A framework of dual replay buffer: balancing forgetting and generalization in reinforcement learning*

Linjing Zhang[1†], **Zongzhang Zhang**[2], **Zhiyuan Pan**[1†], **Yingfeng Chen**[3], **Jiangcheng Zhu**[3], **Zhaorong Wang**[3], **Meng Wang**[3], **Changjie Fan**[3]

[1]School of Computer Science and Technology, Soochow University, Suzhou, China

[2]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

[3]Fuxi AI Lab, Netease, Hangzhou, China

{zlinjing728, zhangzongzhang, owenpzy}@gmail.com

{chenyingfeng1, wangmeng02, fanchangjie}@corp.netease.com

Abstract

Experience replay buffer improves sample efficiency and training stabilization for recent deep reinforcement learning (DRL) methods. However, for the first-in-first-out (FIFO) retention widely used in plain experience replay buffer, forgetting and generalization are problems in long-time training due to the outflow of some experiences, especially in limited buffer size. With the training progressing and the exploration reducing, experiences generated by the learned policy are narrowed regions of the state space, leading the policy to further fit the current experiences and forget the knowledge obtained from previous experiences. In this paper, we propose a reservoir sampling double replay buffer (RS-DRB) framework to alleviate "forgetting" problem, which can be represented by the generalization of the policy. In the RS-DRB framework, experiences are stored into one of the two buffers, i.e., the buffers for exploration and exploitation, according to its exploration, then experiences used for training are sampled from the two buffers with different retention policies. We design an adaptive sampling ratio between the two buffers to balance the distribution of the state space. Empirical results show that RS-DRB gains better training and generalization performance than FIFO and some other retention policies.

1 Introduction

Deep reinforcement learning (DRL) combines the perception ability of deep learning (DL) with the decision-making ability of reinforcement learning (RL) [François-Lavet *et al.*, 2018], and has achieved great success in many fields, from classic physical control problems, such as DC motor [Babuška and Groen, 2010], to more complex environments, such as Atari games [Mnih *et al.*, 2015] and Go [Silver *et al.*, 2016; Silver *et al.*, 2017]. However, there are some problems when directly combining these two powerful methods. The stochastic gradient optimization algorithms used in DL require data to satisfy the i.i.d. assumption, while the data sampled by the RL agent are successive and strongly correlated. In order to resolve this contradiction, the technology of replay buffer [Lin, 1992] was used in DRL. Generally Speaking, experiences, usually donated (s, a, r, s'), generated by interacting with the environment are stored into a buffer, and algorithms samples a batch of transitions uniformly from the buffer for its training. The experience buffer not only solves the above problems, but also improves the efficiency of the experience and the stability of the DRL algorithm [Mnih *et al.*, 2015; Lillicrap *et al.*, 2016].

Since the replay buffer has been successfully applied to DRL, researchers have made improvements of replay buffer. The experience sampling and retention are two core functions of replay buffer, which determine the experiences required for training. Prioritized experience replay (PER) [Schaul *et al.*, 2016] focuses on the instantaneous utility of experiences and implements the prioritized sampling in replay buffer based on the temporal difference error (TD-error). While the retention method is related to the long utility of experiences, which helps to prevent insufficient coverage of the state space and improves the generalization of learned policy. In [De Bruin *et al.*, 2015; de Bruin *et al.*, 2016a; de Bruin *et al.*, 2016b], they add experiences to two replay buffers with first-in-first-out (FIFO) and a distance-based retention policy, and use synthetic experiences to balance the experience distribution between the two distributions respectively generated by the current policy and a uniform policy. They also investigate some proxies to guide the retention and sampling of replay buffer via prior knowledge on control problems [de Bruin *et al.*, 2018]. Hindsight experience replay (HER) in [Andrychowicz *et al.*, 2017] provides another approach to deal with sparse and binary rewards. Its key idea is to learn from failure, actually adding implicit intermediate goals to the replay buffer to facilitate learning.

In this paper, we focus on the retention method, which is one of the core functions in replay buffer. When replay buffer is full, the retention policy decides which experiences should

*This work is in part supported by the National Natural Science Foundation of China under Grant No. 61876119, and the Natural Science Foundation of Jiangsu under Grant No. BK20181432.

†This work was done when interning at Fuxi AI Lab, Netease.

be replaced by new experiences. Almost all replay buffers mentioned above use the FIFO retention. It requires the agent to maintain exploration to preserve the sample diversity in the buffer [de Bruin *et al.*, 2016a]. However, RL algorithms usually use a gradual reduction in exploration in order to accelerate achieving an optimal policy. With the decrease of exploration rate and the improvement of learned policy, the experiences generated by the current policy begin to focus on some certain regions rather than the whole state space, reducing the diversity. What is more serious is that it greatly increases the possibility of "catastrophic forgetting" [Goodfellow *et al.*, 2013], i.e., forgetting the knowledge gained from previous learning. It is because the experiences about the previous knowledge are washed out of the replay buffer over time by the FIFO retention, especially for a small size buffer. It is worth working on holding back the outflow of useful samples to prevent the occurrence of the forgetting problem.

The main contribution of this paper is the introduction of the Reservoir Sampling Double Replay Buffer (RS-DRB) framework to alleviate the "forgetting" problem. The new framework uses a straightforward way to divide experiences into two replay buffers, then maintains the state distribution of the entire experiences by different retention policies in two buffers, and finally samples the experiences from two buffers via an adaptive ratio. In the GridWorld experiment, we use the generalization ability of the policy to represent the problem of forgetting. The RS-DRB framework can obviously alleviate this problem. We also compare the different retention policies on more complex problems, and the results also show that our framework is better.

2 Background

As a separate module, the RS-DRB framework can be combined with different RL algorithms according to different problems. In the experiments of this paper, we use the deep Q-network (DQN) [Mnih *et al.*, 2015] as a baseline algorithm in discrete action problems, and in continuous action problems, we use deep deterministic policy gradient (DDPG) [Lillicrap *et al.*, 2016] as a baseline algorithm for comparison.

2.1 Reinforcement Learning

RL is to train an agent by interacting with an environment. The environment can be simply described by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and a discount factor γ. The goal of the RL agent is to find the policy π^* that maximize the return defined as

$$V_\pi(s) = \Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) | s_0 = s\Big].$$

The Q-function under the policy π is defined as

$$Q_\pi(s, a) = \Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\Big].$$

The Q-function under the optimal policy π^*, denoted Q^*, satisfies the Bellman optimality equation [Bellman, 1958]:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s, a, s') \max_{a' \in \mathcal{A}} Q^*(s', a').$$

2.2 Deep Q-Network

DQN is an end-to-end DRL algorithm whose input is a state and output is the Q-value of each discrete action in this state. In order to break the correlation between the experiences, experiences obtained from interaction with the environment will be first placed in the replay buffer $\mathcal{D}$. When the network updates, it samples a batch of experiences drawn uniformly from the replay buffer, i.e., $U(\mathcal{D})$, and minimizes the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim U(\mathcal{D})}[(y_t - Q(s, a; \theta))^2],$$

where the target value $y_t = (r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta^-))$, θ represents parameters of the current Q-network, and θ^- represents parameters of the target Q-network.

2.3 Deep Deterministic Policy Gradient

DDPG is also an end-to-end DRL algorithm whose input is a state and output is a deterministic action, and can be applied on continuous action problems. In DDPG, two neural networks are required to update: an actor network $\pi : \mathcal{S} \to \mathcal{A}$, and a critic network $\mathcal{Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The critic is updated to minimize the loss function. Different with DQN, the target value is calculated using the action that comes from the actor: $y_t = r + \gamma Q(s', \pi(s'))$. The actor is trained in the direction that maximizes the expected Q-values $\mathbb{E}_\pi[Q(s, \pi(s))]$.

3 Method

The RS-DRB framework, as shown in Algorithm 1, is a combination of double buffers, reservoir sampling and adaptive sampling ratio. Intuitively, exploring actions tends to a uniform policy which may search the entire state space, while exploiting actions tends to the learned policy. So, exploration buffer $\mathcal{D}_r$ stores the transitions according to exploration actions. Exploitation buffer $\mathcal{D}_g$ stores the transitions generated by exploitation actions. Then the training batch is sampled from the above two buffers with a ratio adaptive to the policy update rate. We implement reservoir sampling, which guarantees that all stored experiences are equally sampled or removed to better maintain the coverage of the state space.

3.1 Reservoir Sampling

The idea of reservoir sampling [Vitter, 1985] is to sample k elements from a set of n elements with equal probability, where n is very large or unknown. This method can be transformed into a retention method of replay buffer, that is, the replay buffer of size k is maintained (sampled) in all experiences generated by interaction and experiences in replay buffer have equal probabilities to be overwritten by new experiences.

Denoting the set of all generated experiences as $\mathcal{D}_a$. The algorithm creates a reservoir of size k and fills it with the first k elements of $\mathcal{D}_a$. It then iterates through the remaining elements of $\mathcal{D}_a$ until exhausted. When the i-th ($i > k$) element of $\mathcal{D}_a$ is generated, it has a k/i probability added to the reservoir. If the i-th element has added to the pool, then it has the probability $1/k$ of being selected for replacement. When the $(i+1)$-th element tries to add into the reservoir with the probability $k/(i + 1)$, the probability that the i-th element is not

Algorithm 1 The RS-DRB framework

Input: buffer size k, sampling ratio τ, exploration threshold η, interval time C for updating target network

Initialize two replay buffers $\mathcal{D}_r$ and $\mathcal{D}_g$ with size $k/2$
Initialize an RL algorithm $\mathbb{A}$
for episode = 1, 2, $\cdots$, M **do**
  for t = 1, 2, $\cdots$, T **do**
    Sample an action a_t using current policy from $\mathbb{A}$
    if exploration rate $> \eta$ **then**
      Store transition (s_t, a_t, r_t, s_{t+1}) into $\mathcal{D}_r$
    else
      Store transition (s_t, a_t, r_t, s_{t+1}) into $\mathcal{D}_g$
    end if
    Generate the training batch $\mathcal{D}_s$ from $\mathcal{D}_r$ and $\mathcal{D}_g$ according to τ
    Perform optimization update using $\mathbb{A}$ and batch $\mathcal{D}_s$
    After C updates, update the target network and τ
  end for
end for

replaced by the $(i+1)$-th element is $1 - 1/(i+1)$. By analogy, we can get the probability formula that the i-th element is retained in the reservoir:

$$\begin{aligned} P[(s,a,r,s')_i] &= \frac{k}{i} \times \prod_{n=1}^{S(\mathcal{D}_a)-i} \Big(1 - \frac{k}{i+n} \times \frac{1}{k}\Big) \\ &= \frac{k}{i} \times \prod_{n=1}^{S(\mathcal{D}_a)-i} \Big(\frac{i+n-1}{i+n}\Big) \\ &= \frac{k}{S(\mathcal{D}_a)}, \end{aligned}$$

where $S(\mathcal{D}_a)$ represents the size of $\mathcal{D}_a$. We can find that the probability of each element retained in the reservoir is only related with the sizes of reservoir and $\mathcal{D}_a$. When the buffer is full, FIFO washes out the earliest experiences, while all experiences in the buffer that use the reservoir sampling method have equal probabilities to outflow. That is to say, the reservoir sampling method is more likely to retain early experiences than the FIFO method.

Since experiences of early thorough exploration can better cover the entire state space, the retention method of reservoir sampling may help to maintain the state distribution of the replay buffer.

3.2 Double Replay Buffers

To search the entire state space for a global optimal policy, exploration is necessary. In discrete action problems, an ϵ-greedy policy can control the magnitude of the exploration and the parameter ϵ is the probability of using the random policy. While in continuous action problems, the noise $\mathcal{N}$ is often used to drive exploration. Therefore, we can set a threshold η, to determine whether the action belongs to exploration action a_r or exploitation action a_g.

Then we can denote replay buffer for exploration as $\mathcal{D}_r = \{(s, a_r, r, s')\}$. This buffer stores the transitions from the exploration rate greater than the threshold. In this buffer we implement reservoir sampling for the overwriting purpose. The

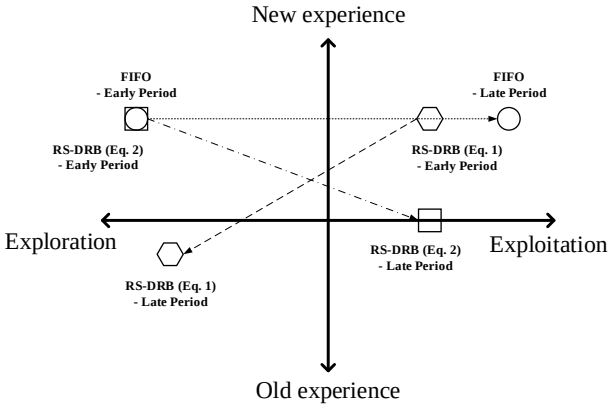

Figure 1: The stream of state distribution of training batch

second buffer for exploitation as $\mathcal{D}_g = \{(s, a_g, r, s')\}$ is to store transitions generated by the greedy policy. This buffer is overwritten by standard FIFO methods. In this paper, the entire buffer $\mathcal{D}$ is the combination of $\mathcal{D}_g$ and $\mathcal{D}_r$, and either $\mathcal{D}_g$ or $\mathcal{D}_r$ has a half size of $\mathcal{D}$.

In early period of training process, experiences in $\mathcal{D}_r$ and $\mathcal{D}_g$ spread across almost the entire state space. When exploration rate decreases and the policy improves, the experiences generated by the current policy are narrowed to some regions of state space, such as one trajectory. In order to maintain the coverage of the entire state space, we implement reservoir sampling in $\mathcal{D}_r$ to preserve more early experiences for maintaining the entire state distribution. On the other hand, $\mathcal{D}_g$ can still focus on the current policy with the FIFO method.

3.3 Sampling Ratio

Sampling ratio τ is a variable to control the proportion of experiences in a training batch sampled from two replay buffers. It is a parameter adaptive to policy change and the policy change is empirically obtained by comparing actions from two existing networks in DQN and DDPG: the current network and the target network.

Given some training batches of experiences with size N_b, we can obtain two sets of actions separately according to current network and target network. By counting the number of the same actions n_b from two sets of actions, we can adjust the sample ratio τ as:

$$\tau = \frac{n_b}{N_b} \times \mathcal{T}_{\max}, \tag{1}$$

where $\mathcal{T}_{\max} \in [0, 1]$ is a hyper-parameter to control the upper bound of τ and the update of τ is the same as the frequency of the target network. In each training batch with size N_b, τN_b experiences are sampled from exploration buffer $\mathcal{D}_r$ and the rest ones are sampled from exploitation buffer $\mathcal{D}_g$.

Directly using Eq. (1) as the sampling ratio may cause a problem: in the early training period, the current network may change frequently, which results in n_b is not very large and τ is small. Therefore we consider the exploration rate ϵ into the sample ratio to guarantee thorough exploration:

$$\tau = \max\Big\{\epsilon, \frac{n_b}{N_b} \times \mathcal{T}_{\max}\Big\}. \tag{2}$$

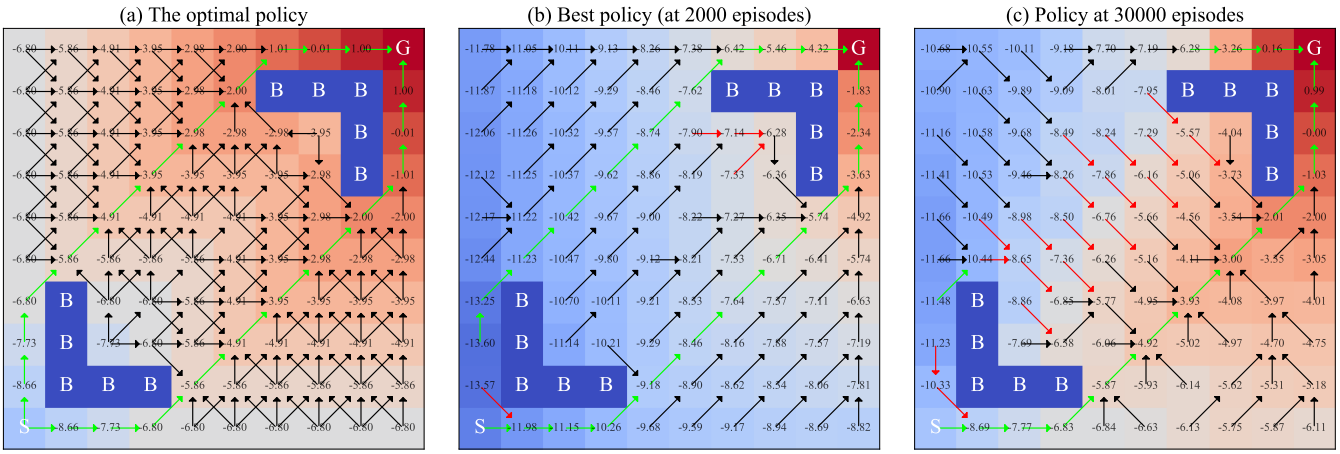

Figure 2: A GridWorld example of the forgetting problem caused by the FIFO retention method

Now according to Eq. (2), in the early training period, τ is dependent on ϵ rather than Eq. (1). In the late training period, the exploration rate ϵ becomes very small and the policy is stable, two sets of actions are almost identical. Since we have retained early experiences of exploration by reservoir sampling to prevent forgetting, we hope to sample some experiences from exploration buffer $\mathcal{D}_r$. Therefore, Eq. (1) contributes more in the later learning.

3.4 Insight

The catastrophic forgetting problem that the neural network forgets the knowledge gained from previous learning is mainly due to the fact that the experiences about the previous knowledge are washed out of the replay buffer over time, especially with a small size buffer. We show how the RS-DRB framework rearranges the state distribution to maintain the coverage of the state distribution. The state distribution of the sampled training batch is represented on a 2D coordinate: exploration-exploitation, old experience-new experience, as in Figure 1. We use dots in different patterns to refer each buffer retention policy. An arrow connecting two dots refers to the stream of the state distribution.

In the training process of the RL algorithm $\mathbb{A}$, the role of the FIFO retention is actually a sliding window. So in Figure 1, FIFO represented by the circle always tends to be new experiences. On the other hand, from the beginning to the end of the training, because exploration rate ϵ is gradually reduced, the replay buffer gradually moves from the exploration to the exploitation.

In the RS-DRB framework, we discuss two streams of the state distribution based on Eq. (1) and Eq. (2), which are represented by hexagon and square separately in Figure 1, and the effects of parameter $\mathcal{T}_{\max}$ on streams.

According to Eq. (1), because the policy changes greatly in the early period, the sampling ratio is small and replay buffers are more likely to sample from the exploitation buffer, it may cause inadequate exploration in early period. In the later period, as the policy changes smaller, the sampling ratio gradually increases to a value close to $\mathcal{T}_{\max}$. Therefore, it will be more inclined to exploration buffer. At the same time, due to the reservoir sampling retention, a part of new experiences are added to the exploration buffer with a certain probability rather than probability 1. In this stream, we set $\mathcal{T}_{\max}$ to be very large, and then there are more experiences sampled from the exploration buffer in a training batch.

When using Eq. (2), we set a relative small value of $\mathcal{T}_{\max}$. At the early period, the parameter τ is the same as the exploration rate ϵ in FIFO. At the later period, the change of the state distribution is the same as the above due to the reservoir sampling – new experiences and old experiences in the replay buffer are more balanced and experiences are likely more evenly distributed across the state distribution. Meanwhile, the exploration rate ϵ becomes very small and τ is dependent on $\mathcal{T}_{\max}$ because the target network and the current network have almost the same policy. At last, the training batch sampled from the two buffers can effectively reduce the probability of the forgetting problem.

4 Experiments

In this section, we compare our RS-DRB framework with some other replay buffer retention methods. We first use DQN to train the policy on a designed GridWorld problem and show that generalization of the policy can represent the forgetting problem. Then, we conduct experiments using DDPG on the Pendulum task as in [de Bruin *et al.*, 2016a] to illustrate the issue of forgetting problem.

4.1 GridWorld

We design a simple GridWorld with 10×10 size. The action space has eight directions, referring the step ahead to the next grid. The two-dimensional coordinate position of each small grid is to represent all states in the environment. Each step the environment gives agent a reward of -1 until it reaches the terminate state. We set some blocks (denoted B, dark blue) in the map to increase the difficulty of finding the optimal policy as shown in Figure 2(a). And we also show the true state value and optimal direction of each grid. The grid colors change

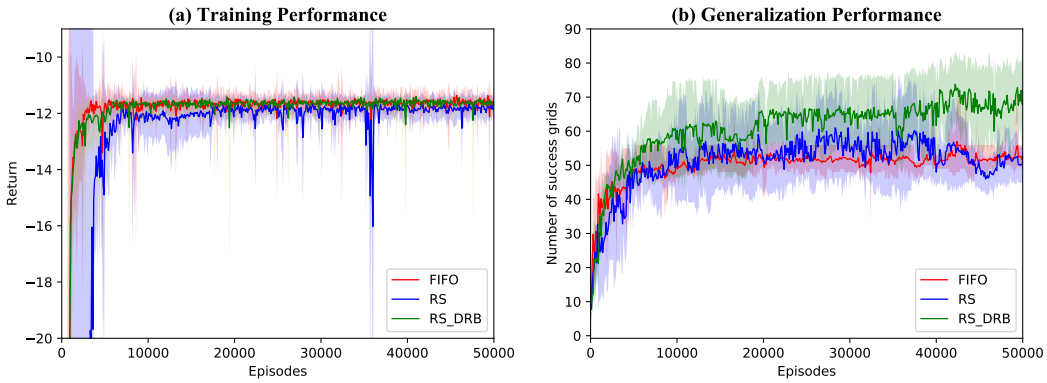

Figure 3: The performance on GridWorld

from blue to red as their state values increase. It is obvious that the optimal policy from the starting point at left-lower corner (denoted S) to the goal point at right-upper corner (denoted G) consists of two trajectories, which are represented by green arrows and these two trajectories are identical in terms of return.

At the beginning, due to the thorough exploration, experiences can cover all the state space. Therefore the agent has learned an almost global optimal policy. In Figure 2(b), the agent at each grid can reach the goal. We can find that the agent prefers the trajectory to the bottom right, and it still remembers the optimal trajectory to the upper left. When the exploration decreases and the neural network refines the policy, the experiences generated by the current policy begin to focus on the preferred trajectory rather than the global optimal policy and the experiences about another optimal trajectory may wash out of the replay buffer by FIFO. In Figure 2(c), red arrows represent the wrong directions. It can be seen that the agent at some grids on the upper left trajectory goes away the shortcut, i.e., first to the optimal trajectory on the lower right, and then to the goal. It has forgotten the knowledge about the optimal trajectory on the upper left.

Configuration

We use DQN which has two hidden layers with 16 and 32 units as our RL algorithm, the parameter τ and the target network are updated per 100 steps. The exploration rate ϵ decays linearly from 1 to 0.1 during 1000 episodes. We totally train 50000 episodes in one trial and evaluate the generalization of the current policy every 500 episodes. The final result is an average of 5 trials. We compare the three retention methods of replay buffer: FIFO, reservoir sampling (RS) and RS-DRB and all of them utilize the uniform sampling method to sample experiences from the replay buffer. The size of replay is 16000 in FIFO and RS, and in RS-DRB, each buffer size is 8000. The hyper-parameter $\mathcal{T}_{\max}$ in GridWorld is set to 0.2.

Metrics

We evaluate the policy through two performances. The training performance is reflected by the cumulative reward from the starting point to the goal. As mentioned above, two optimal trajectory have the same cumulative reward. Therefore, this performance can only show that the agent has found one optimal trajectory, but it cannot show the performance of the agent on the whole GridWorld. The generalization performance is used to supplement the deficiency of the training performance and can quantify the severity of the forgetting problem in this problem. More specifically, we count the number of the grids with the optimal actions (we call them success grids) after training every 500 episodes. This is challenging because the agent must accurately remember the action on each grid, otherwise it will not be able to reach the goal with the smallest steps.

Performance

In Figure 3(a), three retention methods do not affect the agent to find one optimal trajectory. The agent with FIFO or RS-DRB finds one optimal trajectory faster than the agent with RS. Because the RS method retains all the experiences with equal probabilities, it cannot focus on the learned policy. So the agent with the RS retention requires more episodes to find one optimal trajectory. In Figure 3(b), it can be seen that the agent with FIFO or RS only makes almost a half of grids arrive the goal with the shortest trajectory. While RS-DRB can make the agent arrive the goal with the shortest trajectory in more grids than FIFO and RS. This shows that the RS-DRB framework improves the generalization of the policy and alleviates the forgetting problem.

4.2 Pendulum

Pendulum swing-up is a classic continuous control problem. We use the pendulum task from the classic control environment in OpenAI Gym [Brockman *et al.*, 2016]. In this setting, the state s is represented by the angle $\theta \in [-\pi, \pi]$ and the angular velocity $\dot{\theta} \in [-8, 8]$. The action $u \in [-2, 2]$ is the voltage applied to the motor that drives the torque on the pendulum. The reward function is related to the voltage and the relative angle between the current position and the upward equilibrium position. When the pendulum position is vertically upward in equilibrium and the voltage is zero, the agent receives the maximum reward, which is zero. Meanwhile, the dynamic of environment which controls the pendulum cannot reach the equilibrium position directly by applying the maximum voltage, but needs to swing to the upright equilibrium position by swinging left and right.

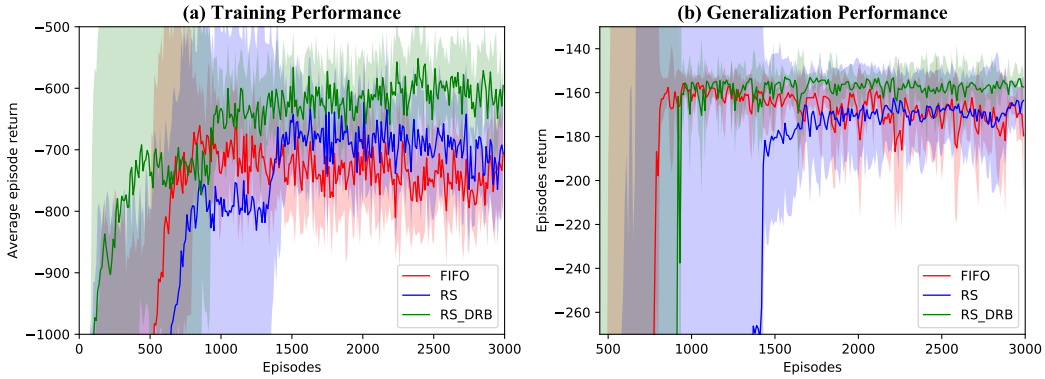

Figure 4: The performance on Pendulum

In [de Bruin *et al.*, 2018], the authors found the forgetting problem in the pendulum with the limited buffer size. If the agent converges to swing to a preferred direction, then it may gradually forget the opposite (symmetrical) direction. Once the initial direction changes, the agent cannot swing up to the upright equilibrium position.

Configuration

We use DDPG in OpenAI Baselines[1] to solve this problem. All settings are default except for the discount factor $\gamma = 0.95$. We utilize the Ornstein-Uhlenbeck noise process [Uhlenbeck and Ornstein, 1930] to drive exploration and set the threshold $\eta = 0.01$ to determine the action whether belongs to exploration or exploitation. The exploration rate ϵ in Eq. (2) decays linearly from 1 to 0.1 during the all 3000 episodes. The evaluation of generalization performs per 10 episodes. The size of replay is 10000 in both FIFO and RS methods, while RS-DRB has two buffers with 5000 capacity. The hyper-parameter $\mathcal{T}_{\max}$ in Pendulum is set to 0.4.

Metrics

We also evaluate two performances. In the training period, we record the episode return from a fixed initial state where the pendulum is vertically downward and its acceleration is zero. In the generalization performance, we set different initial angles θ from $-\pi$ to π with the interval $\pi/36$ and keep the same initial acceleration. Then we calculate the average episode return of these 72 initial states as the generalization criterion. If the agent only remembers one direction, then it may not be able to swing up to the upward equilibrium position at the opposite initial state, and this causes a decrease in the generalization performance.

Performance

Due to the difficulty of continuous problems, more precise control is required. Once the forgetting problem occurs, its impact on performance can be more clearly observed. The FIFO retention method cannot hold its best performance as time progresses. It can be explained that the early exploratory experiences begin to rush out of the buffer, while the new generated experiences are concentrated in the current preference direction. The sample diversity of the buffer becomes insufficient, which is more likely to cause the forgetting problem. From Figure 4, we can find that both of the generalization and training performances of the FIFO method decrease after 1000 episodes. In the RS method, the probability that a new experience generated by the learned policy is the same as the probability of an early exploration experience. Therefore the agent cannot concentrate on the learned policy. This causes that the RS method achieves relative good performances around 1500 episodes, much slower than FIFO and RS-DRB. Our RS-DRB framework, which balances the exploration and exploitation experiences in training, achieves the best performance in terms of episode return and generalization. It can be seen that RS-DRB learned a good policy around 1000 episodes which is the same as the FIFO method. After that, RS-DRB maintains the good generalization performance in all episodes without occurring the forgetting problem. Meanwhile, the training performance still improves.

5 Conclusion

This paper presented a new RS-DRB framework to retain the experiences in the replay buffer. The exploration buffer with the reservoir sampling helps to maintain the coverage of the entire state space and the exploitation buffer, while the FIFO method focuses on the current policy. The adaptive sampling ratio balances the experiences sampled from these two buffers according to the change of the policy. In experiments, we quantify the severity of the forgetting problem through the generalization performance. Empirically, we find that our framework can improve the training and generalization of the policy and alleviate the forgetting problem on discrete and continuous problems. In the future, we would like to combine our RS-DRB framework with some popular sampling methods [de Bruin *et al.*, 2018], and extend it to improve the experience selection mechanism in deep multi-agent RL domains [Lowe *et al.*, 2017; Zheng *et al.*, 2018; Liu *et al.*, 2019]. It would also be interesting to combine the importance sampling mehods in PER with our sampling ratio in more complex games [Foerster *et al.*, 2017].

[1] https://github.com/openai/baselines

References

[Andrychowicz *et al.*, 2017] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5055–5065, 2017.

[Babuška and Groen, 2010] Robert Babuška and Frans C.A. Groen. *Interactive collaborative information systems*. Springer, 2010.

[Bellman, 1958] Richard Bellman. Dynamic programming and stochastic control processes. *Information and Control*, 1(3):228–239, 1958.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *CoRR*, abs/1606.01540, 2016.

[De Bruin *et al.*, 2015] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. The importance of experience replay database composition in deep reinforcement learning. In *Deep reinforcement learning workshop, Advances in Neural Information Processing Systems (NIPS)*, 2015.

[de Bruin *et al.*, 2016a] Tim de Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. Improved deep reinforcement learning for robotics through distribution-based experience retention. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3947–3952. IEEE, 2016.

[de Bruin *et al.*, 2016b] Tim de Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. Off-policy experience retention for deep actor-critic learning. In *Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NIPS)*, 2016.

[de Bruin *et al.*, 2018] Tim de Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. Experience selection in deep reinforcement learning for control. *Journal of Machine Learning Research*, 19(1):347–402, 2018.

[Foerster *et al.*, 2017] Jakob N. Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1146–1155, 2017.

[François-Lavet *et al.*, 2018] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, 2018.

[Goodfellow *et al.*, 2013] Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *CoRR*, abs/1312.6211, 2013.

[Lillicrap *et al.*, 2016] 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. In *International Conference on Learning Representations (ICLR)*, 2016.

[Lin, 1992] Long Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

[Liu *et al.*, 2019] Hao Liu, Alexander Trott, Richard Socher, and Caiming Xiong. Competitive experience replay. In *International Conference on Learning Representations (ICLR)*, 2019.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6382–6393, 2017.

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

[Schaul *et al.*, 2016] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[Uhlenbeck and Ornstein, 1930] George E. Uhlenbeck and Leonard S. Ornstein. On the theory of the brownian motion. *Physical Review*, 36(5):823, 1930.

[Vitter, 1985] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[Zheng *et al.*, 2018] Yan Zheng, Zhaopeng Meng, Jianye Hao, and Zongzhang Zhang. Weighted double deep multi-agent reinforcement learning in stochastic cooperative environments. In *Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, pages 421–429, 2018.