Applied Intelligence
https://doi.org/10.1007/s10489-020-01786-1

SLER: Self-generated long-term experience replay for continual reinforcement learning

Chunmao Li[1] · Yang Li[1] · Yinliang Zhao[1] · Peng Peng[2] · Xupeng Geng[1]

Abstract

Deep reinforcement learning has achieved significant success in various domains. However, it still faces a huge challenge when learning multiple tasks in sequence. This is because the interaction in a complex setting involves continual learning that results in the change in data distributions over time. A continual learning system should ensure that the agent acquires new knowledge without forgetting the previous one. However, catastrophic forgetting may occur as the new experience can overwrite previous experience due to limited memory size. The dual experience replay algorithm which retains previous experience is widely applied to reduce forgetting, but it cannot be applied in scalable tasks when the memory size is constrained. To alleviate the constrained by the memory size, we propose a new continual reinforcement learning algorithm called Self-generated Long-term Experience Replay (SLER). Our method is different from the standard dual experience replay algorithm, which uses short-term experience replay to retain current task experience, and the long-term experience replay retains all past tasks' experience to achieve continual learning. In this paper, we first trained an environment sample model called Experience Replay Mode (ERM) to generate the simulated state sequence of the previous tasks for knowledge retention. Then combined the ERM with the experience of the new task to generate the simulation experience all previous tasks to alleviate forgetting. Our method can effectively decrease the requirement of memory size in multiple tasks, reinforcement learning. We show that our method in StarCraft II and the GridWorld environments performs better than the state-of-the-art deep learning method and achieve a comparable result to the dual experience replay method, which retains the experience of all the tasks.

Keywords Continual reinforcement learning · Catastrophic forgetting · Dual experience replay · Experience replay model

1 Introduction

Reinforcement learning is a branch of machine learning in which an agent learns how to act by interacting with a dynamic environment [1]. With the massive influx of computing resources, deep reinforcement learning has achieved significant success in the field of go [2, 3] or video game [4, 5]. In such settings, the task is always isolated and single. However, in a real-world scenario, to achieve artificial general intelligence, it needs to create a continual learning [6] intelligent agent to learn and remember a series of consecutive tasks. An ideal of a continual learning system should meet the following attributes. First, a continual learning method should retain previously learned capacities. It means that is should be able to present an excellent performance on already encountered tasks. Second, the knowledge extracted from previous tasks should not inhabit the knowledge acquired from the new tasks. Third, when meeting a similar or relevant task, it should have an ability to gain a rapidly improved performance, which is known as positive transfer or constructive interference [7].

It is facing the challenge of continual learning, a body of literature focus on the study of this field. In recent years, with the development of deep learning, amount of studies apply the neural network to learn multiple tasks, where include separate training systems for each task [8, 9] or discovering action hierarchies [10]. However, these methods only perform well in simple tasks. It faces a considerable

✉ Yinliang Zhao
zhaoy@mail.xjtu.edu.cn

[1] Department of Computer Science and Technology, Xianjiaotong University, Xian, 710049, China

[2] Inspir.ai, Beijing, China

Published online: 07 August 2020

challenge when scaling multiple tasks. Training on all tasks simultaneously or having a replay buffer with limitless capacity can provide a chance to train scale multiple tasks.

Nevertheless, in a real-world application such as in industry or robotics situations, it always encounters cases where collecting experience is expensive and complicated. The simultaneous training methods may not be feasible in such a scenario, and the agent has no choice but to learn from only one task at a time. At the same time, the boundaries between tasks are always unknown, and it is hard to maintain a separate buffer for each task, which does also inhabits the possibility of simultaneously training multiple tasks. Besides, when computing and memory resources are constrained, with the number of tasks grows large, storing all the experiences of all tasks is becomes infeasible, and it will eventually terminate the continual learning. Especially, in reinforcement learning settings, where often use a first-in-first-out (FIFO) replay buffer to store the experience, and the environment is also non-stationary. When training multiple tasks in sequence, the new knowledge always erases the previous knowledge, which leads the optimal actions, and a large number of states are no longer visited. As a result, the agent loses the ability to perform well in previous tasks is known as catastrophic forgetting [11].

At present, there are many methods to address the challenge of catastrophic forgetting [12]. For example, the regularized optimization methods [13–15] that alleviate catastrophic forgetting by imposing constraints on the update of the neural weights; the elastic weight consolidation (EWC) is a typical one. However, the accumulation information of Fisher matrix regularizes can over-constrain the parameters of the network, which lessened learning the new tasks. Progressive Network method directly freezes subnetworks trained on individual tasks, and Progress & Compress [15] method uses EWC to consolidate the network after each task has been learned. While the Progress & Compress method dynamically expanding the size of the networks, it also cannot be adapted to the large scale tasks.

Based on Complimentary Learning System (CLS) theory [13], the dual-memory methods make full use of the replay mechanism for active learning during the training process, becoming one of the popular methods in the field of continual learning to address catastrophic forgetting [16]. However, this research did not make an in-depth exploration of the sampling strategy for two experience replays during training, and it becomes inefficient when scaling to multiple tasks. A rank-based method maintains a uniform distribution of experience of all tasks to reduce the forgetting problem [16], which ensures the process of learning not to depend on task boundaries. However, for a fixed memory size, it faces the challenge of choosing what data to store over time. The Generative Replay methods [17–19], which combine the idea of the data generator and the real data of new tasks to overcome catastrophic forgetting, have widely adopted in the deep learning community. Here, the data generator creates samples that mimic the training data of previous tasks. However, in reinforcement learning settings, the states are always correlated in time, and the environment is also non-stationary; it is hard to train a generator's data for reinforcement learning. At the same time, the current replay methods for reinforcement learning are not scalable and limiting due to the requirement of large training samples for continual learning.

In this paper, we propose a dual memory architecture, which combines the Self-generated Long-term Experience Replay (SLER) and short replay memory architecture for learning tasks scalability while averting catastrophic forgetting. The short replay memory stores the experience of current tasks without interfering with previous tasks in the SLER. The SLER retains the knowledge of all previous tasks. During the tasks switch, the short memory transfers the samples, which only contains the initial state, all actions, and all rewards for each episode to the SLER. The SLER combines the Experience Replay Model (ERM) and the samples from the short memory to learn tasks similar to previously seen tasks, where the ERM is a simple model to generate the simulated state sequence of the past tasks. Our method can effectively support scalability for continual learning and not need to know a prior task label to provide the proper samples. We compare with the state-of-the-art techniques for addressing catastrophic forgetting and achieve better or comparable results. Specific contributions are listed below:

- Simplicity. We propose the concept of Experience Replay Model (ERM), learning the sample model of environments through experience replay for the generation of simulation experience;
- Reduce the requirement of memory size. The proposed Self-generated Long-term Experience Replay (SLER) only retains the initial state, actions and rewards. ERM is used to generate the experience of the next state sequence for knowledge retention;
- Not need to know the boundaries of task. By replacing the long-term experience replay in dual experience replay with SLER, we propose a new continual reinforcement learning algorithm which can effectively reducing the need for real experience of past tasks.

In Section 2, we briefly review the basic theory of reinforcement learning and the related definitions of continual reinforcement learning; then we discuss the related work in Section 3. In Section 4,after introducing the Experience Replay Model and the idea of Self-generated Long-term

Experience Replay, we propose a new continual reinforcement learning algorithm; Section 5 introduces our experimental validations, and Section 6 is the summary and future work of the paper.

2 Background

The standard reinforcement learning paradigm is an agent interacting with an environment to maximize long term expected rewards, which abstract the agent as a decision-maker, and models the environment as a Markov decision process (MDP). An MDP is a five-tuple: $\langle S, A, R, T, \gamma \rangle$, where S represents the state space, A represents the action space, R represents the reward function, $T : S \times A \times S \rightarrow [0, 1]$ represents the transition function, and $\gamma \in (0, 1]$ represents the discount factor, respectively. The discount factor plays a significant role in determines the importance of future rewards. At each step, an agent in the current state $s_t \in S$ executes an action $a_t \in A$ according to probability $p(s_{t+1}|s_t, a_t)$ transitions to a successor state $s_{t+1} \in S$, and receives a reward signal $r_{t+1} \in R$. The goal of reinforcement learning is to find a policy $\pi(a_t, s_t)$ to maximize its cumulative reward $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$. The optimal action-value function of Q-learning $Q^*(s_t, a_t)$ represents the maximum expected return that is achievable when selecting an action a_t under the optimal policy π^*. The Bellman equation provides an iterative update for the Q function, $Q^*(s_t, a_t) = E[r_t + \gamma max_{a'} Q^*(s_{t+1}, a')]$.

The Deep Q-Network (DQN) [4] is a method that applies the convolutional neural networks to approximate the action-value function and achieves a notable success in Atari 2600. A DQN takes an observation s as the input of the convolutional neural networks and calculates the action-value function $Q(s, a; \theta)$ for all possible action a given the state s, where θ represents the parameter of neural networks. At each iteration i, by minimizing the expected loss $L_i(\theta_i)$ to train DQN, with target $y_i^{DQN} = r + \gamma max_{a'} Q(s', a'; \theta_i^-)$.

$$L_i(\theta_i) = E_{(s,a,r,s') \sim D} \left[\left(y_i^{DQN} - Q(s, a; \theta_i) \right)^2 \right] \tag{1}$$

θ_i^- represents the parameters of a target network, which keeps constant for many iterations while updating the behavior network $Q(s, a; \theta_i)$. D is the experience replay database, which records the agent's experience (s, a, r, s') in a first-in-first-out (FIFO) queue to train the DQN parameter to minimize the loss function. Notice that, DQN is an offline method, meaning that the experiences in replay buffer jumble together new and previous data. By randomly sample from D, the experience replay buffer plays an essential role in breaking the temporal correlations present in DQN and prevents catastrophic forgetting of older experiences.

We are interested in continual learning setting, where an agent solves a series of reinforcement learning tasks, rather than a single task. The objective of multi-task as the following:

$$\frac{1}{m} \sum_{j=1}^{m} \frac{1}{n_j} \sum_{i=1}^{n_j} L_i(\theta_i) \tag{2}$$

Where n_j represents the number of experiences for task j, m represents the total number of tasks, and θ is common across for all tasks. And a typical assumption in multiple tasks of learning is that each task draws from a different distribution.

When a system continuously learns multiple tasks, benefiting from the experience replay mechanism of DQN, we can maintain an unlimited capacity experience replay buffer to preserve knowledge for every task to prevent catastrophic forgetting of older experiences. However, in real-world applications, the capacity of the buffer is always limited; the standard FIFO implementation of an experience replay buffer will lose earlier experiences.

To address this problem, we propose a Self-generated Long-term Experience Replay (SLER) for recording experience. Through the use of the Experience Replay Model (ERM) to generate the simulation experience of previous tasks to retain knowledge effectively efficiently, we can eliminate the requirement of FIFO size and address the catastrophic forgetting challenge.

3 Related work

Continual learning, also known as lifelong learning [7], has achieved a great deal of attention in the deep learning community. The goal of continual learning is to learn a system to complete multiple tasks continually. Since its inception of the idea by Thrun and Mitchell [7], it has been widely applied in real-world applications where the data arrives in streams, such as autonomous driving or learning of robotic agents [6, 20]. However, due to the limitation of the computing resources, with a large number of tasks, these methods cannot adequately deal with the scalability issues. Nowadays, researchers focus on preserving prior knowledge within the networks for continual learning. The core challenge of methods is the catastrophic forgetting.

The regularization methods alleviate catastrophic forgetting by adding a penalty term on the loss function, which can constrain the update of the neural network. For example,

Kirkpatrick et al. propose the Elastic Weight Consolidation (EWC) [13], which applying the diagonal of the Fisher matrix to estimate the importance of weights to prevent forgetting. However, the accumulation matrix information of Fisher regularizes can over-constrain the parameters of the network, which lessened learning the new tasks. The Compress and Progress [15] method combines the EWC and the Policy Distillation [21] to iteratively transfer learned policies into a single network to prevent forgetting. Due to existing too high regularization among these methods, the model may saturate after several tasks. It is hard to deal with the large scale issues.

Based on Complimentary Learning System (CLS) theory, the dual-memory methods make full use of the replay mechanism for active learning during the training process, becoming one of the popular methods in the field of continual learning to address catastrophic forgetting [16]. This method can be linked to the interaction between the hippocampus and neocortex to avoid catastrophic forgetting in mammals [11]. However, this research did not make an in-depth exploration of the sampling strategy for two experience replays during training, and it becomes inefficient when scaling to multiple tasks. A rank-based method maintains a uniform distribution of experience of all tasks to reduce the forgetting problem [16], which ensures the process of learning not to depend on task boundaries. However, for a fixed memory size, it faces the challenge of choosing what data to store over time.

The Rehearsal approaches are the simplest way to solve the catastrophic forgetting problems, which save raw samples as a memory of past tasks. However, it needs a replay buffer to retain previously learned knowledge of all tasks, and it is also an ideal way to solve large scale issues. Instead of modeling the past from few samples as done in Rehearsal methods, the Generative Replay method [17–19], which combine the idea of the data generator and the real data of new tasks to overcome catastrophic forgetting, have widely adopted in the deep learning community. Here, the data generator creates samples that mimic the training data of previous tasks. However, most of the Generative Replay methods aim to address the classification tasks. In reinforcement learning settings, the states are always correlated in time, and the environment is also non-stationary; it is hard to train a generator's data for reinforcement learning. At the same time, the current replay methods for reinforcement learning are not scalable and limiting due to the requirement of large training samples for continual learning. Inspired by the work showing by the poor of dual-memory replay method and the Pseudo-Rehearsal methods, we propose a Self-generated Long-term Experience Replay to reduce the constraints on memory size to solve catastrophic forgetting.

4 Method

In current reinforcement learning settings, an agent interacts with the environment to deal with consecutive multiple tasks. To alleviate the occurrence of catastrophic forgetting, which the new experience is automatically supplant the previous experience, it needs to record a larger number of experiences in a First-In-First-Out (FIFO) buffer. However, the size of the replay buffer is always limited, and the boundary tasks are unknown in real-world applications. However, the traditional replay-based method needs to know the prior task labels to help the continual learning system to recall the proper samples. Therefore, the current replay-based methods for reinforcement learning is not scalable. To alleviate this challenge, inspired by the generative replay model, we propose a new continual reinforcement learning algorithm that applies the Self-generated Long-term Experience Replay to generate the experience of previous tasks to alleviate catastrophic forgetting. The process of our training consisted of two steps: short-term system DQN training and the Self-generated Long-term Experience Replay training. When a new task is coming, we use the standard DQN to train solely and store the experiences of the current task in short-term replay buffer. At the same time, we transfer the knowledge from the short-term replay buffer to the Self-generated Long-term Experience Replay to retain knowledge for previous tasks. And our continual reinforcement learning framework, as shown in Fig. 1. Next, we will elaborate to introduce how the structure of Self-generated Long-term Experience Replay works.

4.1 Experience Replay Model (ERM)

As shown in Fig. 2, the Experience Replay Model (ERM) is composed of two parts: the Model Learner and the Replay Generator. The Model Learner aims to learn an environment sample model to generate the simulation experience, and use the Mean Square Error (MSE) as the loss function to update the parameters. The structure of the Replay Generator is the same as the Model Learner. When the task switching, the Replay Generator will copy the parameters of the Model Learner, so that it can use to retain the knowledge of the previous tasks. The Replay Generator plays a significant role in generating the simulation experience to reduce the requirements of large training corps for continual learning. When a new task is coming, we first sample the experiences (S_{new}, A_{new}, S'_{new}) from the short-term experience replay to the Model Learner to construct the environment sample model. And the Reply Generator only records the previous tasks' initial experience (s_{old}, a_{old}) into the ERM. Because the Replay Generator share the parameters with the Model

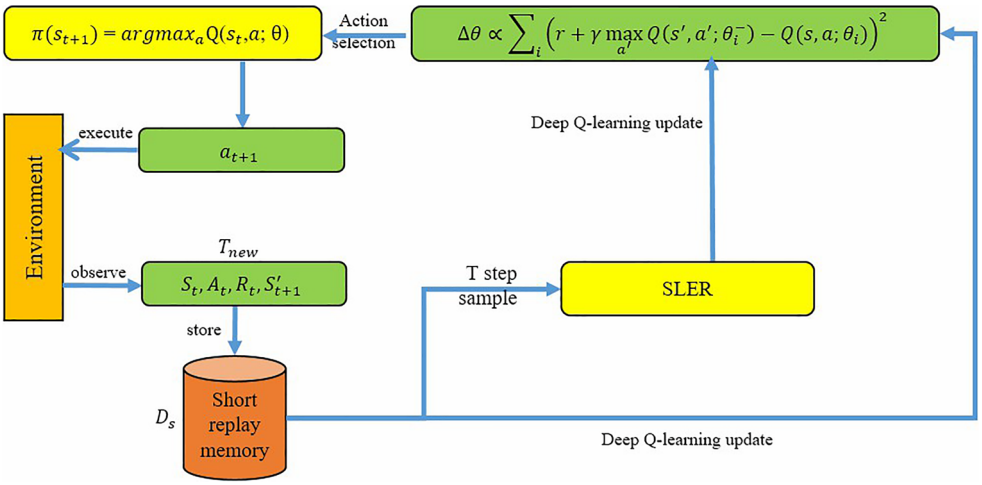


Fig. 1 Continual Reinforcement Learning using Self-generated Long-term Experience Replay (SLER) to learn tasks

Learner, it can generate the next states (s'_{old}) of the previous tasks. In this way, the ERM can continuously learn the new task and not forget the old tasks, which can help the agent to overcome the catastrophic forgetting challenge.

4.2 Self-generated long-term experience replay (SLER)

Based on the generation ability of the Experience Replay Model (ERM), we propose the structure of Self-generated Long-term Experience Replay (SLER). As shown in Fig. 3, we only save the initial state S_0, all rewards R, and actions A of each episode in SLER, and applying ERM to generate the intermediate sequence states. In Fig. 3, each pink rectangular box represents the experiences of an episode, after each iteration, we use the ERM to generate the new minibatch sample of the state, and delete the previous state(which can reduce the requirement of memory size). After a while, the SLER only saves the initial states, and the generation experience states which in a particular moment (S_x, S_y...).

Due to the inaccuracy of the ERM itself, the SLER needs to retain a small number of real experiences to improve the effectiveness of generating experience. As shown in Fig. 4, the left parts represent the correct experience replay. We use a small number of real experiences to replace the simulation experience to ensure that the sample model of the environment corresponds with the actual model. If the accuracy of the model high, it does not need to save too many real experiences. If the accuracy of the model is low, it needs to save more real experiences to improve the effectiveness of SLER .

4.3 Proposed algorithm

We adapt the DQN [4] algorithm as the baseline learning algorithm. We use short-term experience replay D_s to retain the experiences of a new task (the task being learned), and Self-generated Long-term Experience Replay (SLER) to generate all past tasks' experiences. As shown in Fig. 5, at each iteration, we use the samples from both D_s and SLER to train the Agent and the Model Learner of ERM. When

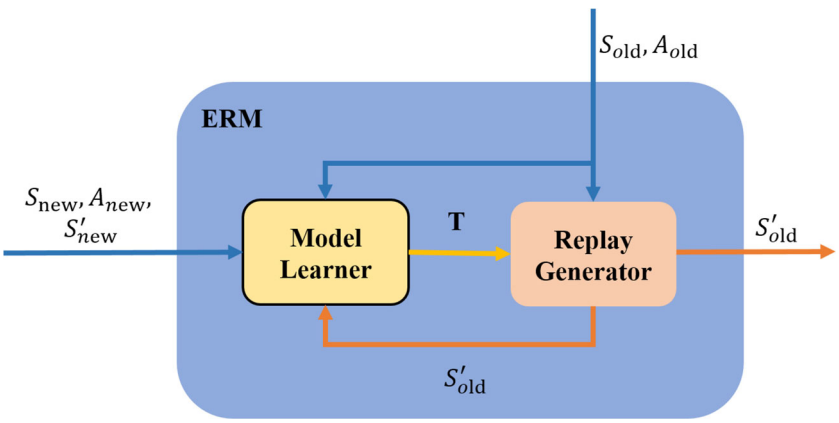


Fig. 2 The architecture of Experience Replay Model (ERM). The Model Learner continuously learned the environmental sample model from both real experiences (S_{new}, A_{new}, S'_{new}), (S_{old}, A_{old}) and generated the next states S'_{old}. When the Model Learner updates the task switches (each training T times), the parameters of Replay Generator, so the Replay Generator can be used to generate the next states S'_{old} of old tasks' environments

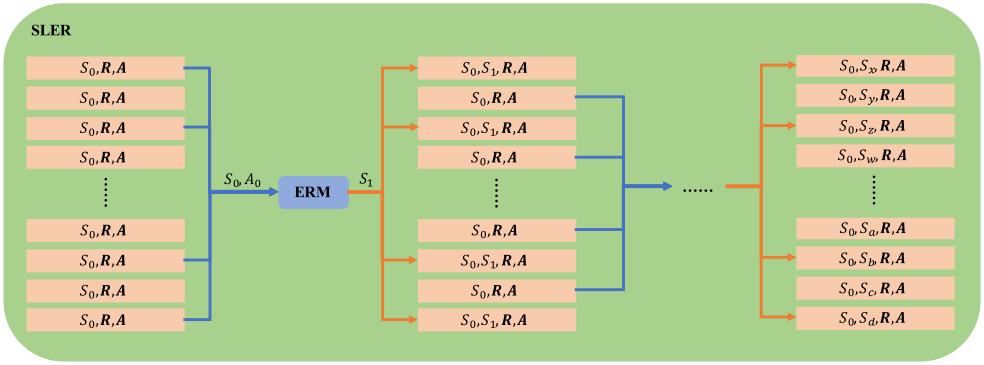


Fig. 3 The architecture of Self-generated Long-term Experience Replay (SLER). Saving only the initial state S_0, all rewards R, and actions A of each episode. The intermediate sequence states can be generated by the Experience Replay Model (ERM). It generates mini-batch samples after each iteration, and each time when generating the new state, delete the previous one

the learning of all tasks is completed continually, the agent can handle all tasks simultaneously without forgetting any of them.

The core parameters of the algorithm are shown in Table 1. The Minibatch size represents the total number of samples extracted from the short-term experience replay and SLER. The Initial exploration value and the Final exploration value are empirically optimal settings (in practice, the final exploration value above 0.05 may result in training failure). The Learning start step is scaled-down according to the number of training steps of the environment. Besides, we also verify the pixel-state learning ability of the algorithm. The network consists of a two-layer convolutional neural network (CNN) plus a fully connected network. The first layer of CNN accepts the image input of the environment, and the output layer size is still the number of actions.

The value network of the proposed algorithm is a fully connected network with two hidden layers composed of [512, 512] nodes. The input layer is the state size, the output layer is the action size, and the activation function of the hidden layers network is ReLU [22]. The input of the Experience Replay Model network is the state of the environment and the action taken by the agent, and the output of the Model Learner network is the next state of the environment. The number of nodes at each hidden layer is [16, 64, 256, 64, 16]. The activation function of the output layer is sigmoid, and the other layers use the ReLU activation function. The model is updated using the Adam optimizer, and apply the mean-squared error to calculate the loss function. The short-term experience replay uses the Prioritized Experience Replay [23] strategy to accelerate learning. The proposed algorithm also uses the Dropout technique [24] to increase the ability to resist the over-fitting of the value network.

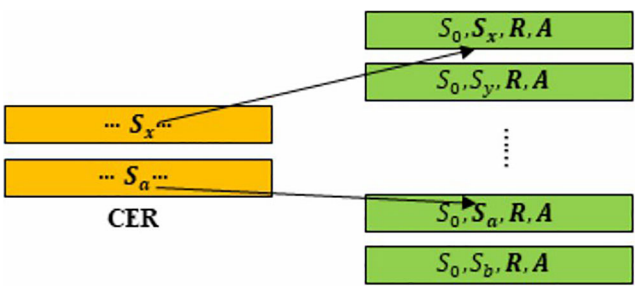


Fig. 4 The architecture of Correct Experience Replay (CER)

The SELR algorithm is shown in Algorithm 1, the algorithm input is a randomly initialized neural network weight, which is used to fit the action-value function in the Q-learning algorithm, and uses this as a reference to select the action based on the state of the environment. The output is the optimal parameter after continuous task learning, which can solve all tasks at the same time. The algorithm first initializes the correct experience replay memory D_r, and the short-term experience replays D_s, the parameters of the action-state function Q, the parameters of the target network, and the optimal network (line 1). At the beginning of each task, the algorithm transfers the state and reward pairs in the short-term experience replay into the correct experience replay memory, resets the short-term experience replay, and updates the optimal network parameters (line 4). The initialization state (line 6) is obtained from the environment interaction at the beginning of each episode, and the ϵ-greedy algorithm and Q-Network receive the optimal action (lines 8 to 9) at each step in the episode. The move is performed and get the state and reward of the environment, and store it in the short-term experience replay (lines 10 to 11). Update M with real and generated transitions by step learning sample random minibatch of transitions from D_s, generate transitions by M (range 13 to 14). The parameters of the network and the parameters of the target network are updated every C steps (line 16).

Algorithm 1 SELR algorithm.

Input: Action-value function Q with random weights θ
Output: Action-value function Q with random weights θ^*

Initialize short-term replay memory D_s and correct experience replay memory D_r, generator buffer M with random weights θ_m, action-value function Q with random weights θ, target action-value function Q^- with weights $\theta^- = \theta$, optimal action-value function Q^* with weights $\theta^* = \theta$;
begin
  for $task = 1\ to\ J$ **do**
    Store D_s in D_r and generator buffer in M and $\theta^* = \theta$;
    for $episode = 1\ to\ E$ **do**
      Initialize first state s_1;
      for $t = 1\ to\ T$ **do**
        With probability ϵ select a random action a_t;
        Otherwise select $a_t = argmax_a Q(s_t, a, \theta)$;
        Execute action a_t in emulator and observe reward r_t and s_{t+1};
        Store transition (s_t, a_t, r_t, s_{t+1}) in D_s ;
        Set $s_t = s_{t+1}$;
        Update M with real and generated transitions by step learning;
        Sample random minibatch of transitions from D_s, generate transitions by M;
        Preform a gradient descent step with respect to the θ and θ^* ;
        Every C steps reset $Q^- = Q$
      end
    end
  end
end

5 Experiments and results

5.1 Environments

StarCraft II is a well-known real-time strategy game in which players need to complete the construction base, collect resources, and produce team troops. The ultimate goal is to defeat the enemy and make as little damage as possible. The game is extremely complex and strategic which has become a new challenge in the field of reinforcement learning. Using the API of the StarCraft II jointly released by Blizzard Games and Deepmind [25], we constructed a continual learning task set for verification and comparison of the continual reinforcement learning algorithms. The screenshot of the task set is shown in Fig. 6, the left side is the unit (Marine) controlled by our algorithm, and the right side is the enemy (Zergling) with different parameters.

This task sets and the specific parameters are shown in Table 2. Our unit needs to learn the strategy of *Never Approaching* (NA), *Keep Fleeing* (KF), and *Hit and Run* (H&R). In Task H&R, if our unit defeats the enemy, it will receive a reward 1, or the reward is assigned 0. In the other two tasks, our unit will receive a reward 1 only if it is alive at the end of the episode, or the reward is assigned 0. This series of environments have large randomness. To ensure the operations consist of players, only one action command is transmitted every 8 frames, and the corresponding enemy will have different actions in different rounds. Therefore, the learning result will be slightly different each time.

To test our SLER method can learn more tasks adaptively, we change the role of opposite enemies in eight tasks, separately. As shown in Table 3, we use different Zerg units to represent the role of enemies. The agent should be able to adopt different strategies to against different enemies after completing the sequence tasks training. For example, if the enemies with faster speed, larger life value and high attack power, the agent needs to adopt a strategy of Never Approaching; if the enemies with small life value, slow speed, and low attack power, the agent needs to take a policy of Hit and Run; if the health and speed of the enemies in an average level but have a high attack power, the agent needs to adopt a strategy of Keep Fleeing.

Besides, we also use the GridWorld navigation task sets (GridWorld I and II) a to test the validity and accuracy of our algorithm. As shown in Fig. 7, the grid layout is $11 * 11$, and the environment is set to 100 steps per episode. The agent needs to move from the starting point (gray box) to the target point (black box) to receive a reward 1 for each step. The dark grey part is the obstacles, which the agent cannot pass. The GridWorld I consists of three tasks, with the targets at the down-left (DL), up-right (UR), and down-right (DR). The environment of this GridWorld I is relatively stable it is more suitable for verifying the effectiveness of the algorithm and acting as the comparing experiments. The GridWorld II consists of four tasks; the agent needs to reach the up-left (UL), down-left (DL), down-right (DR), and up-right (UR) four different targets to get rewards. The other settings are the same as the GridWorld I environment. GridWorld II is used to evaluate whether our algorithm can deal with more tasks.

The action space of the StarCraft II includes the movement in the four directions of up, down, left and right, and the attack action against the target enemy. The state-space includes the two-dimensional coordinate information and

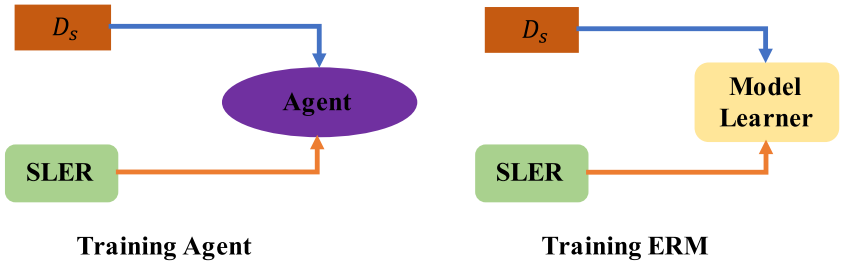


Fig. 5 At each iteration, our algorithm uses the samples from both short-term experience replay (D_s) and SLER to train the Agent and the Model Learner of ERM

the Health Point of our unit and the enemy. In the GridWorld environments, the action space includes the movement in the four directions of up, down, left, and right. The corresponding state space is the coordinate information of the agent and the target.

5.2 Results

5.2.1 The importance of correct experience replay

At first, we verify the continual learning ability of the algorithm when retaining 0%, 5%,10%, 15%, 25%, and 45% real experiences. The experimental results are shown in Fig. 8. The X-axis represents the evaluation of the training model at every 2000 iterations, and the Y-axis represents the accumulative rewards for all tasks. The left figure shows the experimental results of the GridWorld I. The environment of the GridWorld I is relatively stable; the slight difference between the generate experiences and the real experiences will lesson the accuracy of the experiment. We can see that when the retention number is less than 15%, the agent cannot master any task. Therefore, it is necessary to retain a large number of real experiences to get a high learning preference. As shown in Fig. 8, when retaining 25% real experiences, it presents a strong preference to resist forgetting. So we retain 25% for subsequent experiments to train GridWorld I tasks. The right figure is the experimental results on the StarCraft II task sets. Because the environment is relatively random than GridWorld I environment, it does not need to save too much real experience for training. Although saving more real experiences can resist catastrophic forgetting, but it also decreases the performance of the memory is limited. In subsequent experiments, we retain 15% of real experiences for the StarCraft II environment to evaluate algorithm performance.

5.2.2 Evaluation and comparison for the continual learning ability of SLER

Here we verify the results of continual learning ability on the StarCraft II tasks and the GridWorld I tasks separately. The baseline algorithms are the DQN and the Reservoir sampling method [16] with a dual-memory replay mechanism. Reservoir sampling has proven to be the most effective long-term empirical replay compression method which can obtain the whole distribution of the experience. Since the StarCraft II environment is relatively randomness, the results of the demonstration are the average of 10 experiments. The GridWorld I and GridWorld II environment are relatively stable; the results of the demonstration are the average of 5 experiments. The X-axis represents the evaluation of the training model for every 2000 iterations, and the Y-axis represents the reward for each task. The color of the background indicates the task currently being trained, and the shadow around the solid line is the variance of multiple pieces of training.

The experimental results in the GridWorld I are shown in Fig. 9. The label represents the target position, including down-left (DL), up-right (UR), and downright (DR). We can see that the algorithm uses SLER to generate sequence experience for knowledge retention effectively alleviates the

Table 1 The core parameters of the algorithm

Hyperparameter	Value	Description
Minibatch size	60	The number of samples used for each random gradient descent update, here the total number of D_s and SLER
Target network update frequency	500	Parameter update frequency of the Target network
Discount factor	0.99	Discount factor γ in DQN update
Learning rate	0.001	Algorithm learning rate
Initial exploration	1	Initial exploration value of ϵ-greedy exploration
Final exploration	0.05	The final exploration value of ϵ-greedy exploration
Learning start step	1000	Number of steps to start training

Fig. 6 The screenshot of the StarCraft II continual learning task set. The left side is the unit (Marine) controlled by our algorithm, and the right side is the enemy (Zergling) with different parameters

occurrence of catastrophic forgetting in continual learning. Compared with the dual experience replay algorithm (DER), which uses all tasks' practical experience for learning, our method only saves 25% real experience for training and also obtain a comparable result and effectively alleviate the catastrophic forgetting.

As shown in Fig. 9, the traditional DQN algorithm only can retain the experience of the current task, so the agent can only master the third task. The dual experience replay algorithm, which preserves 25% of the real experience (DER 25%), can only mater the second and the third task. It indicated that due to the limitation of memory size, the experiences of the first task is discarded during training the third task. The reservoir sampling [16] (DER 25% RS) can alleviate catastrophic forgetting, which obtaining comparable results. However, the reward of the third task is lower than us, and there is also a large fluctuation in the process of task training.

The experimental results of the StarCraft II environment (Fig. 10) is shown that in a highly random environment, the algorithm with SLER retaining, which saves less (15%) real experiences can still achieve good learning results. The DQN algorithm and the dual experience replay algorithm that employs 15% real experience (DER 15%) to train the agent, to some extent, forgetting the previous experiences. With the reduction of real experience, the process of training the second task is more prominent. The reservoir sampling (DER 15% RS) also fails to achieve the same performance as our algorithm, and there is also a large fluctuation in the process of training.

Table 2 The parameters of the StarCraft II continual learning task set

Task number	HP of enemy	Speed of enemy	Skill to master
1	100	Fast	NA
2	100	Normal	KF
3	70	Slow	H&R

Table 3 The parameters of the StarCraft II continual learning task set

Task number	Enemy unit	View radius	Health value	Speed
1	Baneling	8	30	Normal
2	Hydralisk	9	80	Normal
3	Mutalisk	11	120	Fast
4	Corruptor	10	200	Fast
5	Overlord	12	225	Slow
6	Queen	9	175	Slow
7	Roach	9	145	Normal
8	Zergling	8	35	Fast

In short, our algorithm can achieve the win rate of above 90% on GridWorld I and 85% on StarCraft II, when the agent solves three tasks after the sequence training. That means it masters three tasks at the same time and effectively alleviates the occurrence of catastrophic forgetting.

5.2.3 Limit-size buffer

Storing all previous experiences in replay buffer is an ideal way to alleviate catastrophic forgetting. As shown in Fig. 9 and Fig. 10, we test whether our method can deal with the limited-buffer problem by saving a small number of experiences in SLER. To ensure the fairness, the reservoir sample method can decide when to replace new experiences in the replay buffer [16]. We find that the SLER method, which we proposed to perform well and can effectively reduce catastrophic forgetting even with a replay buffer that only needs the small size of experiences. It means that using the SLER method can reduce the requirement of memory size and alleviate catastrophic forgetting.

More memory size is also needed if using pixel-state as input. We are comparing the dual experience replay algorithm (DER), which retains the real experiences of all

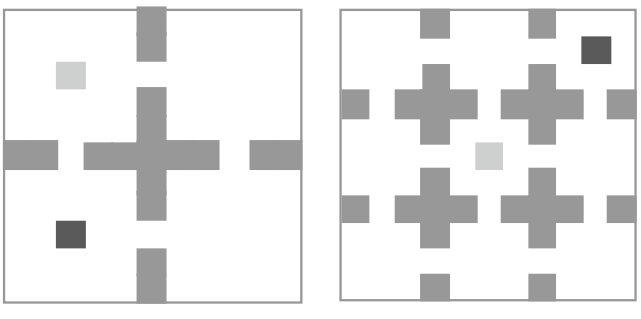


Fig. 7 An example task of both GridWorld I and GridWorld II. The grey box is the agent, and the black box is the target. The dark grey part is the obstacles. The agent needs to reach the target for rewards as more as possible in each episode

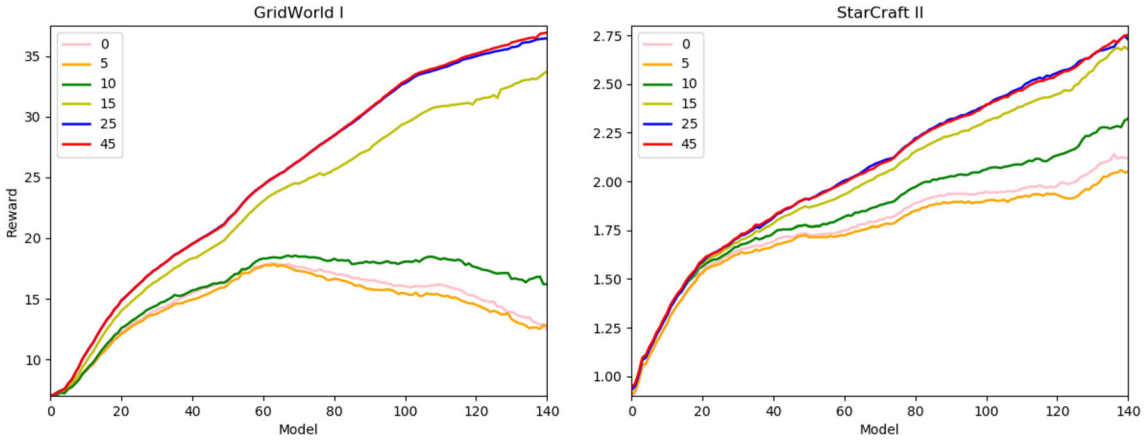


Fig. 8 The impact on the performance of the algorithm by retaining a different amount of real experience. The GridWorld I (Left) is more sensitive to error of generated experience than the StarCraft II (Right)

tasks, to verify our method cannot be constrained by the memory size. The experiment of results is shown in Fig.11. We can see that our algorithm still has good continual learning ability. Although it has a larger learning variance than the learning by all real experiences, it presents the same learning effect with the dual memory method; even it is only saving 25% of the actual experiences of previous tasks. Moreover, due to applying the image as input, the agent masters the two optimal paths at the same time on the down-right (DR) task (Fig.12), and only learn one of them when learning with the extracted features.

5.2.4 Learning more tasks

Depending on the knowledge of previous tasks to help the agent adaptively learn more tasks is an ideal of continual learning. In the Starcraft II environment, as shown in Table 4, We evaluate whether our method can learn more tasks. The data in the table represent the win rate with the current strategy, and the short line indicates that the agent will not adopt this strategy. The experimental results show that the agents will adaptively choose the optimal strategy and obtain a relatively high preference when facing different enemies.

We also verify the continual learning ability in a more complex environment. As shown in Fig. 7, the GridWorld II setting is more complicated than the GridWorld I. Whether the agent obtains the knowledge in GridWorld I task sets can effectively solve the tasks in GridWorld II is relatively attractive in continual learning. The results of the continual learning ability of our algorithm in the GridWorld II environment are shown in Fig. 13. Moreover, we also compare the results using the dual experience replay algorithm (DER) method, which contains all real experiences for training. We can see that the algorithm with SLER, which combing the simulation experiences and a small number of real experiences for training, can resist the occurrence of catastrophic forgetting. The third task can only be mastered when the fourth task starts training, and the fourth task cannot reach the optimal reward until the end of the training. Although the environment of GridWorld II is relatively complex than the GridWorld I, our method still obtain a comparable result.

5.2.5 Comparison to P&C and EWC

Finally, we compare our method with P&C and EWC, in which the methods assume that the boundaries between the tasks are known and be regarded as the state-of-the-art method for continual reinforcement learning for alleviating catastrophic forgetting. We use SLER with 25% real experiences in GridWorld I tasks and 15% real experiences in StarCraft II tasks. The results of the experiment are shown in Fig. 14. We can see that our method obtains comparable performance than Progress & Compress (PC) and better performance than Elastic Weight Consolidation (EWC), despite our method only retain slightly real experiences.

5.2.6 The time complexity analysis

We also analyze the time complexity between our method and the state-of-the-art methods, such as P&C and EWC, which aim to alleviate catastrophic forgetting for continual reinforcement learning. The results of the experiment are shown in Fig. 15. We can see that our method obtains significant performance than Elastic Weight Consolidation (EWC) and achieve a comparable preference than Progress & Compress (PC) despite our method only retain slightly real experiences. It is because of that, the EWC method applying the Fisher matrix to estimate the importance of weights to prevent forgetting. With the number of the tasks, the accumulation matrix information of Fisher regularizes can

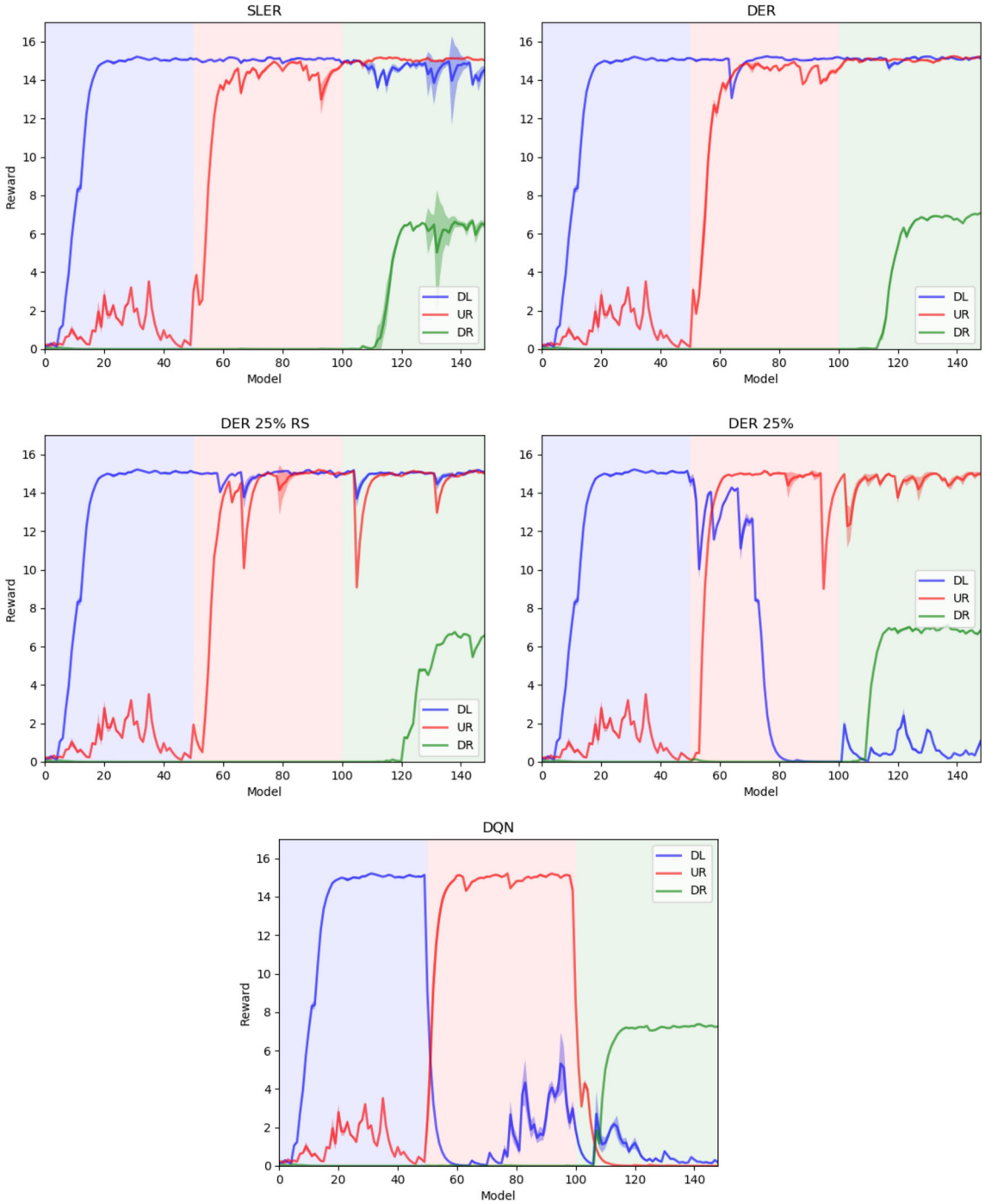


Fig. 9 Our algorithm (SLER) get comparable results with dual experience replay (DER) in GridWorld I. When reserving the same amount of real experiences, it also has a better ability on knowledge retention with reservoir sampling (DER 25% RS), FIFO dual experience replay (DER 25%) and DQN

increase the training time steps. The Compress and Progress method combines the EWC and the Policy Distillation to iteratively transfer learned policies into a single network to prevent forgetting, which decreases the updates of the networks and achieve comparable results with us.

5.2.7 The training results of our algorithm

The training results of the GridWorld I task sets are shown in Fig. 16. We can see that, during the first time, the agent needs to explore the environment; it always be trapped in

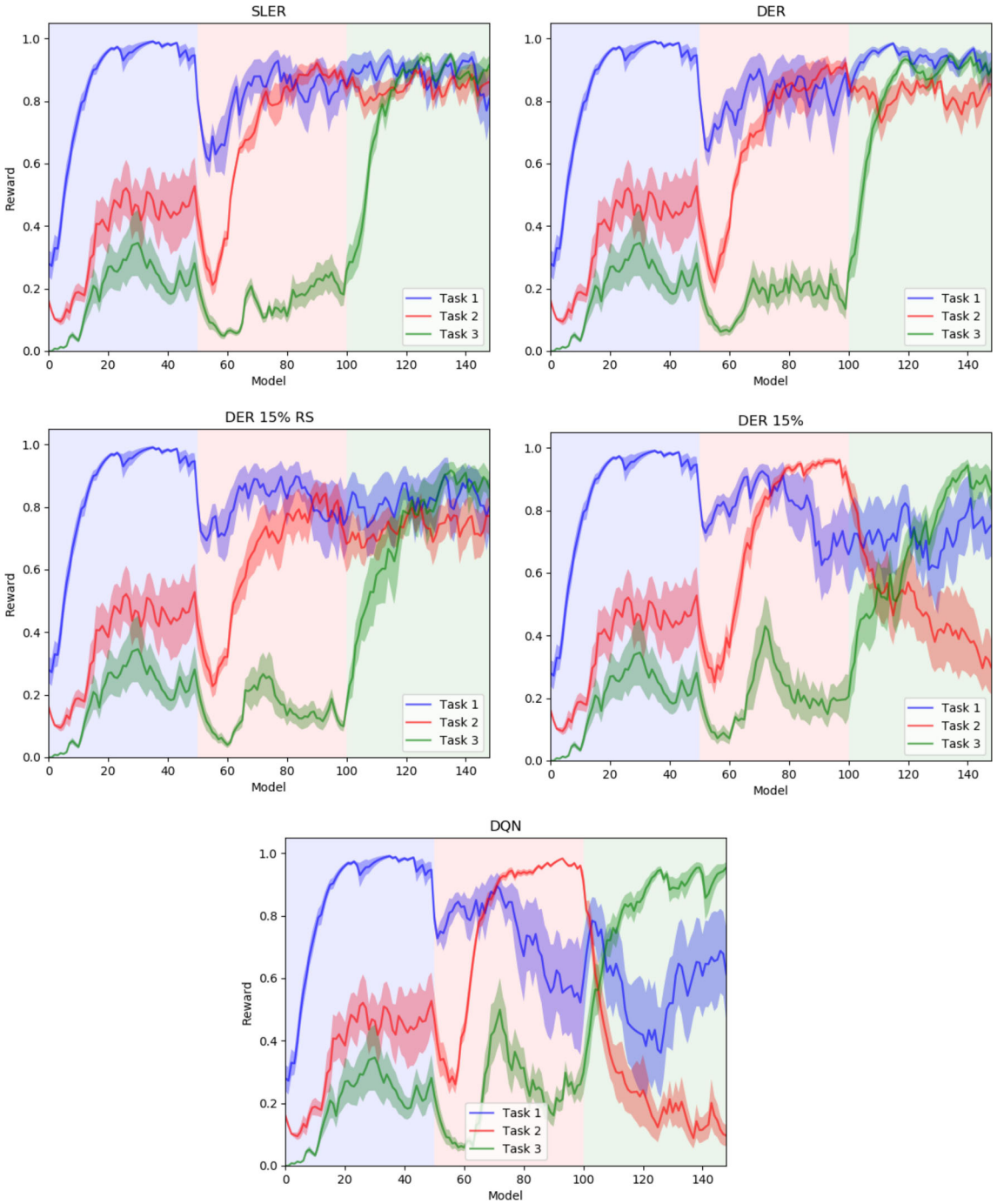


Fig. 10 Our algorithm (SLER) get comparable result with dual experience replay (DER) in StarCraft II. When reserving the same amount of real experience, it also has better ability on knowledge retention with reservoir sampling (DER 25% RS), FIFO dual experience replay (DER 25%) and DQN

the left-upper corner. After mastering both toward down-left and toward up-right tasks, the agent can complete the down-right tasks easily.

The training results of the StarCraft II, as shown in Fig.17. For each subfigure, the left picture represents the agent coordinate distribution after mastering the skill of the task, the middle picture represents the agent execute the attack command, and the right represents the enemy coordinate distribution. We can see that the agent can acquire the corresponding skills after a while.

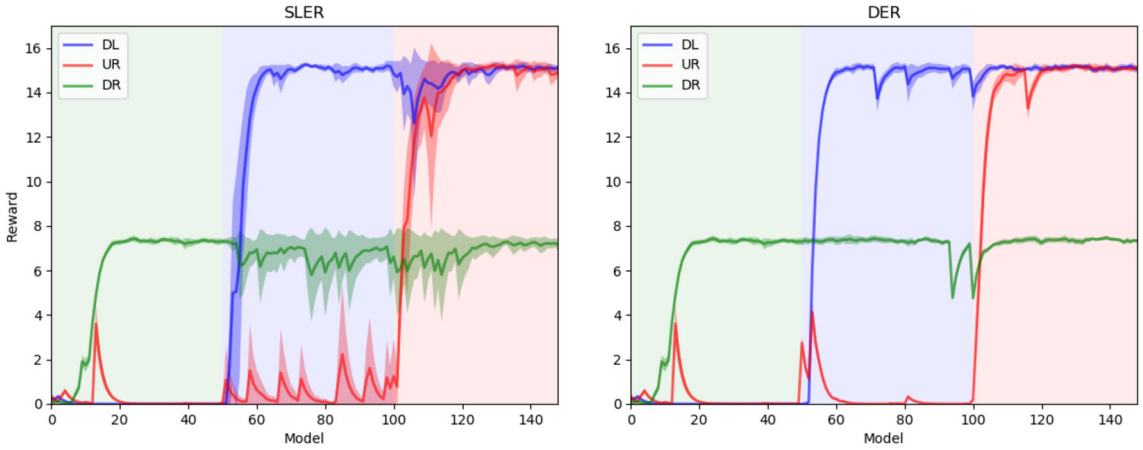


Fig. 11 Learning directly from pixel-state has a larger variance, but still comparable with dual experience replay algorithm (DER)

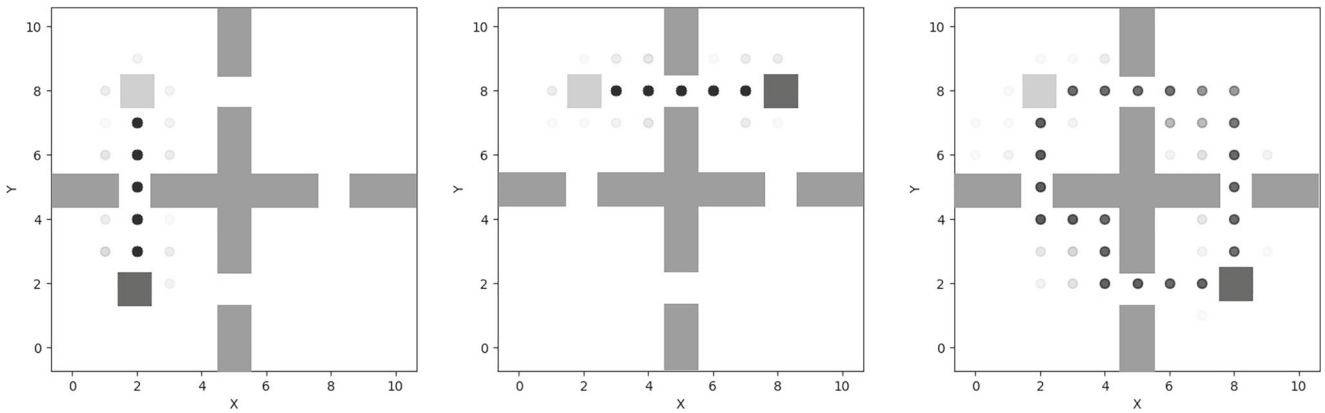


Fig. 12 The agent learning directly from pixel-state can master the two optimal paths at the same time on the down-right (DR) task, and only learns one of them when learning with the extracted features

Table 4 The parameters of the StarCraft II continual learning task set

Enemy unit	Hit and Run winning rate	Keep Fleeing winning rate	Never Approaching winning rate
Baneling	1	–	–
Hydralisk	–	–	1
Mutalisk	–	–	1
Corruptor	–	1	1
Overlord	–	–	1
Queen	–	–	1
Roach	–	0.32	1
Zergling	1	–	1

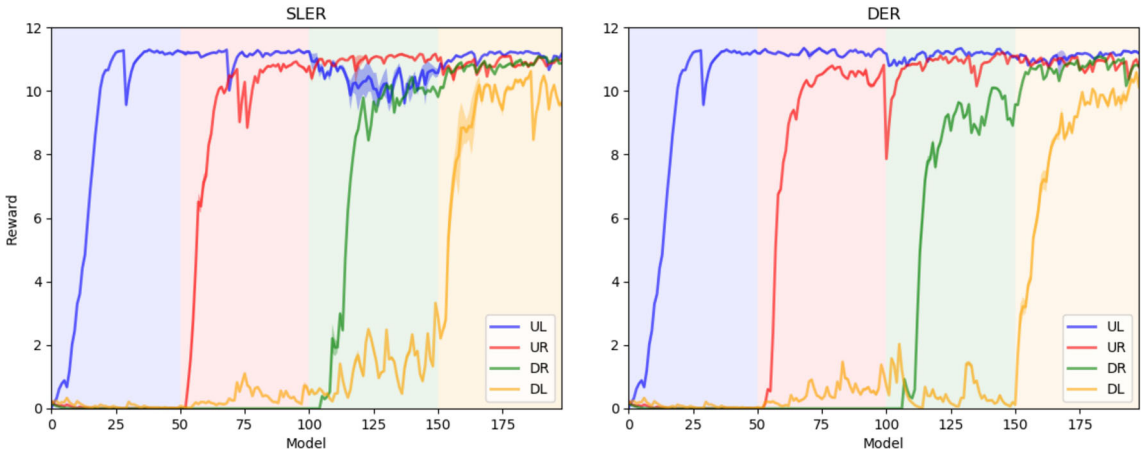
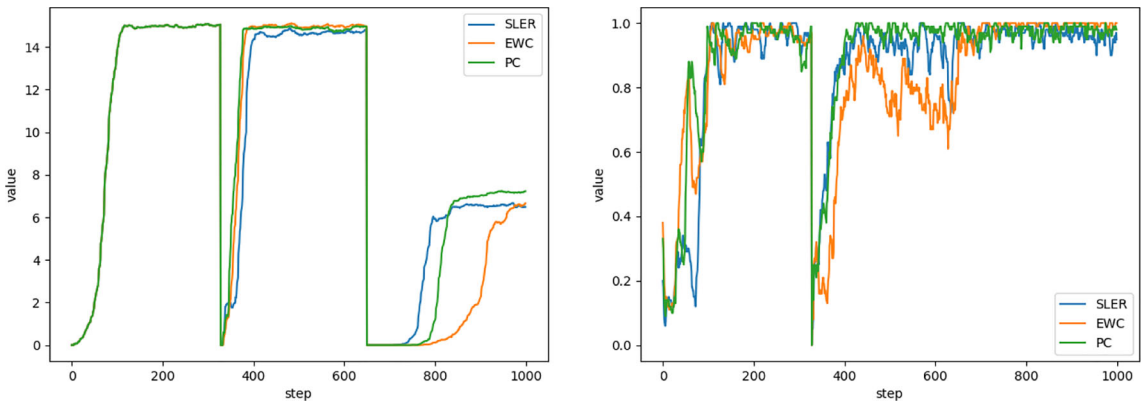
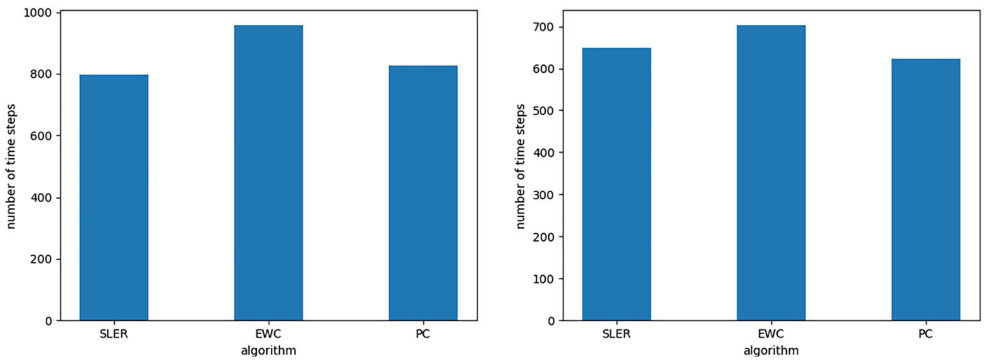


Fig. 13 On more tasks, our algorithm (SLER) gets comparable results with dual experience replay algorithm (DER), which reserves all tasks' real experience



(a) GridWorld (b) StarCraftII

Fig. 14 (**a**)Comparison of SLER to Progress & Compress (P&C) and Elastic Weight Consolidation (EWC) in GridWorld setting. (**b**) Comparison of SLER to Progress & Compress (P&C) and Elastic Weight Consolidation (EWC) in Starcraft II. Overall, our method performs comparably to or better than P&C, and significantly better than EWC and baseline



(a) GridWorld (b) StarCraftII

Fig. 15 (**a**)Comparison of SLER to Progress & Compress (P&C) and Elastic Weight Consolidation (EWC) in GridWorld setting. (**b**) Comparison of SLER to Progress & Compress (P&C) and Elastic Weight Consolidation (EWC) in Starcraft II. Overall, our method performs comparably to or better than P&C, and significantly better than EWC and baseline to or better than P&C, and significantly better than EWC and baseline

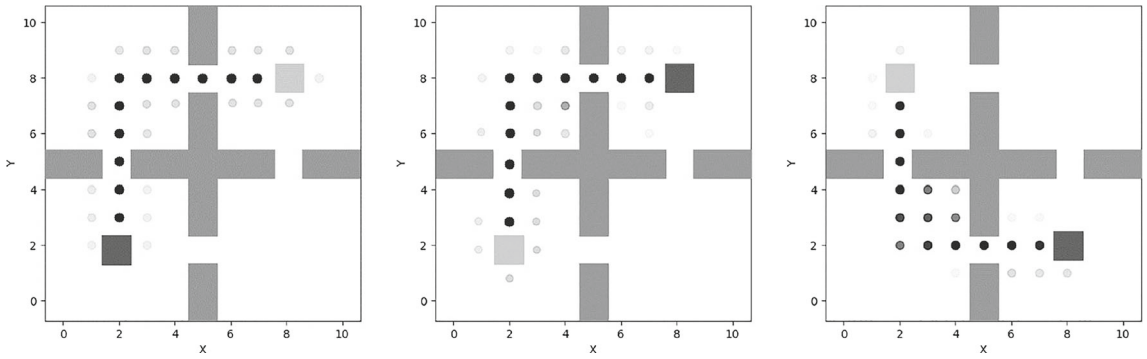


Fig. 16 The training process of the GridWorld I

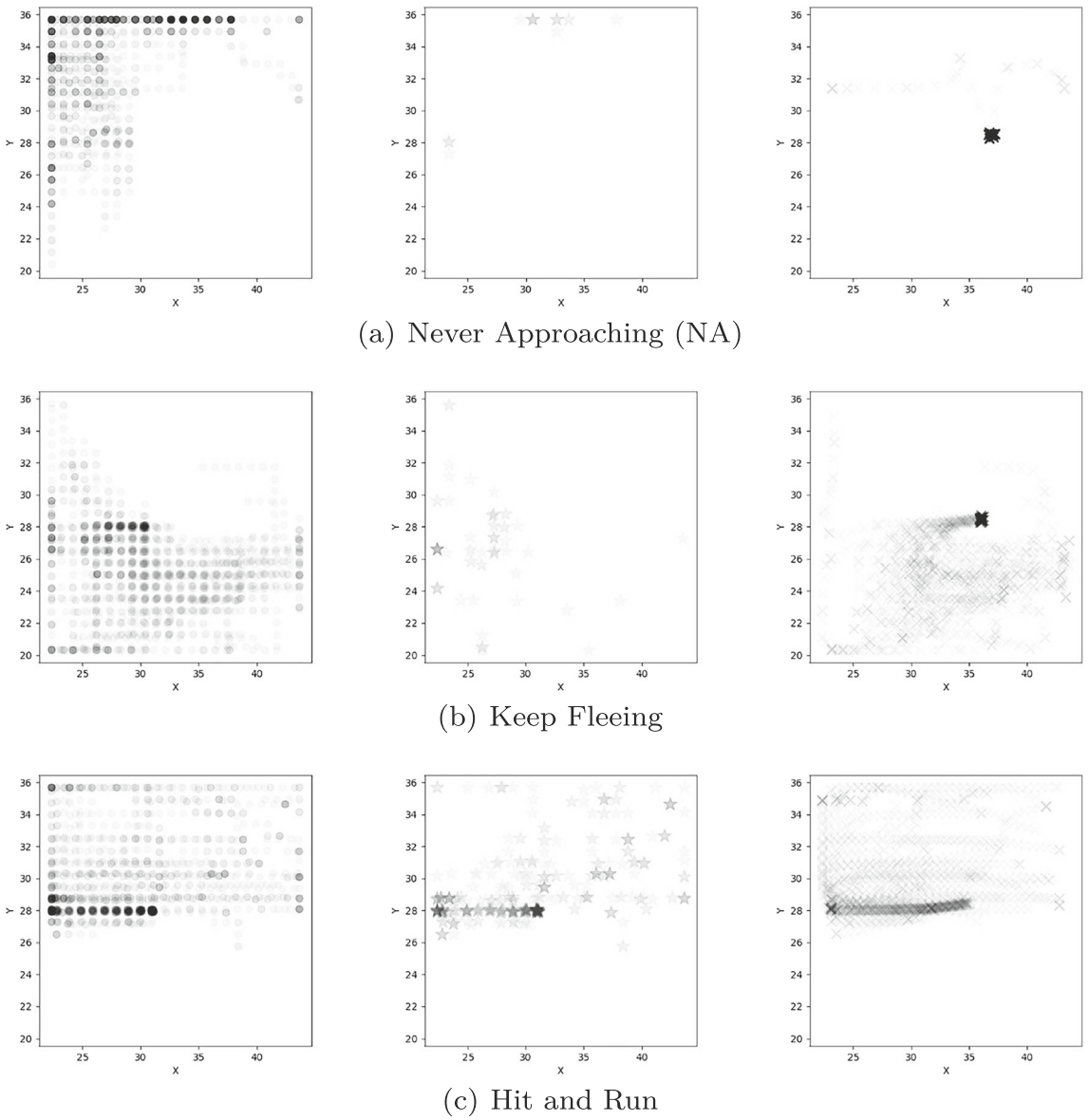


Fig. 17 The training process of the StarCraft II

6 Conclusion and future work

We propose a new continual reinforcement learning algorithm based on Self-generated Long-term Experience Replay. The algorithm uses the dual experience replay algorithm architecture as the benchmark and learns an environmental sample model through experience replay to generate simulation experience for knowledge retention. Instead of the traditional dual experience replay method, which needs to save all previous knowledge. We only retain the initial state, all actions, and all rewards in the short-term experience replay then generate all subsequent states in a self-generating manner to reduce the requirement of the previous experiences. The experimental results show that our algorithm can effectively mitigate the occurrence of catastrophic forgetting in the StarCraft II and GridWorld I continual learning environments, achieving relatively comparable results to the state-of-the-art continual learning method. In the GridWorld II environment, our algorithm also achieves relatively comparable results and presents the ability to handle more tasks. By directly inputting the image information of the GridWorld I environment for learning, the algorithm can obtain relatively comparable results.

Next, we will continue to explore more details of the StarCraft II environment and design more efficient and more stable task sets for the evaluation of continual learning algorithms. Also, we will study how to extend the proposed algorithm to the multi-agent environment, overcoming the problems of mutual interference and cooperation between multiple agents.

Acknowledgements We thank our colleagues for their collaboration and the present work. We also thank all the reviewers for their specific comments and suggestions. This work is supported by the National key research and development plan 2018YFC0832300.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

References

1. Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT Press, Cambridge
2. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, et al. (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484
3. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, et al. (2017) Mastering the game of go without human knowledge. Nature 550(7676):354
4. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529
5. Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P, et al. (2019) Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature 575(7782):350
6. Ring MB (1997) Child: A first step towards continual learning. Mach Learn 28(1):77
7. Thrun S (1995) A lifelong learning perspective for mobile robot control. In: Intelligent Robots and Systems. Elsevier, New York, pp 201–214
8. Rusu AA, Rabinowitz NC, Desjardins G, Soyer H, Kirkpatrick J, Kavukcuoglu K, PascanuR., Hadsell R (2016) Progressive neural networks. arXiv:1606.04671
9. Yin H, Pan SJ (2017) Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In Thirty-First AAAI conference on artificial intelligence
10. Tessler C, Givony S, Zahavy T, Mankowitz DJ, Mannor S (2017) A deep hierarchical approach to lifelong learning in minecraft. In Thirty-First AAAI conference on artificial intelligence
11. McClelland JL, McNaughton BL, O'reilly RC (1995) Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review 102(3):419
12. Parisi GI, Kemker R, Part JL, Kanan C, Wermter S (2019) Continual lifelong learning with neural networks: A review. Neural Networks
13. Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, Grabska-Barwinska A, et al. (2017) Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences p 201611835
14. Aljundi R, Babiloni F, Elhoseiny M, Rohrbach M, Tuytelaars T (2018) Memory aware synapses: Learning what (not) to forget. In Proceedings of the European Conference on Computer Vision (ECCV), 139–154
15. Schwarz J, Luketina J, Czarnecki WM, Grabska-Barwinska A, Teh YW, Pascanu R, Hadsell R (2018) Progress & compress: A scalable framework for continual learning. arXiv:1805.06370
16. Isele D, Cosgun A (2018) Selective experience replay for lifelong learning. In Thirty-Second AAAI conference on artificial intelligence
17. Lesort T, Caselles-Dupré H, Garcia-Ortiz M, Stoian A, Filliat D (2019) Generative models from the perspective of continual learning. In 2019 International Joint Conference on Neural Networks (IJCNN) IEEE 1–8
18. Wu C, Herranz L, Liu X, van de Weijer J, Raducanu B, et al. (2018) Memory replay gans: Learning to generate new categories without forgetting. In Advances in Neural Information Processing Systems 5962–5972
19. Shin H, Lee JK, Kim J, Kim J (2017) Continual learning with deep generative replay. In Advances in Neural Information Processing Systems, 2990–2999
20. Tanaka F, Yamamura M (1997) An approach to lifelong reinforcement learning through multiple environments. In 6th European Workshop on Learning Robots, 93–99
21. Rusu AA, Colmenarejo SG, Gulcehre C, Desjardins G, Kirkpatrick J, Pascanu R, Mnih V, Kavukcuoglu K, Hadsell R (2015) Policy distillation. arXiv:1511.06295
22. Maas AL, Hannun AY, NG AY (2013) Rectifier nonlinearities improve neural network acoustic models. In Proc. icml, 30:3

23. Schaul T, Quan J, Antonoglou I, Silver D (2015) Prioritized experience replay. arXiv:1511. 05952
24. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1):1929
25. Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets AS, Yeo M, Makhzani A, Küttler H., Agapiou J, Schrittwieser J, et al. (2017) Starcraft ii: A new challenge for reinforcement learning. arXiv:1708.04782

Chunmao Li received the B.Sc. degree in computer science and technology in 2012 and the M.Sc. degree in computer science and technology in 2015, and he is currently pursuing the PhD degree in department of computer science and technology. He researches include machining learning, deep reinforcement learning.

Yang Li received the B.S. degree in Computer Science and Technology from Xi'an Jiao tong University, China, in 2016, and a M.S. degree in Computer Science and Technology from Xi'an Jiao tong University in 2019. He is finding a Ph.D position now and his research interests are deep reinforcement learning, continual learning and computer vision.

Yinliang Zhao is currently a Professor in Xi'an jiao tong University, China. He received the the B.Sc. degree in in computer science and technology in 1983 and the M.Sc. degree in computer science and technology in 1988, and the PhD degree in computer science and technology in 1998. He researches include machining learning, deep reinforcement learning and parallel computing.

Peng Peng is currently a research scientist from inspir.ai, focusing on the research of game ai with deep reinforcement learning and continual learning. He was an algorithm expert from the Alibaba group and received his PhD degree from the department of computer science and engineering at the Hong Kong University of Science and Technology (HKUST). He has published over 10 papers in the field of Data Mining, Database and Machine Learning. He is also one of the main authors of StarCraft Commander (SCC), a superhuman AI in StarCraft II, which beat the top professional players in StarCraft II.

Xupeng Geng received the B.Sc. degree in computer science and technology in 2019. He is currently pursuing the M.Sc. degree with the department of computer science and technology. He researches include deep reinforcement learning.