution, whereby the solution is good enough to solve the task. We propose to do this via our novel approach of hippocampal replay combined with a count-based approach (see also Massi et al. (2022) for variants of hippocampal replay in RL), which consolidates successful trajectories and repeats them consistently. This allows our method to learn faster and adapt better to novel environments in real-world systems.

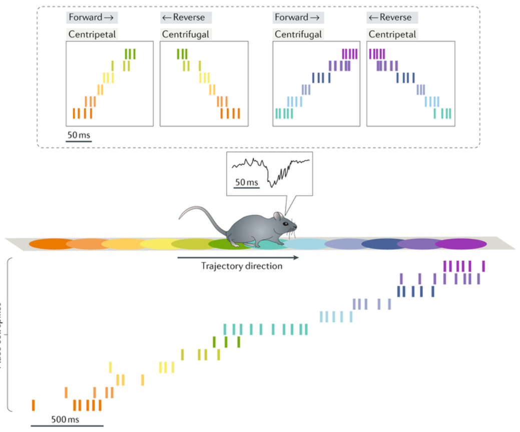

Figure 1: Hippocampal replay in mice, which showcases forward play (pre-play) and reverse play (replay), which are involved in memory retrieval and consolidation for processes such as decision-making. Extracted from Fig. 2 of Joo & Frank (2018). There is replay occurring for both 1) past visited states and 2) future unvisited states. We only focus on replay for past visited states, and use these insights in designing Algorithm 1 for consolidating learnt experiences. We posit that the replay for future unvisited states can help with lookahead planning for decision making.

Our Contributions. We propose two key improvements to Go-Explore:

1. We incorporate a count-based approach, which uses past memory to balance between explore and exploit, for more efficient state selection and exploration in the “Go" and “Explore" phases.
2. In order to improve consolidation of learnt trajectories, we propose a novel approach of *hippocampal replay* to reset state visit and selection counts of states along the successful trajectory. We demonstrate that hippocampal replay helps an agent remember successful trajectories which solve its current environment and enables it to perform consistently.

2 Methods

We detail the agents we use in our paper[1]. There are a total of 4 different agents:

1. **Random Agent.** This agent chooses a valid move randomly, and serves as a worst-case baseline.
2. **Go-Explore Agent.** Go-Explore was implemented similar to Ecoffet et al. (2019, 2021), except that we select states deterministically in the “Go" phase for faster training.
3. **Go-Explore-Count Agent.** While Go-Explore uses a random policy for exploration, Go-Explore-Count performs the best action based on a count-based selection function.
4. **Explore-Count Agent.** Explore-Count is Go-Explore-Count without the “Go" phase.

Memory. We use a similar memory update procedure as in Ecoffet et al. (2019, 2021). For each visited state, we store in memory: 1) trajectory of actions to reach that state, 2) the number of moves to reach it, 3) the reward, 4) the number of selections of the state in the "Go" phase (initialized to 0), and 5) the number of visits to the state in the "Explore" phase (initialized to 0). For each state we select/visit, we increment the selection/visit counts by 1. For the next state lookahead, we will update its memory if the one-step lookahead trajectory has a higher reward, or has the same reward but a shorter trajectory. This is an efficient method to keep the shortest performant trajectory in memory.

“Go" and “Explore" State selection. The selection function used for selecting the "Go" state as well as for choosing the best state to "Explore" in all the Go-Explore variants is given by:

$$\alpha \cdot reward + \kappa\sqrt{moves} - \gamma\sqrt{numselected + numvisited}, \tag{1}$$

where $moves$ represent the minimum number of timesteps taken to reach that state, $numselected$ represents the number of times the state has been selected by the “Go" phase, and $numvisited$ represents the number of times the state has been visited in the “Explore" phase. This will choose

[1]Source code for the experiments can be obtained from https://github.com/tanchongmin/Hippocampal-Replay

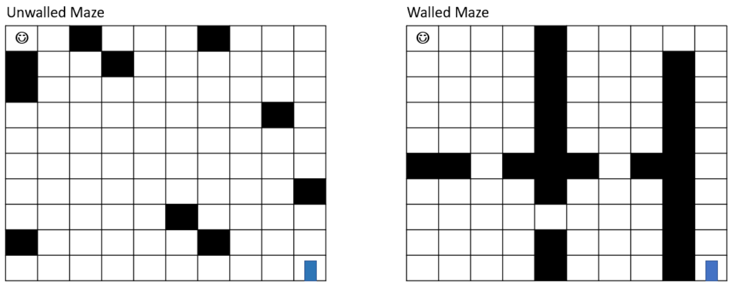

Figure 2: (Left) Unwalled Maze and (Right) Walled Maze of dimension 10x10. The smiley face represents the agent, the blue rectangle represents the door, white cells represent open area and black cells represent obstacles. Agent seeks to move to door position. Actions are up, down, left and right.

states with higher reward and more moves (greater chance of being at the frontier of explored states), while choosing states that have been relatively unvisited and not selected more often. This explore-exploit equation is similar to the UCT algorithm in Monte Carlo Tree Search (MCTS) (Chaslot et al., 2008) which decays the exploration term logarithmically so as to encourage greedy action selection in the long run. These attributes are all extracted from memory for the state in consideration.

α, κ, γ are hyperparameters for tuning. Here, we set them as 10, 1, 10 respectively.

Hippocampal Replay. Hippocampal replay was performed upon finding a memory with a trajectory that solves the task. This allows us to reset the selection and visit counts of the entire path of states leading to the goal state to 0 so as to facilitate exploration along a good path. Unlike Mnih et al. (2013) and Schaul et al. (2015), we do not randomly sample or priority sample the buffer, but instead just perform one forward play (pre-play) in order to do memory retrieval and get the entire list of states along the trajectory, and a reverse play (replay) so as to reset state selection and visit counts and update intrinsic rewards (if any). This is akin to biological pre-play and replay patterns shown in hippocampal replay (Joo & Frank, 2018) in mice. It is more efficient, as in one hippocampal replay step, all the states along the successful trajectory path can be updated, as opposed to updating just one state sampled from the buffer. The algorithm is detailed in Algorithm 1 and illustrated in Fig. 1.

Algorithm 1 Hippocampal Replay

```
1: procedure HippocampalReplay($env$, $trajectory$):
2:     Consolidate the list of states in the successful trajectory, in chronological order   ▷ Pre-play
3:     Visit states in reverse order from the goal, and update the memory of each state   ▷ Replay
```

3 Experiments

We test our agents on two discrete environments - Unwalled Maze and Walled Maze (see Fig. 2). Unwalled Maze contains obstacles randomly placed in 10% of the squares, while Walled Maze contains obstacles arranged in 4 chambers, with a narrow passageway at the end. Both environments have sparse rewards where a reward of 1 is only obtained when agent reaches the door and 0 otherwise. We run the environment 100 times, and report solve rate (number of solves out of 100), first solve run (related to *adaptability* - first run the agent solves the environment), first solve memory size, minimum, maximum, average steps (related to *efficiency* - statistics of the best, worst, average solved run). Memory size refers to the number of distinct states that are stored in memory.

4 Results and Discussion

We evaluate our agents on the Unwalled Maze and Walled Maze 100x100 environment with and without hippocampal replay, and detail the results in Tables 1 and 2 respectively.

Hippocampal Replay. From the results, we can see that hippocampal replay does not affect first solve run and memory size, since hippocampal replay only happens after a successful first solve.

Table 1: Performance results for Unwalled Maze 100x100 with and without hippocampal replay. '-HR' represents with hippocampal replay. Bolded text represents better performance.

Overall		**First Solve**		**Steps to Solve**		
Agent	**Solve Rate**	**Run**	**Memory size**	**Avg**	**Min**	**Max**
Random	1/100	15	-	9980.0	9980.0	9980.0
Go-Explore	0/100	-	-	-	-	-
Go-Explore-HR	0/100	-	-	-	-	-
Go-Explore-Count	78/100	**1**	**7597**	**5533.1**	**4312.0**	9498.0
Go-Explore-Count-HR	**100/100**	**1**	**7597**	6003.3	5984.0	**7854.0**
Explore-Count	98/100	**1**	**7597**	**3501.8**	**1436.0**	8286.0
Explore-Count-HR	**100/100**	**1**	**7597**	5984.0	5984.0	**5984.0**

Table 2: Performance results for Walled Maze 100x100 with and without hippocampal replay. '-HR' represents with hippocampal replay. Bolded text represents better performance.

Overall		**First Solve**		**Steps to Solve**		
Agent	**Solve Rate**	**Run**	**Memory size**	**Avg**	**Min**	**Max**
Random	0/100	-	-	-	-	-
Go-Explore	0/100	-	-	-	-	-
Go-Explore-HR	0/100	-	-	-	-	-
Go-Explore-Count	**100/100**	**1**	**7552**	4918.2	**4718.0**	6362.0
Go-Explore-Count-HR	**100/100**	**1**	**7552**	**4912.0**	4912.0	**4912.0**
Explore-Count	52/100	**1**	**7552**	7039.0	**3094.0**	9758.0
Explore-Count-HR	**100/100**	**1**	**7552**	**4912.0**	4912.0	**4912.0**

After the first solve, having hippocampal replay causes the agent to explore less, which leads to higher solve rate and lower variation in steps to solve. While the agent may not find the optimal solution (higher minimum steps compared to the non-hippocampal replay counterparts in all cases), the unique usage of hippocampal replay to reset state visitation and selection counts help to create a "highway" of high value for states along the good trajectory (see Fig. 3), and helps the agent repeat the trajectory consistently, albeit with some small variation based on explore-exploit selection.

Count-based Exploration. Go-Explore fails to solve the hardest 100x100 environment for both Walled and Unwalled Maze, likely because the "Go" step biases choosing longer trajectories. With the step limit of 10000, this can cause it to not have enough steps remaining to explore from the "Go" state to reach the goal. Go-Explore-Count and Explore-Count manages to solve both environments, highlighting that count-based strategies are more efficient than random exploration.

5 Effects of Changing Hyperparameters

In order to better understand the contributions of the various terms in the selection function, we compare the performance of Go-Explore, Go-Explore-Count and Explore-Count with different hyperparameters from the set {0, 1, 10} for (1). The results are detailed in Table 3.

Reward. Changing α does not impact the results as rewards are sparse in this setting.

Moves. Setting κ to 1 leads to the best performance (highest solve rate, lowest first solve run). κ at 0 or at 10 leads to poor performance, which shows that the number of moves is acting as a tie-breaker in order to better select the states at the frontier of exploration, but should not be the dominant term in the selection equation to prevent just selecting states with long trajectories.

State Visit and Selection Counts. Having a high γ of 10 leads to more exploration and improves the chance of finding a solution with lower minimum steps. However, setting γ to 0 leads to no solves at all, which highlights that there must be exploration in order to guide search for a solution.

Overall. Regardless of the hyperparameters, it can be seen that hippocampal replay leads to higher solve rate, but also higher minimum steps to solve. This further confirms the earlier intuition that with hippocampal replay, we can get consistent performance, but at a cost of a less optimal solution.

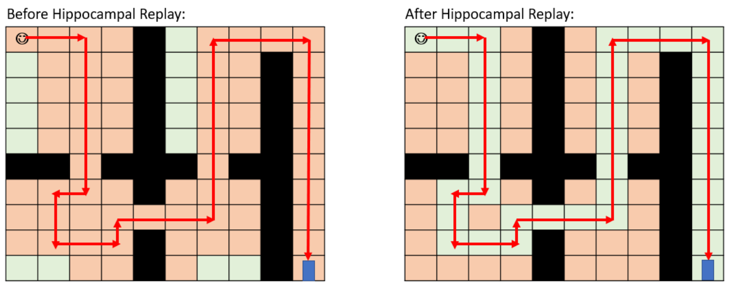

Figure 3: The Exploration Highway. (Left) Before hippocampal replay and (Right) After hippocampal replay. This is a hypothetical diagram denoting the low value states (red) and high value states (green), and obstacles (black). The red arrows indicate the discovered path, smiley face represents the agent, and blue rectangle represents the door. Hippocampal replay thus helps assign the states along the discovered path to be of high value and hence more likely to be visited again in the future.

Table 3: Performance results for Walled Maze 100x100 with and without hippocampal replay using various hyperparameters. '-HR' represents with hippocampal replay. 'X' refers to any value in {0, 1, 10}. Bolded text refers to best performance within the given hyperparameters.

Overall					First Solve		Steps to Solve		
α	κ	γ	**Agent**	**Solve Rate**	**Run**	**Mem**	**Avg**	**Min**	**Max**
X	1	1	Go-Explore-Count	**100/100**	**1**	**7454**	4831.3	4620.0	6704.0
			Go-Explore-Count-HR	**100/100**	**1**	**7454**	4804.0	4804.0	**4804.0**
			Explore-Count	41/100	**1**	**7454**	**3940.8**	**2510.0**	9098.0
			Explore-Count-HR	**100/100**	**1**	**7454**	4804.0	4804.0	**4804.0**
X	0	1	Go-Explore-Count	0/100	-	-	-	-	-
			Go-Explore-Count-HR	0/100	-	-	-	-	-
			Explore-Count	17/100	**2**	**7476**	8031.8	**5730.0**	9746.0
			Explore-Count-HR	**99/100**	**2**	7476	**7286.0**	7286.0	**7286.0**
X	10	1	Go-Explore-Count	0/100	-	-	-	-	-
			Go-Explore-Count-HR	0/100	-	-	-	-	-
			Explore-Count	14/100	**19**	**8581**	**2782.6**	**2260.0**	**3596.0**
			Explore-Count-HR	**82/100**	**19**	**8581**	3596.0	3596.0	**3596.0**
X	1	0	All	0/100	-	-	-	-	-
X	1	10	Go-Explore-Count	**100/100**	**1**	**7552**	4918.2	4718.0	6362.0
			Go-Explore-Count-HR	**100/100**	**1**	**7552**	**4912.0**	4912.0	**4912.0**
			Explore-Count	52/100	**1**	**7552**	7039.0	**3094.0**	9758.0
			Explore-Count-HR	**100/100**	**1**	**7552**	**4912.0**	4912.0	**4912.0**

6 Conclusion and Future Work

The field of RL has come a long way, and we now seek to focus on fast real-world adapation, which we propose to do with Go-Explore using the count-based approach and hippocampal replay. Hippocampal replay enables consolidation of good trajectories for higher chance of repetition in the future. This makes it suitable for use in a real-life robot, where optimization is a good-to-have (unlike in game-based RL environments such as Chess or Go), and fast adaptation to new situations is key.

Goal-Directed Intrinsic Reward. We have also combined hippocampal replay with a goal-directed "intrinsic" reward (e.g. Manhattan Distance) to boost the sparse reward signal, and this results in better performance. The results will be shown in a future paper.

Incorporating Neural Networks for Scaling. Our proposed memory-augmented method works well for discrete state and action spaces. For continuous spaces, we could utilize neural networks as the exploration policy, similar to policy-based Go-Explore (Ecoffet et al., 2021).

Acknowledgements

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-PhD-2021-01-003[T]). We would also like to thank the reviewers for their insightful comments which have helped improve this paper.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pp. 216–217, 2008.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Hannah R Joo and Loren M Frank. The hippocampal sharp wave–ripple in memory retrieval for immediate use and consolidation. *Nature Reviews Neuroscience*, 19(12):744–757, 2018.

Elisa Massi, Jeanne Barthélemy, Juliane Mailly, Rémi Dromnelle, Julien Canitrot, Esther Poniatowski, Benoît Girard, and Mehdi Khamassi. Model-based and model-free replay mechanisms for reinforcement learning in neurorobotics. *Frontiers in Neurorobotics*, 16, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.