EasyChair Preprint

№ 5456

Curious Exploration and Return-based Memory Restoration for Deep Reinforcement Learning

Saeed Tafazzol, Erfan Fathi, Mahdi Rezaei and Ehsan Asali

May 4, 2021

Curious Exploration and Return-based Memory Restoration for Deep Reinforcement Learning

Saeed Tafazzol[1], Erfan Fathi[1], Mahdi Rezaei[2], and Ehsan Asali[3]

[1] Advanced Mechatronics Researcher Lab, IR. {s.tafazzol,e.fathi}@qiau.ac.ir
[2] Institute for Transport Studies, University of Leeds, UK. m.rezaei@leeds.ac.uk
[3] The University of Georgia, USA. ehsanasali@uga.edu

Abstract. Reward engineering and designing an incentive reward function are non-trivial tasks to train agents in complex environments. Furthermore, an inaccurate reward function may lead to a biased behaviour which is far from an efficient and optimised behaviour. In this paper, we focus on training a single agent to score goals with binary success/failure reward function in Half Field Offense domain. As the major advantage of this research, the agent has no presumption about the environment which means it only follows the original formulation of reinforcement learning agents. The main challenge of using such a reward function is the high sparsity of positive reward signals. To address this problem, we use a simple prediction-based exploration strategy (called Curious Exploration) along with a Return-based Memory Restoration (RMR) technique which tends to remember more valuable memories. The proposed method can be utilized to train agents in environments with fairly complex state and action spaces. Our experimental results show that many recent solutions including our baseline method fail to learn and perform in complex soccer domain. However, the proposed method can converge easily to the nearly optimal behaviour. The video presenting the performance of our trained agent is available at http://bit.ly/HFO_Binary_Reward.

Keywords: Deep Reinforcement Learning · Replay Memory · Sparse Binary Reward · Prediction-based Exploration · Parametrised Action Space · Soccer 2D Simulation · Half Field Offense.

1 Introduction

Machine learning is one of the main sub-categories of AI with applications in various domains such as healthcare [1], autonomous vehicles [2] or robotics systems [3]. The general objective is to teach single or multiple agents to successfully perform a task with a minimum guidance from a human. Reinforcement Learning is a leading paradigm for teaching hard-to-code tasks to robotic agents. A reinforcement learning agent mainly relies on the experience it gains through some trial and error runs. Such an agent focuses on finding a balance between exploration and exploitation and it tries to learn actions with long-term benefits. In other words, the agent repeatedly interacts with an unknown environment with the goal of maximising its cumulative reward [4]. To enable the agent to perceive

rewards from the environment, a human expert has to design a reward function specific to the domain which is subjective to the expert's knowledge about the task. Moreover, the hand-crafted reward function may compromise the goal we truly care about. To address the difficulty with the design of a decent reward function for a reinforcement learning agent, this paper focuses on a problem mentioned in [5] namely, reward engineering. M. Hausknecht and P. Stone propose a method capable of training a single agent to learn how to score on an empty goal from scratch, based on a hand-crafted reward function. Without this reward function, the agent is hopeless to learn anything. In order to overcome such a limitation, we suggest a prediction-based exploration strategy (namely, curious explorer) which is much more successful than a mere random action selection algorithm. Using curious exploration, the agent attempts to perform actions which it has less clue about its outcome (this behaviour is called "curiosity" in animals [6]). This way, the agent is always curious about novel situations while learning from previous experiences. In other words, the exploration will be highly affected by the curiosity of the agent, leading to a better convergence rate to the nearly-optimal behaviour. According to our experiments, the agent will eventually score goals. Our proposed exploration strategy turns the problem of absent positive reward signals into a sparse one. To increase the density of positive reward signals in replay memory, we utilise a different replay memory than the standard one, in which seeks to remember more valuable memories rather than bleak ones.

One of the main contributions of this work is taking advantage of a binary reward function which directly focuses on the task's main objective. Furthermore, note that our work only follows the standard formulation of RL which only relies on the next state as well as reward signal, both coming from the environment.

We use RoboCup 2D Half Field Offense (HFO) domain [7] to perform our experiments. Using the proposed method in [5] as our baseline, we focus on the task of scoring on an empty goal with a binary success/failure reward function. However, instead of a hand-crafted reward signal, we suggest using a binary reward signal. To deal with the challenges in exploiting sparse binary reward signals, we suggest using Curious Exploration as well as Return-based Memory Restoration. The experimental results show that after experiencing sufficiently enough game episodes, the agent can reliably score goals almost every time.

The rest of this paper is organised as follows: the related work is discussed in Section 2; Section 3 explains the background on deep reinforcement learning in parametrised action space and the HFO domain. Section 4 presents our proposed method and finally, Section 6 shows and analyses the experimental results followed by future work and conclusion.

2 Related Work

Experience replay memory [8] plays a vital role in the performance of Deep RL algorithms. Many advancements have been made to improve the efficiency of this component. For instance, Hindsight Experience Replay (HER) [9] saves all

the state transitions of an episode in the replay buffer not only with the original goal of the episode, but also with a subset of other goals. The work uses a binary reward signal and can converge fast to the solution. HER is explicitly designed to handle binary reward which has been rarely used in other works. Nonetheless, it is hard if not impossible to implement HER in adversarial situations.

Another work called Prioritized Experience Replay (PER) [10] that modifies the sampling process of the experience replay memory. PER relies on Temporal Difference (TD) error to prioritise over the memories in the replay buffer. Handling a replay memory involves two decisions; which experiences to store, and which experiences to replay. PER addresses only the latter while our proposed method focuses on the former.

Intrinsic Curiosity Module (ICM) [11] is an exploration method that learns the state space encoding with a self-supervised inverse dynamics model. ICM wisely chooses the features of the state space; i.e., it only considers the features that can influence the agent's behaviour and ignores others. Having the ability to ignore the state factors that the agent has no incentive to learn them enables ICM to solve the noisy-TV problem. Since our environment does not suffer from noisy-TV problem and also the state space prediction in 2D soccer simulation environment is not very complex, in our work, we adhere to the original version of Curious Exploration.

Zare et al. [12] have implemented a goalie cooperating defender's decision making in HFO domain using Deep Deterministic Policy Gradient (DDPG). Their work exploits binary success/failure reward function and predefined hand-coded actions (so called "high-level actions"). Having used the high-level behaviours, the ratio of positive rewards is high enough to learn the task. In our case, we use the original HFO actions (so called "low-level actions"). This way, the agent is not prone to subjective implementation of hand-coded actions and learns the task from scratch, objectively.

3 Background

In this section we provide the fundamental background about Deep Deterministic Policy Gradient (DDPG) [13] (a robust deep reinforcement learning algorithm) and Half Field Offense Domain [7] as the prerequisite for the section Methodology to control our agent(s) in continuous action space of the soccer robot field.

3.1 Deep Deterministic Policy Gradient

This method uses Actor/Critic architecture which decouples the value learning and action selection (Fig. 1). Actor network μ parametrised with θ_μ gets the *state* as input in order to output the desired action. Critic network Q parametrised with θ_Q gets the *state* and *action* as input, then estimates the Q-value for the input action. To update Q-value the critic uses Temporal Difference (TD) [14] equation which defines a critic loss function for neural network setting as follows:

$$L_Q(s, a|\theta^Q) = (Q(s, a|\theta^Q) - (r + \gamma Q(s', \mu(s'|\theta^\mu)'|\theta^Q)))^2 \quad (1)$$

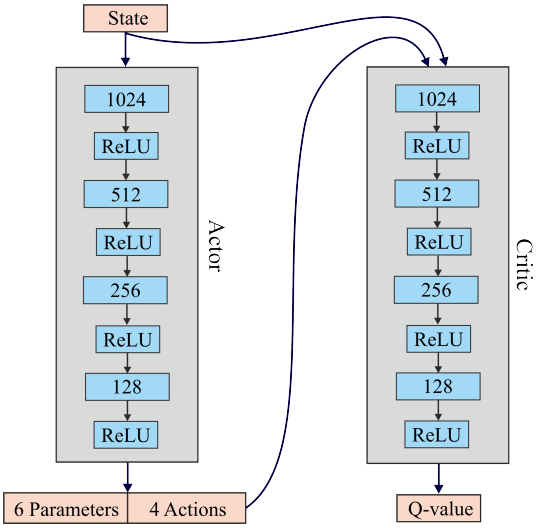

Fig. 1: Actor/Critic network architecture used in parametrised action space. Note that along with continuous parameters, four discrete scalars are used to indicate which action is active [5]

On the other hand, the loss function for the actor model is simply $-Q$ which yields to the following gradient with respect to the actor's parameters:

$$-\nabla_{\theta^\mu} Q(s, a|\theta^Q) = -\nabla_a Q(s, a|\theta^Q) \nabla_{\theta^\mu} \mu(s|\theta^\mu) \tag{2}$$

A Parametrised Action Space Markov Decision Process (PAMDP) [15] is defined by a set of discrete actions, $A_d = \{a_1, a_2, ..., a_k\}$. Each discrete action $a \in A_d$ features m_a continuous parameters, $\{p_1^a, p_2^a, ..., p_k^a\} \in \mathbb{R}^{m_a}$. In this process, an action must be chosen first; then, paired with associated parameters. Soccer 2D simulation [16,17] environment is modelled as a PAMDP with four actions including dash, turn, kick, and tackle while each action has its own parameters.

To take advantage of DDPG, Hausknecht and Stone [5] use a scalar for each discrete action to determine which action must be chosen. Along with the parameters for each action, this constitutes a continuous action space (Fig.1). For example, four discrete scalars for the actions (dash, turn, kick, and tackle) are paired with their parameters (a total of 6 parameters) in HFO domain. As mentioned before, DDPG's critic network utilises TD formula to update the Q-value function. However, there is also an alternative option for updating the Q-value function, called Monte Carlo (MC) [4], which is able to learn online from experience. Nonetheless, opposed to TD-methods, Monte Carlo approaches do not bootstrap value estimates and instead learn directly from returns. In other words, On-policy MC employs on-policy updates without any bootstrapping, while Q-Learning uses off-policy updates with bootstrapping. Both TD and MC try to roughly calculate the action-value function $Q(s, a)$ directly from experience tuples of the form $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$. Furthermore, having visited all state-value pairs an infinite number of times, both methods can provably converge to

optimally. Nevertheless, the fundamental contrast between the two methods can be realised by investigating their update targets. The update target formula for Monte Carlo is defined as follows:

$$\hat{R} = \sum_{i=t}^{T} \gamma^{i-t} r_i \tag{3}$$

and the update target function for Temporal Difference is defined as:

$$q_{target} = r + \gamma Q(s', \mu(s'|\theta^{\mu})'|\theta^{Q}) \tag{4}$$

To stabilise the learning procedure, [18] proposes a mixed update paradigm for critic network in the following form:

$$y = \beta \hat{R} + (1 - \beta) q_{target} \tag{5}$$

where β is a scalar $\in [0, 1]$ indicating the mixture of returned reward in the episode ($\hat{R}$) and the target value for q-learning (q_{target}). In our experiments, we use $\beta = 0.2$. According to the definition of y, we modify the loss function for the mixture of the two algorithms as follows:

$$L_Q(s, a|\theta^{Q}) = (Q(s, a|\theta^{Q}) - y)^2 \tag{6}$$

3.2 Half Field Offense Domain

Half Field Offense (HFO) [7] domain is a simplified soccer 2D simulation task where agents attempt to defend or score goals. HFO features a continuous state space and a parametrised action space where we will explore more in the following sections. In the context of State Space, the agent uses egocentric continuously-valued features. These features include the angle and the distance to important objects in the field such as ball, players, and landmarks. As mentioned before, HFO features a parametrised action space. The mutually-exclusive discrete actions to choose from are as follows:

1. $DASH(power, direction)$ moves in the indicated direction $\in [-180, 180]$ with power $\in [0, 100]$. It must be noted that agent moves faster in the direction that aligns with its body angle compared to other directions.
2. $TURN(direction)$ turns according to the indicated direction $\in [-180, 180]$
3. $KICK(power, direction)$ kicks the ball if the agent is kickable according to power $\in [0, 100]$ and direction $\in [-180, 180]$.
4. $Tackle(power)$ tackles the ball or other players with power $\in [0, 100]$. Note that tackle is usually effective in defensive behaviour and we do not use it in our experiments since we train agents for offensive behaviour.

The reward signal used in [5] is hand-crafted in order to guide the agent through the process of scoring a goal. The reward signal in their work is defied as follows:

$$r_t = d_{t-1}(a, b) - d_t(a, b) + \mathbb{I}_t^{kick} + 3(d_{t-1}(b, g) - d_t(b, g)) + 5\mathbb{I}_t^{goal} \tag{7}$$

where $d_t(a,b), d_t(b,g)$ are respectively the distance of the agent from the ball and the distance of the ball from the goal centre at time t. $\mathbb{I}_t^{kick}$ is a binary variable indicating that the agent has touched the ball for the first time at time t.$\mathbb{I}_t^{goal}$ is also a binary variable indicating that the agent scored a goal at time t.
However, for our research we only use the last term or more precisely:

$$r_t = 5\mathbb{I}_t^{goal} \tag{8}$$

We show that our method, unlike the one in [5], is able to learn the task of scoring goals with this new reward signal formulation.

4 Proposed Method

Reinforcement learning is hopeless to learn anything without encountering a desirable reward signal in exploration phase [4]. In our case, due to the fact that a random agent never scores a goal, by using a success/failure reward signal, our baseline article [5] fails to learn the task. To overcome this issue, we use a prediction-based exploration strategy called "Curious Exploration". However, this exploration strategy degrades the problem of completely absent positive reward signal into a sparse one. To deal with the sparsity, we modify the replay memory in a way that it tends to remember more promising memories and we call it "Return-based Memory Restoration". Complete source code for our agent is available at https://github.com/SaeedTafazzol/curious_explorer. In the following subsections, we will describe the two mentioned approaches in more detail.

4.1 Exploration and Exploitation Agents

Our framework takes advantage of two internal agents that take responsibility to perform either the exploration or the exploitation task. Indeed, one agent only focuses on the exploration by looking for novel states while the other agent concentrates on exploitation attempting to score goals. However, since only one of these two agents can be active at a time, the soccer agent has to follow a procedure to switch between the two. A widely used policy to obtain a satisfactory trade-off between exploration and exploitation is $\epsilon - greedy$ [4] where ϵ indicates the probability of taking a random action. In our framework, ϵ is the probability of activating the exploration agent. Hausknecht and Stone [5] anneal ϵ from 1.0 to 0.1 over the first 10,000 updates in the experiments. Nevertheless, this annealing function only works well with a hand-crafted incentive reward function. Regarding the fact that our reward signal is a binary function which is quiet different from our baseline's reward function, we use a different annealing function which anneals the amount of exploration by a constant factor every time a positive reward signal is gained. This choice of annealing function relies on the fact that a reinforcement learning agent cannot learn anything with a dead reward signal. In other words, it is pointless to expect the agent to perform exploitation when it has not seen any reward signal other than zero.

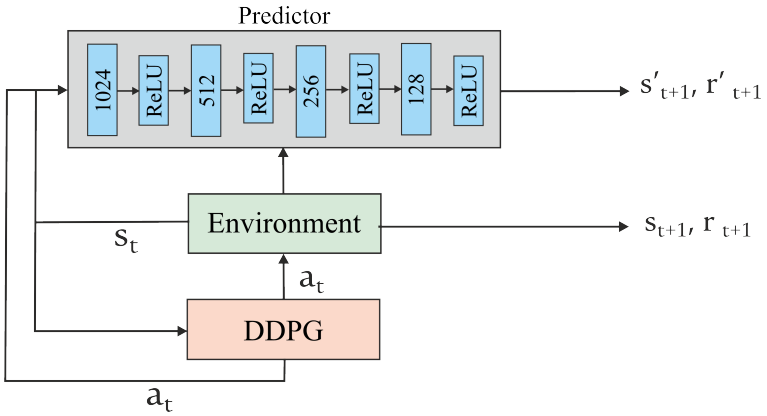

Fig. 2: The overall structure of Curious Exploration (CE) algorithm

4.2 Curious Exploration (CE)

Exploration in reinforcement learning assists the agent to observe new patterns which can lead to a better learned policy [4]. Random action selection is a commonly used exploration approach; however, it is highly probable to visit previously seen states while exploration. Therefore, exploring big environments is problematic using random selection. As an example, a random moving agent in the space has the expectation of staying at the same place. Therefore, to visit more novel states, we should design a proper exploration strategy.

The first challenge to solve is to come up with a method to identify and differentiate the novel states from previously seen states in continuous state-space. One potential solution is to have a model responsible for learning the forward dynamics of the environment. This way, if the model can predict the future state, we can assume that the agent has previously been in a similar situation and vice versa. Note that the prediction accuracy is a good metric to examine the performance of the learned forward dynamics model. To formulate the learning procedure, we exploit the formulation of Markov Decision Process for reinforcement learning. To elaborate more on this, lets assume the agent is at s_t attempting to take the action a_t. Performing the given action, the agent's predictor should be able to identify s'_{t+1} and r'_{t+1} as the foreseen values for s_{t+1} (the next state) and r_{t+1} (the upcoming reward), respectively. The predictor P then has the following loss function:

$$L_P = ||(P(s_t, a_t) - (s_{t+1}, r_{t+1}))||^2 \tag{9}$$

where (s_{t+1}, r_{t+1}) is the concatenation of the next state and the upcoming reward vectors. Such a loss function formulation, which is only dependent to the transitions, enables the predictor to learn the model in a supervised manner. Having an evaluation metric to measure the status novelty (L_P), we can now add the exploration component to our framework. To do so, we can use the predictor's loss function as a reward signal (which is called "intrinsic reward") for an explorer agent (Fig.2). Opposed to the prediction model, which is a regression problem,

reaching a novel state may require the agent to perform a series of actions (not necessarily one action). This means that the exploration process is basically a reinforcement learning problem that tends to reach the desired outcome in the long run. Having that in mind, we can calculate intrinsic reward in the following form:

$$r_t^i = L_P \tag{10}$$

Interestingly enough, animals use the same notion to explore their world more efficiently. More precisely, they like to examine things which they are unsure of the outcome; this intention in animals is called "Curiosity" [19]. The reason this method closely follows the curiosity behaviour is that our agent tries to perform actions with unknown outcomes. We show that this method guides the agent to score goals during the training process which leads to generate numerous promising memories. The experimental results show that such amount of promising memories is sufficient for the agent to learn how to score goals.

4.3 Return-based Memory Restoration (RMR)

The utilisation of curious exploration may lead to some memories with positive reward signals; however, the memories may be too sparse for the agent to properly learn a difficult task in complex environments. For example, the soccer agent may only score a goal every few episodes.

In order to deal with the sparsity of the memories with positive rewards, we modify the structure of the replay memory to motivate it to remember more valuable memories. To do so, one potential solution is to use the reward saved in each transition as an evaluation metric. Nevertheless, this may not perform well since there exist just a few memories with positive reward signal throughout the experiment. Alternatively, we exploit the total discounted reward (return) or more precisely $\sum_{i=t}^{T} \gamma^{i-t} r_i$. Having the return value, we can calculate the probability of forgetting the evaluated memory (f):

$$f = e^{-\alpha(return)} \tag{11}$$

where α is a hyper-parameter indicating how forgetful our replay memory is. Increasing the α value leads to a less forgetful replay memory. In our experiments, we set $\alpha = 1$. To utilise this formula, we use a modified architecture of experience replay memory rather than its original version [8]. The original replay memory architecture enjoys a queue (FIFO) to store the memories and removes the memory at the head of the queue without any exception. However, RMR will examine the memory at the head location and calculates the probability of forgetting f it. Then, with probability of f the memory is forgotten and with probability of $1 - f$ the memory is restored back to the tail of the queue (Fig.3). Below, Alg.1 1 represents the Pseudo-code of the RMR's memory handling algorithm:

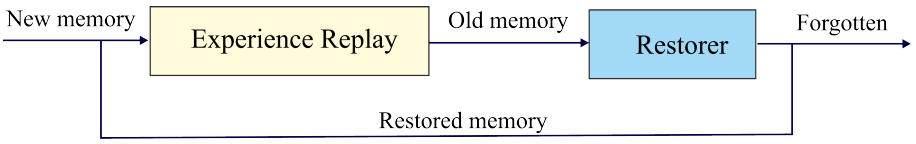

Fig. 3: The architecture of Return-based Memory Restoration (RMR)

Algorithm 1: Adding a new memory to RMR's replay memory

Input: new_memory
oldest_memory = queue.pop();
while $e^{-\alpha(oldest\_memory.return)} < random(0,1)$ **do**
  queue.add(oldest_memory);
  oldest_memory = queue.pop();
end
queue.add(new_memory);

5 Experimental results

We evaluated the performance of the proposed method in the parametrised HFO domain for the task of approaching the ball and scoring a goal. Three types of agents were considered as follows:

- The baseline agent which using epsilon-greedy with random action selection (DDPG).
- The baseline agent enhanced with curious exploration (DDPG + CE).
- The baseline agent enhanced with curious exploration and Return-based Memory Restoration (DDPG + CE + RMR)

We used binary reward formulation for all three types of the agents. For each agent, we conducted three independent train and test experiment. Each training and testing rounds take 50,000 and 1,000 episodes, respectively. Each training round took almost six hours on a Nvidia GeForce GTX 1070 GPU platform.

The first agent, i.e. the standard DDPG failed to learn anything and never scored a single goal using binary reward function. Note that DDPG agent can perform the task with nearly-optimal performance using a hand-crafted reward function. In contrast, both DDPG + CE and DDPG + CE + RMR agents could reliably learn to score goals. It can be seen that DDPG + CE + RMR achieves a better performance in terms of the gained rewards per episode (Fig.4).

As represented in Fig.4, DDPG + CE + RMR can converge faster to the optimal behaviour compared to DDPG + CE which is because of paying attention to more valuable and promising memories (maintained through RMR). In fact, the increased performance of DDPG + CE + RMR model is probably because of remembering earlier memories with positive return reward. Indeed, the agent without RMR fails to learn much in first episodes in which positive reward signal

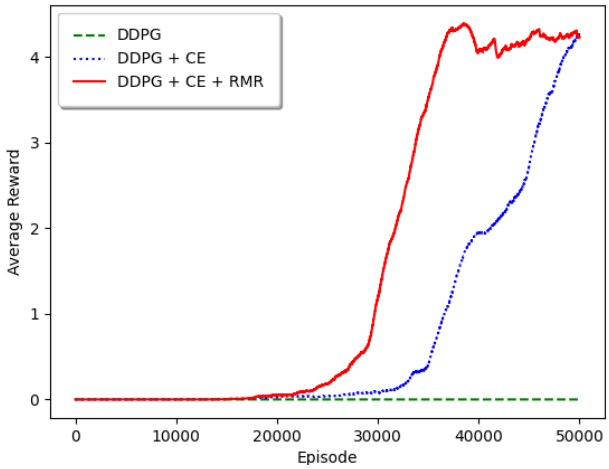

Fig. 4: A comparison of average rewards among the three types of agents. The rewards are averaged over five rounds of experiments for each agent.

	DDPG	**DDPG + CE**	**DDPG + CE + RMR**
Round 1	0.000	0.978	0.989
Round 2	0.000	0.977	0.988
Round 3	0.000	0.917	0.986
Round 4	0.000	0.867	0.964
Round 5	0.000	0.858	0.963
Avg	0.000	0.919	**0.978**

Table 1: The success rate achieved by the agents examined in our experiments. As expected, DDPG + CE + RMR agent shows the best performance among the three.

is seen due to the fact that it quickly forgets them. On the other hand, by using RMR, sparse positive signals become more dense in replay memory. However, when the positive reward signal becomes dense enough, the agent without RMR also learns how to score goals in our environment. In contrast, this may not be the case when the agent has to deal with a more complex environment. In other words, reward signal may stay sparse after exploration when the task is hard to be learned.

As mentioned before, we ran 5 rounds of experiment (training and testing) for each type of agent and calculated the success rate per round. Table 1 shows the success rate achieved by each agent in each round as well as the average result acquired by the agent. According to the results, DDPG completely fails to do the task; DDPG + CE can achieve %91.9 success rate and DDPG + CE + RMR outperform both other methods by almost a margin of %6.

For a further analysis on Curious Exploration, the predictor's loss and explorer's reward gain are plotted (Fig. 5). As can be seen, higher reward gain of the explorer (Fig.5.a) corresponds to the higher prediction error (Fig.5.b). The first spike in predictor's loss (after learning basic dynamics) happens when the agent starts interacting with the ball. After noticing a strange behaviour from

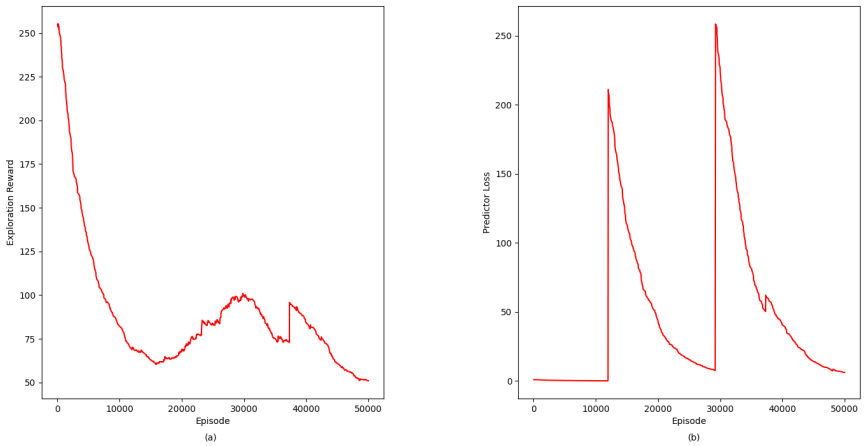

Fig. 5: a) Exploration reward graph gained by the exploration agent; b) Prediction loss graph that shows two spikes revealing the time that the agent first intercepts the ball and when the agent scores a goal for the first time

the ball (e.g. when the ball is kicked), the agent becomes extremely curious about this object. By interacting and examining this object while trying to learn its dynamics, it will eventually score goals. Now, because it did not know about positive reward signal gained by scoring a goal, it becomes curious about scoring. Through this process, the amount of exploration is annealed towards exploitation.

6 Conclusion and Future Work

This paper presented a framework capable of learning how to score on an empty goal in HFO domain with only a binary success/failure reward using a combination of DDPG, Curious Exploration, and Return-based Memory Restoration methods. The research showed how the training process can be improved by exploiting Return-based Memory Restoration which seeks to remember more valuable memories. The experimental results confirmed our proposed method can achieve nearly-optimal behaviour while the baseline method entirely failed to learn any tasks.

The prediction-based exploration method, used in this article, is prone to noisy-TV problem. In our environment, however, we did not face such an issue; but, this may change with more complex situations where it may include stochastic agents. To deal with this challenge, we suggest applying ICM [11] to stabilise the exploration process.

Furthermore, the behaviour of RMR must be further examined in more complicated environments. An interesting possibility is to use RMR (which modifies the memory maintenance procedure in experience replay memory) along with a state-of-the-art sampling method such as Prioritized Experience Replay (PER). The learning performance of the method could be also investigated for more complex tasks using only a binary success/failure reward function.

References

1. Mahdi Rezaei and Mohsen Azarmi. Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic. *Applied Sciences*, 10(21):7514, 2020.
2. Mahdi Rezaei, M Sarshar, and M.M. Sanaatiyan. Toward next generation of driver assistance systems: A multimodal sensor-based platform. In *Int Conf. on Comp and Automation Engineering (ICCAE)*, volume 4, pages 62–67, 2010.
3. Ehsan Asali, Farzin Negahbani, Saeed Tafazzol, Mohammad Sadegh Maghareh, Shahryar Bahmeie, Sina Barazandeh, Shokoofeh Mirian, and Mahta Moshkelgosha. Namira soccer 2d simulation team description paper 2018. *RoboCup 2018*, 2018.
4. Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.
5. Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.
6. Daniel E Berlyne. Curiosity and exploration. *Science*, 153(3731):25–33, 1966.
7. Matthew Hausknecht, Prannoy Mupparaju, Sandeep Subramanian, Shivaram Kalyanakrishnan, and Peter Stone. Half field offense: An environment for multiagent learning and ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*. sn, 2016.
8. Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.
9. Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.
10. Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
11. Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR, 2017.
12. Nader Zare, Mahtab Sarvmaili, Omid Mehrabian, Amin Nikanjam, Seyed Hossein, Aref Sayareh Khasteh, Omid Amini, Borna Barahimi, Arshia Majidi, and Aria Mostajeran. Cyrus 2d simulation 2019.
13. Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
14. Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.
15. Warwick Masson, Pravesh Ranchod, and George Konidaris. Reinforcement learning with parameterized actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.
16. Hidehisa Akiyama. Agent2d base code, 2010.
17. Hidehisa Akiyama and Tomoharu Nakashima. Helios base: An open source package for the robocup soccer 2d simulation. In *Robot Soccer World Cup*, pages 528–535. Springer, 2013.
18. Matthew Hausknecht and Peter Stone. On-policy vs. off-policy updates for deep reinforcement learning. In *Deep Reinforcement Learning: Frontiers and Challenges, IJCAI 2016 Workshop*, 2016.
19. Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.