IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 6, 2020, accepted July 12, 2020, date of publication July 15, 2020, date of current version July 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009329

Sample Efficient Reinforcement Learning Method via High Efficient Episodic Memory

DUJIA YANG, XIAOWEI QIN, XIAODONG XU, CHENSHENG LI, AND GUO WEI
CAS Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China, Hefei 230026, China

Corresponding author: Xiaowei Qin (qinxw@ustc.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFA0701603, and in part by the Natural Science Foundation of Anhui Province under Grant 2008085MF213.

ABSTRACT Reinforcement Learning (RL), especially Deep Reinforcement Learning (DRL), has made great progress in many areas, such as robots, video games and driving. However, sample inefficiency is a big obstacle to the widespread practical application of DRL. Inspired by the decision making in human brain, this problem can be solved by incorporating instance based learning, i.e. episodic memory. Many episodic memory based RL algorithms have emerged recently. However, these algorithms either only replace parametric DRL algorithm with episodic control or incorporate episodic memory in a single component of DRL. In contrast to preview works, this paper proposes a new sample-efficient reinforcement learning architecture which introduces a new episodic memory module and incorporates episodic thought into some key components of DRL: exploration, experience replay and loss function. Taking Deep Q-Network (DQN) algorithm for example, when combined with DQN, our algorithm is called High Efficient Episodic Memory DQN (HE-EMDQN). In HE-EMDQN, a new non-parametric episodic memory module is introduced to help calculate the loss and modify the predicted value for exploration. For the sake of accelerating the sample learning in experience replay, an auxiliary small buffer called percentile best episode replay memory is designed to compose a mixed mini-batch. We show across the testing environments that our algorithm is significantly more powerful and sample-efficient than DQN and the recent episodic memory deep q-network (EMDQN). This work provides a new perspective for other RL algorithms to improve sample efficiency by utilising episodic memory efficiently.

INDEX TERMS Episodic memory, sample efficiency, reinforcement learning, deep learning.

I. INTRODUCTION

The field of artificial intelligence, especially reinforcement learning (RL) algorithm, develops rapidly in recent years. RL algorithms have been successfully applied in many fields, such as robot control [1], [2], autonomous driving [3], playing video games [4] and navigation [5]. Taking deep reinforcement learning (DRL) for example, the algorithm named Deep Q-Network (DQN) got a superhuman level score in a range of Atari 2600 video games [6], [7]. There are more sophistic systems like AlphaGo which defeated international champion Li with 4:1 [8], AlphaStar that won top human contestants in StarCraft [9], etc. However, reinforcement learning algorithms still face many challenges at present. In addition to the performance problems, like other machine learning algorithms, DRL also has complexity problems: spatial complexity, computational complexity and sample complexity [10]. The cost of solving spatial, computational, and sample complexity of these successful cases is huge, especially for model-free reinforcement learning which is more practical and focused recently. In Atari 2600 game AI, the agent needs to train 50 million game frames (equivalent to 926 hours played by human players) to reach the game level that human players can get in a few minutes or dozens of minutes [7]. In the training of the go program, in addition to using the battle data and classic chess records of human expert players, AlphaGo also played 30 million rounds of self-play. As for the hardware configuration, according to [8], [11], the hardware for training AlphaGo is amazing: 1,202 CPUs and 176 GPUs. Therefore, it still needs continuous efforts from academia and industry to solve these complexity problems, for the sake of practical applications of DRL.

The associate editor coordinating the review of this manuscript and approving it for publication was Fuhui Zhou.

The two primary sources of sample inefficiency in DRL are incremental parameter adjustment and weak inductive bias [12]. Fortunately, according to subsequent research, both of these factors can be mitigated, allowing DRL to proceed in a much more sample-efficient manner. There have been proposed some extensions to improve DQN by modifying the basic architecture of DQN. Double DQN [13] decouples action selection and value estimation in target network to reduce the over-estimate. Authors in [14] propose a dueling DQN that splits action value into state value and advantage value. Prioritised Replay [15] further improves Double DQN by optimising the replay strategy of replay buffer in DQN. Authors in [16] combine the above three improved algorithms as well as multi steps to form an enhanced DQN, which is in a similar way to the more powerful Rainbow [17]. The return-based off-policy control algorithm [18] improves reward propagation and the back up mechanism of Q-learning.

Besides, there are other special insights from psychologists, psycholinguistics, and neuroscientists that might help improve RL. One specific technique is fast learning through episodic memory (EM). Inspired by the rapid complementary approach of decision making in the brain [19], there have been various attempts to incorporate external memory modules recently which improve the quality of decision making and fasten the learning process. This form of fast learning is supported by the hippocampus and related medial temporal lobe structures in our brain [20]. Hippocampal learning is thought to be instance-based [21], in contrast to the cortical system which represents generalised statistical summaries of the input distribution [22]. It is found that human beings utilize multiple learning, memories and decision making systems to efficiently implement the task in different situations. One of these external memories is episodic memory, which keeps an explicit record of past events and use this record directly as a point of reference when the agent is going to make new decisions.

Aiming at the problem of sample complexity commonly existing in model-free reinforcement learning algorithms, this paper focuses on how to improve the efficiency of samples and reduce the cost in practical application. Sample efficiency refers to the amount of data required for a learning system to attain any target level of performance [12]. There are some works propose to design new episodic memory module or incorporate these existing module to build a more efficient method. Based on episodic control [23], the agent records highly rewarding experiences and follows a policy which replays sequences of actions that previously yielded high returns. For example, MFEC [24] and NEC [25] propose different episodic memory modules for efficient decision making, instead of parametric neural networks. Others like EMDQN [26], EVA [27], EBU [28], etc., involve episodic memory to their algorithms to make them more powerful as well as more efficient.

However, the aforementioned methods either only replace parametric DRL algorithm with episodic control or incorporate episodic memory in single component of DRL. In this paper, we introduce a sample-efficient reinforcement learning architecture that make high efficient use of episodic memory with both new episodic memory module and incorporate episodic thought into some key components of DRL: exploration, experience replay and loss function. The main contribution of this paper is described as follows:

1) proposing a new episodic memory module: change the two episodic buffers in kd-tree of EMDQN (update in the manner of parameter updating of two networks in DQN) with just one episodic buffer of hash table; change the maximum update episodic memory module in EMDQN to a new module that updates with N-step learning and small difference monotonic increasing. These improvements help reduce the storage requirements as well as make the learning progress more stable.
2) introducing an architecture that make high efficient use of episodic memory: Based on the architecture of EMDQN who adds extra errors calculated by EM to the loss function of DQN, we also take advantage of EM to shift the value predicted by estimation network for exploration to the combined value of both predicted value and episodic memory-queried value. By this mean, the policy is closer to the optimal policy by better exploiting the experience in the replay memory (RM) and fill the RM with more useful data. What's more, we introduce an external small memory which stores recent top percentile best episodic data. The mini-batch is sampled between these so called percentile best episodic memory and the original replay memory in order to fasten the reward propagation and total training progress through the learning of more valuable samples.

The remainder of this article will be organized as follows. In Section II, the background of reinforcement learning is introduced. We will give a brief review of related works recently, which focuses on improving the performance or data efficiency of RL in Section III. In Section IV, we will propose a sample-efficient reinforcement learning architecture that make high efficient use of episodic memory by introducing a new episodic memory module and incorporating episodic thought into some key components of DRL. Experiment process and result analysis are provided in Section V to demonstrate the effectiveness of our model. Section VI concludes the paper with some discussion as well as future works.

II. BACKGROUND

A. REINFORCEMENT LEARNING

In fact, reinforcement learning is considered as a sequential decision making problem that an agent interacts with an environment over discrete time steps. Usually, this problem is modeled as Markov Decision Process (MDP) [29]: $M = (S, A, R, P, s_0)$, where S is the state space, A is the action space, R is the reward function, P is the transition probability between states and s_0 is the initial state. At the time step t, the agent observes a state $s_t \in S$ and takes an action $a_t \in A$ according to the policy $\pi(a|s_t)$. Then the environment transfers to a new state s_{t+1} with probability $P(s_t, a_t)$ and returns a scalar reward $r_t \in R$ to the agent. The goal of the agent is to learn an optimal policy π^* that is able to get the maximum expectation of cumulative discounted return $R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}$, where $\gamma \in (0, 1]$ is the discount rate.

When it comes to estimating the value of states and actions, there are value function $V_\pi(s)$ and action value function $Q_\pi(s, a)$ respectively. Starting from state s and following a policy π, $V_\pi(s)$ is the expected amount of discounted return over the future. The action value function $Q_\pi(s, a)$ is the $V_\pi(s)$ when considers the starting action a:

$$V_\pi(s) = \mathbb{E}_\pi[R_t|s_t = s] \tag{1}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a] \tag{2}$$

For value based RL, the agent tries to learn an approximation optimal policy by acting greedily on these values. Taking the classical RL algorithm Q-learning [30] for example, it's a table-based method and uses the well-known temporal difference (TD) to iteratively update $Q(s, a)$ as well as find the optimal value:

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{3}$$

B. DEEP Q-NETWORK

Deep Q-Network (DQN) [7] is the first algorithm that combines deep learning with Q-learning to solve the problem of capacity limitation and sample correlation of the traditional table-based RL. The DQN model is parameterized by weights and biases denoted as θ and the action value is estimated by neural networks denoted as $Q(s, a|\theta)$. At the iteration i during training, a mini-batch of experience $E = \{e_1, \ldots e_n\}$ is sampled uniformly from the replay memory for updating the networks, where $e_t = (s_t, a_t, r_t, s_{t+1})$ and n is constant. DQN also adopts the idea of TD to calculate the loss function of the mini-batch:

$$L(\theta_E) = \mathbb{E}_{((s_t,a_t,r_t,s_{t+1})\ D)}[(y_t - Q(s_t, a_t|\theta_E))^2] \tag{4}$$

where θ_E is the parameters of evaluation network and target value $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'|\theta_T)$ is computed by the target network with parameters θ_T. θ_E is updated by minimizing the loss function through gradient descent and θ_T is the copy of θ_E every setting steps.

The main contribution of DQN is: 1) using deep neural network (DNN) to approximate value function; 2) introducing experience replay from RM to train the RL off-line; 3) using a separate target network to calculate TD errors.

C. EPISODIC MEMORY

In this paper, the environments we considered are episodic, in which each episode ends in a special state called the terminal state and followed by a reset to a standard starting state or to a sample from a standard distribution of starting states [29]. So for episodic RL, the expected discounted return is denoted as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the total time steps at the ending of each episode. Accordingly, the episodic reward is the sum of total real rewards in each episode, e.g. total scores in video games: $Score = \sum_{t'=1}^{T} r_{t'}$.

III. RELATED WORKS

In this section, we will review the related works of various recent reinforcement learning methods which incorporate episodic memory to improve performance as well as data efficiency.

A. NEW EPISODIC MODULE

Model Free Episodic Control (MFEC) is one of the first few works which involved episodic memory in reinforcement learning [24]. The author leverages an episodic control model: record highly rewarded experiences and follow a policy that replays sequences of actions that previously yielded high returns. This model is non-parametric and data-efficient when compared to traditional gradient-based DRL. [25] proposes Neural Episodic Control (NEC), which consists of three components: a convolutional neural network that embeds raw pixel images into slow-changing keys, a differentiable neural dictionary (DND), and a final network that converts context-based lookup from the DND to estimate values for action selection. DND is an episodic memory module with fast-updating values, and allows estimating the value of a new state according to the similarities between stored neighbor states which is more reasonable compared to MFEC. However, these two table-based methods rely on finding k nearest neighbour (kNN) states which is very time-consuming and lack generalization when compared to parametric DRL. Based on advantage actor critic architecture, [31] introduces a trainable episodic memory module which uses a reservoir sampling technique to avoid maintaining all visited states in memory. States from the memory are drawn from a distribution over all n-subsets of visited states parameterized by weights which requires quite complicated computation. NEC-RP [32] just reduces the number of parameters to learn by switching the fully connected layer of NEC to random projection which is slightly more stable and efficient than NEC.

B. INCORPORATION OF EPISODIC MEMORY

Recently, there have been growing interests in introducing episodic thought into some key components of deep reinforcement learning algorithms, such as exploration, experience replay and loss function.

1) exploration: Authors in [33] propose a new exploring policy that encourages the agents to find states with curiosity. Computed by EM, this curiosity is a novelty bonus and is summed up with the real task reward to provide the combined reward for RL learning. Instead of trying to find surprising states, Ephemeral Value Adjustments (EVA) [27] modifies the value for action selection predicted by neural network with an estimated value function which is found by planning over nearest

experience tuples from the replay buffer. However, both the curiosity computation and memory-based planning of EVA are quite model-complex and time-consuming.

2) experience replay: EBU [28] performs backward updating in deep-learning setting by using episodic memory instead of the random batch-sampling in experience replay. As for AE-DDPG in [34], the author designs two memory buffers for experience replay, namely "Memory" and "HMemory". The former is the same as the RM in DQN, while the latter stores data in trajectories. However, "HMemory" only memorizes whole episodic data who gets the highest score at present, which possibly leads to severe lacking of sample diversity (see Section IV).

3) loss function: Authors in [35] propose a NEC2DQN algorithm which simply replaces the target network with NEC at the beginning of training to form combined target values and to compute loss function further. It learns faster than Double DQN or N-step DQN in the Atari game Pong, but it impairs the advantage of generalization of DQN and lacks the continuous and effective use of episodic memory. Recently, the author in [26] combines parametric module of DQN with non-parametric module of episodic control with the purpose of improving both sample efficiency as well as module generalization. This EMDQN method is better than DQN and surpasses both MFEC and NEC. However, the utilization of episodic memory in EMDQN is single, because it is merely been applied to the loss function of DQN. Besides, its policy for updating value with maximum replacement is slightly aggressive which may cause stability problem in training. What's more, EMDQN use kd-tree to construct the memory table with two same size buffers updated in the manner of asynchronous updating of two networks in DQN, which is time-consuming and memory-consuming.

In this paper, drawing on the experience of EMDQN, we propose a new sample-efficient reinforcement learning method which introduces a new EM module and incorporates episodic thought into some key components of DRL efficiently: exploration, experience replay and loss function. Experiments in Section V show our algorithm is significantly more sample-efficient than both DQN and the better one — EMDQN.

IV. METHOD

Our work focuses on proposing a new perspective for other RL algorithms to high efficiently use episodic memory to improve sample efficiency. In this section, we take DQN algorithm for example and introduce the overall architecture of our High Efficient Episodic Memory DQN (HE-EMDQN) algorithm in detail, which is illustrated in Fig. 1 and descried in *Algorithm* 1 and *Algorithm* 2. The solid lines in Fig. 1 represent the working flow of traditional DQN, while the dash lines are the pipeline we designed to make an efficient use

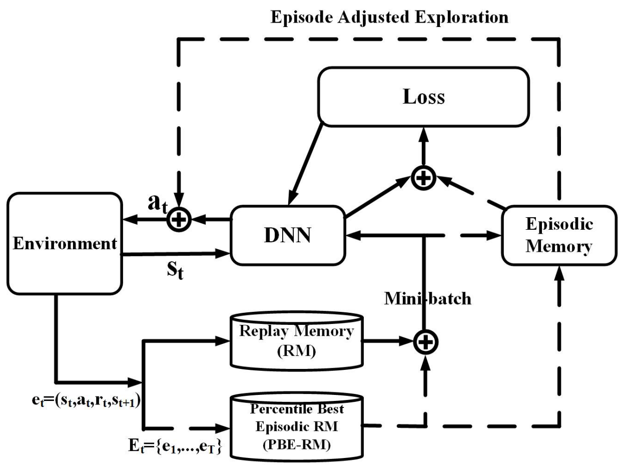

FIGURE 1. High efficient episodic memory DQN (HE-EMDQN).

of episodic memory. At the time step t, the agent observers state s_t from the environment and takes action a_t according to the episode adjusted exploration (EAE) strategy. Then the environment transforms to new state s_{t+1} and rewards the agent with r_t. The experience $e_t = (s_t, a_t, r_t, s_{t+1})$ is stored in replay memory (RM). When this episode comes to an end, episodic memory (EM) is updated in a new episodic control manner and the whole episodic data (also called trajectory) $E = \{e_1, \ldots e_T\}$ will be pumped into the percentile best episode replay memory (PBE-RM) if its episodic reward is larger than a dynamic bound. Both replay memories are sampled to compose a mixed mini-batch which is used to compute a combined loss and update the DQN networks further. Below, we will introduce the improvement of each aspect of the algorithm in detail.

A. NEW EPISODIC MEMORY MODULE

As mentioned in Section I, MFEC and NEC are two famous episodic control based methods and are both non-parametric. They are all tabular episodic memory modules with different ways of updating. We denote the table in episodic memory as $Q^{EC}(s, a)$.

At the end of each episode, MFEC updates the table $Q^{EC}(s, a)$ by changing the value within the table to a bigger expected discounted return directly:

$$Q^{EC}(s_t, a_t) = \begin{cases} R_t & \text{if } (s_t, a_t) \notin Q^{EC} \\ \max\{Q^{EC}(s_t, a_t), R_t\} & \text{otherwise} \end{cases} \tag{5}$$

This update mode propagates rewards very fast but may cause some values in $Q^{EC}(s, a)$ change too much to keep a stable learning.

For NEC, values in this module are updated in the similar way as the classic tabular Q-learning algorithm (Eq. 3) but with N-step instead. Unlike MFEC, $Q^{EC}(s_t, a_t)$ is calculated as a weighted average of its kNN states' values in DND, whose weights are given by normalised distance between the lookup key and the corresponding neighbor key in EM.

Algorithm 1 High Efficient Episodic Memory DQN (HE-EMDQN)

```
Input: $n_{step}$, $update_{step1}$, $update_{step2}$, mini-batch size $k$, $\alpha$, $\eta$, $\lambda_0$, $\lambda_1$, $\lambda_2$
1: Initialize $tr = []$ $RM = []$, $PBE - RM = []$, $EM = []$
2: for each episode do
3:     for t = $1, 2, 3, \dots$ T do
4:         Receive observation $o_t$ from environment.
5:         Let $s_t = \phi(o_t)$.
6:         With probability $\epsilon$ select a random action $a_t$.
7:         Otherwise estimate $Q_{EVA}(s_t, a)$ for each action $a$ via (13) and select $a_t = \text{argmax}_a Q_{EVA}(s_t, a)$.
8:         Execute action $a_t$, observe reward $r_t$ and next state $s_{t+1}$.
9:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $RM$ and trajectory $tr$.
10:        if $t >= n_{step}$ then
11:            Update $EM$ via function UPDATE_EM($tr$, $True$,$n_{step}$).
12:        end if
13:        if t MOD $update_{step1} == 0$ then
14:            Sample mini-batch according to the sampling strategy in (11).
15:            Calculate mini-batch's loss via (14) and importance sampling weights via (10).
16:            Update $\theta_E$ in the evaluation network.
17:        end if
18:    end for
19:    Update $EM$ via function UPDATE_EM ($tr$, $False$,$n_{step}$)
20:    Calculate $Score$ and $score\_bound$ via (9)
21:    if $Score >= score\_bound$ then
22:        Pump $tr$ into $PBE - RM$
23:    end if
24:    Let $tr = []$
25:    if t MOD $update_{step2} == 0$ then
26:        Copy $\theta_E$ to $\theta_T$.
27:    end if
28: end for
```

This update module is more stable and reasonable than MFEC. However, this weighted average manner decreases the advantage of the maximum return propagation. What's worse, NEC needs to find kNN states and calculate weights frequently, which is very time-consuming.

In our algorithm, we design a new episodic memory module that combines the fast return propagation of MFEC and the stable learning strength of NEC. Firstly, the observation of high dimension o_t of the environment is mapped to an embedding state s_t through an embedding function ϕ. There are many embedding functions, such as Gaussian function [36], Convolution neural network [25], simihash [37]. Since the embedding work is not our focus, we use the Gaussian projection which approximately preserves relative distances in

Algorithm 2 Update Episodic Memory

```
Input: $tr$: sample sequence in one episode, $b\_N\_step$: whether to use N-step learning estimation
    function UPDATE_EM($tr$, $b\_N\_step$, $n\_step$)
2:      Get the sequence length $T$ of $tr$
        if $b\_N\_step$ then
4:          $seq = tr(T - n\_step + 1 : T)$, $(s_t, a_t, r_t, s_{t+1}) = tr(T - n\_step)$
            Compute N-step learning estimation $Q^{EC_N}_{new}(s_t, a_t)$ via (7) and (8)
6:          Update $Q^{EC}(s_t, a_t)$ in $EM$ via (6)
        else
8:          for t = T, T $- 1, \dots, 1$ do
                $(s_t, a_t, r_t, s_{t+1}) = tr(t)$
10:             $Q^{EM}_{new}(s_t, a_t) = R_t(s_t, a_t)$
                Update $Q^{EC}(s_t, a_t)$ in $EM$ via (6)
12:         end for
        end if
14: end function
```

the original space according to the Johnson-Lindenstrauss lemma [36], same as MFEC and EMDQN.

For module updating, we compose fast return propagation, tabular Q-learning algorithm and N-step learning into our updating strategy. For the new $Q^{EM}_{new}(s_t, a_t) = R_t(s_t, a_t)$, it will be added to episodic memory directly if it is not in the memory. According to [23], the episodic control is given as "each time the subject experiences a reward that is considered large enough (larger than expected a priori) it stores the specific sequence of state-action pairs leading up to this reward, and tries to follow such a sequence whenever it stumbles upon a state included in it. If multiple successful sequences are available for the same state, the one that yielded maximal reward is followed.". Hence, $Q^{EM}(s_t, a_t)$ should be updated if and only if $Q^{EM}_{new}(s_t, a_t) > Q^{EM}(s_t, a_t)$. Since EM is used to fasten the learning progress of parametric DRL, the small step-sizes in learning is needed in case of 'catastrophic interference' [12]. To solve the problem, double buffers are used in kd-tree in EMDQN: one changes every episode, while the other real EM values are updated following the updating pace of target network. For our new episodic memory module, there is only one buffer in EM and it is updated in the same way as Eq. 3 whose stability is ensured by the small learning factor α and convergence is shown in [30]. In brief, this new module is calculated as follows:

$$Q^{EM}(s_t, a_t) = \begin{cases} Q^{EM}_{new}(s_t, a_t) & \text{if } (s_t, a_t) \notin Q^{EM}, \\ Q^{EM}(s_t, a_t) & \\ +\alpha[Q^{EM}_{new}(s_t, a_t) - Q^{EM}(s_t, a_t)] & \\ \qquad \text{if } Q^{EM}_{new}(s_t, a_t) > Q^{EM}(s_t, a_t), \end{cases} \tag{6}$$

where the learning rate $\alpha \in [0, 1]$.

Specially, for some environments, the agent needs to take action with far many steps to complete an episode. For example, the Atari game Pong usually ends with thousands of steps. In this case, episodic memory will get quite few updating, leading to a large variance. So we introduce N-step learning in a similar way as NEC. When the past steps in each episode is no less than hyper-parameter N, the $Q^{EM}_{new}(s_t, a_t)$ in Eq. 6 is replaced by N-step estimated value $Q^{EC_N}_{new}(s_t, a_t)$:

$$Q^{EC_N}_{new}(s_t, a_t) = \sum_{j=0}^{N} \gamma^j r_{t+j} + \gamma^N \max_{a'} Q_{mix}(s_{t+N}, a'), \quad (7)$$

where $Q_{mix}(s_{t+N}, a')$ is the weighted sum of $Q(s_{t+N}, a'|\theta_T)$ from target network and the $Q^{EM}(s_{t+N}, a')$:

$$Q_{mix}(s_{t+N}, a') = \begin{cases} Q(s_{t+N}, a'|\theta_T) & \text{if } (s_{t+N}, a') \notin Q^{EM} \\ \lambda_0 Q(s_{t+N}, a'|\theta_T) & \\ +(1-\lambda_0)Q^{EM}(s_{t+N}, a') & \text{otherwise,} \end{cases} \quad (8)$$

$\lambda_0 \in [0, 1]$ is a constant parameter. This trick of weighted sum is used to utilize the episodic memory to reduced the common overestimation problem of DQN. Details are shown in *Algorithm* 2.

B. MIXED EXPERIENCE REPLAY

The experience replay with RM in DQN is biologically inspired that sequences of experience are replayed, either during awake resting or sleep in the hippocampus of rodents [38]. What's more, the biological study [39] and Peak-End Rule [40] show that people tend to remember scenes with high value, the peak returns and the end returns greatly. So the author in [34] redesigns experience replay by introducing the idea of episodic control called AE-DDPG. Based on DDPG algorithm, AE-DDPG uses two experience memory buffers, namely called "Memory" and "HMemory". The first one is the same as the RM in DQN, while the latter "HMemory" only stores episodic data whose episodic reward surpasses the best score at present, i.e. best episode (BE). Then mini-batches using for updating the networks of DDPG are sampled with fixed probability between these two memories respectively. Although AE-DDPG has higher learning efficiency than DDPG in experiments, this simple mixed experience replay policy has the following problems:

- Loss of Sample Diversity: The "HMemory" in AE-DDPG only stores episodic data who gets the maximum score at present. This strategy limits the memory with small subset of the experience samples, in other word with poor diversity. This lack of diversity will make the system prone to over-fitting, and may further lead to an unstable performance [15]. For example, because of random exploration, one episode of the Atari game WizardOfWor may get a high point by accident in the early stage of training, when the network is far from stabilization. On this occasion, "HMemory" will store this episodic data and rarely change for a long time, which is quite sample-monotonous.
- Introduce Bias: In machine learning, the expected value estimation of random updates depends on the updates from the same distribution as the expectation. Mixed experience replay changes the sample distribution with some parts of samples been replayed more frequently but uncontrollable and thus introduces bias for the final convergence.

In light of the existing problems above, we introduce a new mixed experience replay policy with percentile best episode replay memory and importance sampling. Details are shown as follows:

- Percentile Best Episode Replay Memory: Different stages of training require different diversity of samples. The usual policy of the decreasing exploring rate in almost all RL algorithms shows that the requirements of diversity is decreasing as well. Hence in our storage rule, when one episode is finished, this trajectory is pumped into PBE-RM if its *Score* is no less than a dynamic value *score_bound*. After every episode, *score_bound* is calculated as:

$$score\_bound = percentile(Past\_Rew, dy\_per), \quad (9)$$

where *Past_Rew* is the last M episodic rewards and percentile value *dy_per* is changing dynamically according to the training steps. The higher the percentile, the higher the value density in PBE-RM, and the samples in it are less diverse as well. Fig. 2 shows *Score* of each episode during training in two games with DQN ("score"). Inspired by the well-known "80/20 Rule" that in a given process, 80% of the impact comes from 20% of the input, this *score_bound* should maintain a relatively stable high level while reflect the overall trend of the agent's real-time score during training for the sake of diversity and stability. So we set *dy_per* stepping up with initial value 80 and step increasing, and it is calculated as the percentile value in a sliding window of the last 200 episodic rewards (Fig. 3). In Fig. 2, we can see that our dynamic bound ("score_PBE") follows the whole training trend while keeps a relative stable rising (stability) and holds a high but not the highest value (diversity) when compared to the 80 percentile score ("score_80") and best episode score ("score_BE").
- Importance Sampling: Importance sampling [41] is a common technique for estimating an expectation under one distribution given samples from a different distribution. Refer to [15], we correct this introduced bias through applying importance-sampling weights

$$w_t = \left(\frac{1/N_{RM}}{1/N_{PBE-RM} * p}\right)^{\eta} \quad (10)$$

to the loss of every sample of mini-batch that sampled from PBE-RM δ_i, where N_{PBE-RM} and N_{RM} represent the two replay memory sizes at present, $\eta \in [0, 1]$ is increasing with steps, and p is the probability to involve mixed experience replay rule. These weights can be folded into the Q-learning update by using $w * \delta_i$, instead of the usual δ_i.

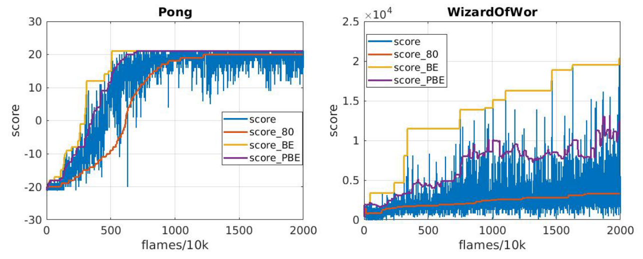

FIGURE 2. Scores of all training episodes of DQN and 80 percentile score ("score_80"), best episode score ("score_BE") and our dynamic *score_bound* ("score_PBE") of these scores.

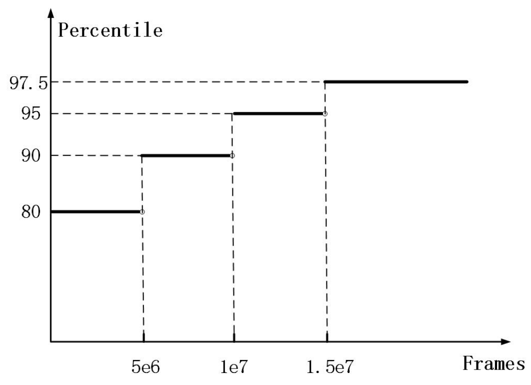

FIGURE 3. The dynamic percentile of percentile best episode replay memory.

At the stage of experience replay, two methods of sampling are combined to form the mini-batch:

$$\mathcal{B} = \begin{cases} Combine[Sample_{RM}((1-\mu)*k), \\ \quad Sample_{PBE-RM}(\mu * k)] & \text{if } p < \rho, \\ Sample_{RM}(k) & \text{otherwise,} \end{cases} \tag{11}$$

where k is the mini-batch size, $\mu \in [0, 1]$ is the combination ratio that controls the number of samples sampling between RM and PBE-RM. $Sample_{RM}(k)$ means random sampling k samples from RM. $Sample_{PBE-RM}(k)$ is a similar definition. During every experience replay, we use a random variable p to switch sampling rule between normal experience replay and mixed experience replay: if and only if p is smaller than a constant $\rho \in [0, 1]$, the mixed experience replay is executed. By this means, we increase the value density of samples for learning while keep a good diversity of the mini-batch.

C. EXPLORATION AND LOSS

Exploration and exploitation is a common dilemma in RL, whose focus is how to take action at every step. The agent may take many futile actions in the environment if its exploration policy is bad, leading the algorithm sample-inefficient in the end. EVA modifies the values for action selection during exploration by planning over experience tuples from the replay buffer near the current state:

$$Q_{EVA}(s_t, \cdot) = \lambda Q(s_t, \cdot|\theta_E) + (1-\lambda)\frac{\sum_{k=1}^{K} Q_{NP}(s_k, \cdot)}{K}, \tag{12}$$

where s_k is one of the K kNN states of the current state s_t and $Q_{NP}(s_k, \cdot)$ is its planning values of all actions. Since EVA needs to find kNN states in an extra buffer, it is inevitably time-consuming. Besides, its trace computation algorithm of estimating Q_{NP} just use the segment information of episode. We apply the thought of EVA to our method, but take a simple policy:

$$Q_{EVA}(s_t, a) = \begin{cases} Q(s_t, a|\theta_E) & \text{if } (s_t, a) \notin Q^{EM}, \\ \lambda_1 Q(s_t, a|\theta_E) \\ +(1-\lambda_1)Q^{EM}(s_t, a) & \text{otherwise,} \end{cases} \tag{13}$$

where $\lambda_1 \in [0, 1]$. In every action selection of exploitation, the agent looks up every action of state s_t in episodic memory and selects an action according to $a_t = \operatorname{argmax}_a Q_{EVA}(s_t, a)$. The value adjustment is executed if and only if (s_t, a) is contained in episodic memory. This simple exploration policy utilise episodic memory to efficiently exploit environment while it does not cost too much time.

In the light of the research in neuroscience [42], striatum (i.e. reflex) and hippocampus (i.e. memory) are two learning systems in human brain, where they compete and cooperate with each other to constitute a third system that generates motivational (outcome-predictive) signals in decision making. EMDQN mimic this third system in the brain by combining TD errors $(Q(s, a|\theta_T) - Q(s, a|\theta_E))^2$ and EM errors $(Q^{EM}(s, a) - Q(s, a|\theta_E))^2$. We adopt this idea who combines the two errors from DQN and episodic memory respectively:

$$L(s, a) = \begin{cases} (Q(s, a|\theta_T) - Q(s, a|\theta_E))^2 & \text{if } (s, a) \notin Q^{EM}, \\ (Q(s, a|\theta_T) - Q(s, a|\theta_E))^2 \\ +\lambda_2(Q^{EM}(s, a) - Q(s, a|\theta_E))^2 & \text{otherwise,} \end{cases} \tag{14}$$

where $\lambda_2 \in [0, 1]$. Note that EM is derived from the non-parametric EC learning system. Like the target network in DQN, its output values work as another true values and are used to calculate EM errors. Similar to optimality tightening method [43], which has been proven to speed up reward propagation through tightening strategies, this learning signal is the special case of the following formula when $L^{max} = U^{min} = Q^{EM}(s, a)$:

$$\min_{\theta} \sum_{(s_j, a_j, r_j, s_{j+1}) \in \mathcal{B}} [(q_j - y_j)^2 + \lambda(L_j^{max} - q_j)_+^2 + \lambda(q_j - U_j^{min})_+^2], \tag{15}$$

where q_j and y_j are the output of evaluation network and target network of (s_j, a_j) respectively, λ is a penalty coefficient and $(x)_+ = \max(0, x)$ is the rectifier function. L^{max} and U^{min} are the largest lower bound and the smallest upper bound of (s_j, a_j) respectively, which are computed as constraints in a constant number of future and past steps in one episode according to the Bellman optimality equation. The difference is that these bonds are episodic rewards from EM in the modified loss function. By distilling information from EM to the parametric model, additional constraints who constrain the output of estimation network to be around Q^{EM} are added to DQN. Thus, the fast converging property of episodic memory is introduced to speed up the reward learning in DRL.

We will compare our HE-EMDQN with EMDQN in next section.

V. EXPERIMENTS AND RESULTS

We test our algorithm on the Arcade Learning Environment provided by OpenAI Gym [44], which is a benchmark suite of Atari 2600 games. These games have high dimensional observations and require complex policies to acquire high expected rewards. They are widely used for performance comparison in RL at present.

A. EXPERIMENTAL SETUP

Because of the limited computing resources in our lab, we evaluate HE-EMDQN, EMDQN and DQN over 20 randomly selected games (according to the alphabetical order) of Atari 2600, which contain the games with good performance and were shown in EMDQN ('Defender' is not included in the recent version of atari in OpenAI Gym). See these games in Table 1.

TABLE 1. Score comparison of EMDQN, DQN and our algorithm.

Game	DQN (20M)	EMDQN (20M)	HE-EMDQN (20M)
Asterix	4111.7	4246.7	**5641.7**
Atlantis	140120	197010	**306640**
BattleZone	32167	**36233**	32500
Breakout	**365.9**	240	326
ChopperCommand	3133.3	3846.7	**5320**
DemonAttack	3072.2	3781.7	**4390.5**
Enduro	1579.1	1637.7	**1878.2**
FishingDerby	20.3	**31.2**	29.6
Gravitar	545	766.7	**1210**
Hero	7324.3	7782.5	**15748.3**
JamesBond	**605**	460	515
Krull	8567.4	8260.8	**9481.9**
Pong	20.8	20.9	**21**
Qbert	6460	10645	**14332.5**
Robotank	29.4	48.7	**50.4**
StarGunner	**10000**	2923.3	3656.7
TimePilot	5226.7	**8723.3**	8400
UpNDown	15863	17350	**19962**
WizardOfWor	3493.3	**5470**	5023.3
Zaxxon	9070	13320	**14660**

The parameters of EMDQN and DQN are all set the same as [26]. As for HE-EMDQN, the networks and basic hyper-parameter settings are set as DQN. Like EMDQN, we also use Gaussian projection to map the observation to 4 dimension state, whose distribution obeys $N(0, 1/\sqrt{4})$. In contrast, the projection precision is limited to 5 decimal places to form a string key for storage and query. The rewards in all games are clipped to [−1,1]. The following hyper parameters are tuned on StarGunner and Pong. The buffer size of PBE-RM is 3e4, and the N-step estimation is set to 200 steps. If EM or PBE-RM is full, the oldest sample is discarded to store the new one. The *score_bound* is the *dy_per* percentile of the last 200 episodic rewards, where the *dy_per* is stepping up from 80 to 97.5 (Fig. 3). The learning rate α is set to 0.25. At the stage of experience replay, we set the mini-batch $k = 32$ with the possibility of mixed experience replay $\rho = 0.9$ and combination ratio $\mu = 0.1$. The adjustment parameter λ_0, λ_1, λ_2 are all set to the same value 0.1.

We train these three algorithms with 20 million frames and evaluate them every 250000 frames. During evaluation, every agent runs for 30 episodes with 30 no-op operations at the start of each episode [24]. The final result of each evaluation is the average of these 30 scores. Each game is run 3 times with different random seeds.

B. RESULTS

1) RESULTS ON GAMES

The highest average testing score of every training for score comparison among our algorithm, EMDQN and DQN are reported in Table 1, just as [7]. This score is a common measure that shows the best performance of an algorithm. From Table 1, we can find that HE-EMDQN is far better than DQN on 17 games. For EMDQN, our method scores much higher on 16 games (80% percent on 20 games), while just slightly worse than EMDQN in the other 4 games. Following [13], we also compute human-normalized scores across games among these three algorithms to evaluate the overall performance:

$$score_{normalized} = \frac{score_{agent} - score_{random}}{score_{human} - score_{random}}. \tag{16}$$

Note that the 'random' and 'human' scores of all games are the same as shown by [7]. Results (mean and median) are shown in Table 2.

TABLE 2. Performance comparisons on mean and median human-normalized scores over 20 games.

	DQN (20M)	EMDQN (20M)	HE-EMDQN (20M)
Mean	231.0%	251.3%	**316.3%**
Median	108.7%	148.0%	**167.1%**

Besides the final results in the two tables above, we show the testing curves of 8 games during training in Fig. 4. These curves indicate the progress of average best performance of all algorithms as training continues. Obviously, HE-EMDQN performs far better than DQN as well as EMDQN, especially in Gravitar, Pong and Qbert. Taking Pong for example, although the final results in Table 1 are close, their sample efficiency varies greatly. It takes EMDQN 9.75 million training frames to get the average testing score of 20 (the highest score is 21 for Pong), while only 4.75 million training frames are needed for HE-EMDQN.

Seeing from Table 1, Table 2 and Fig. 4 together, HE-EMDQN shows a better performance than DQN and EMDQN. In other words, our improvement measures on HE-EMDQN do help improve the performance of EMDQN as well as sample efficiency. In next section, extra experiments are done to analysis the effects of different improvement measures of our architecture in improving sample efficiency respectively.

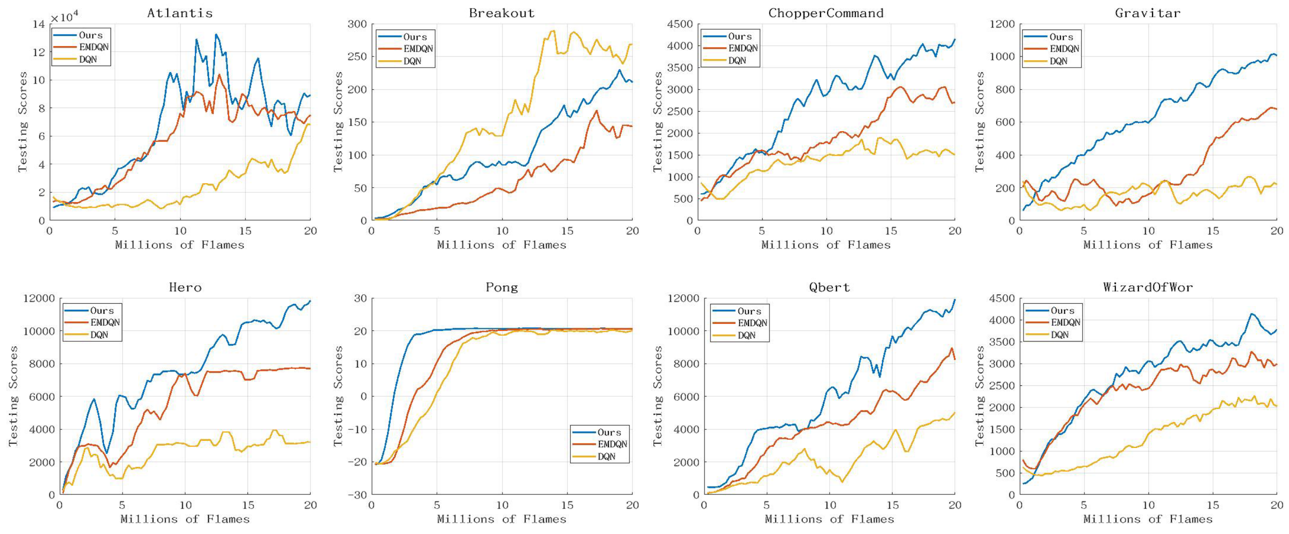

FIGURE 4. Testing curves of HE-EMDQN (Ours), EMDQN, DQN on representative games. Scores are smoothed using moving average over 4 testing epochs. Each game is run 3 times with different random seeds.

2) ANALYSES ON OUR ARCHITECTURE

First of all, We replace the maximum update module of EMDQN with our new EM module and compare it with EMDQN on 2 games. In order to see the learning ability further, the real training curves are shown in the left column of Fig. 5. It is worth noting that the training mean scores are the mean of scores of last 100 episodes during training. Results show that our new EM module works as good as the maximum update module of EMDQN, and even better than EMDQN. However, EMDQN uses kd-tree who contains a state buffer and a real tree to construct the memory table. The buffer updates in real time and is copied to the tree at the same pace of updating target network for the sake of stability. Thanks to the small monotonic rise policy of our episodic module which ensures a stable updating of EM, we use hash table with only one buffer in HE-EMDQN. In the experiments, our EM contains a hash table with buffer size 10^7. The EM in EMDQN is a kd-tree table with buffer size of 5×10^6 per action, whose total size is double. For some games with larger action space, such as Jamesbond, which contains 18 possible actions, the fixed buffer size of EMDQN is $2\times18\times5\times10^6$. In addition, as the complexity of hash table is $O(1)$ which is smaller than the complexity $O(log_2 N)$ of kd-tree, the training time of EMDQN with our new EM module is smaller than the original EMDQN in the experiments as well.

As for experience replay, the whole HE-EMDQN (Ours) and HE-EMDQN with best episode replay memory (Ours-BE) are tested (Fig. 6). Best episode replay memory is the policy proposed by AE-DDPG. HE-EMDQN with PBE-RM is a little worse than HE-EMDQN with BE-RM in the early stage, but stably better than the latter in the later stage. Specially, HE-EMDQN with BE-RM confronts a continuous performance degradation in WizardOfWor in the late stage which is not the problem of HE-EMDQN with PBE-RM. We argue that the strategy of percentile best and importance sampling help reduce the problem of diversity in BE-RM which is quite important for the convergence in the later stage of training. We also analysis the effect of episode adjusted exploration in the exploration state of DRL. Obviously, applying episode adjusted exploration to the original exploration (Ours-No EAE) in RL do help to improve the performance, either in the early or late stages of training. Similar to the instance-based decision making of human beings, this EAE policy utilises experience in episodic memory to exploit environment efficiently.

All in all, the above experiments and analyses demonstrate that our EM related improvement measures of new EM module, experience replay and exploration take effect in improving the sample efficiency respectively (the effect of loss function is proved by EMDQN). Although each improvement measure makes little difference, it makes great

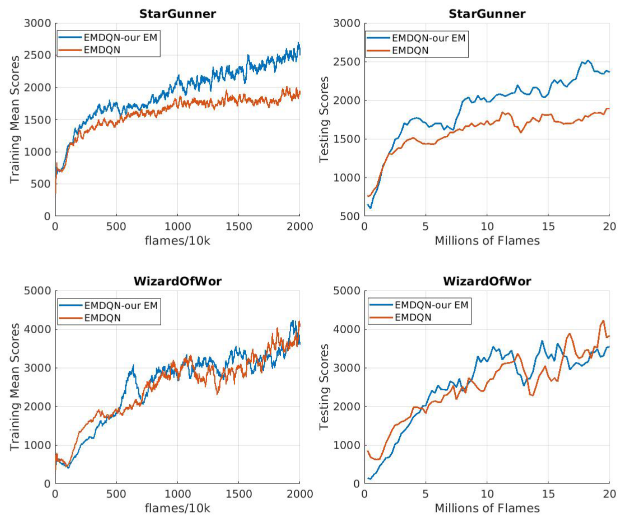

FIGURE 5. Training and testing curves of EMDQN with our new EM module (EMDQN-our EM) and EMDQN on representative games. The scores are smoothed using moving average over 4 epochs with the same random seed.

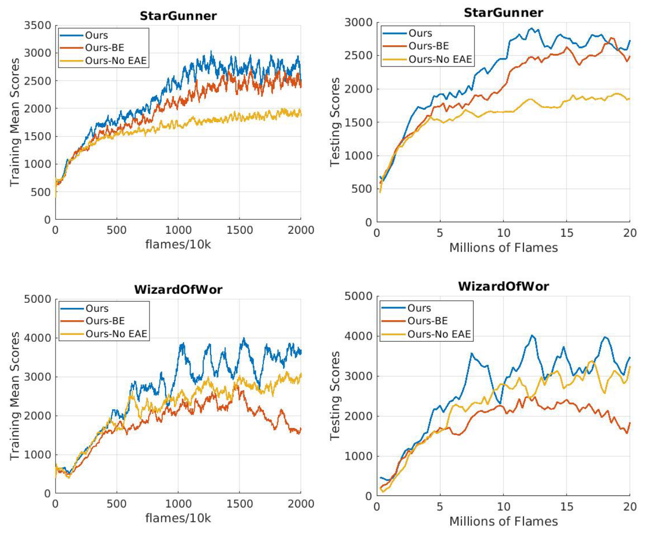

FIGURE 6. Training and testing curves of original HE-EMDQN (Ours), HE-EMDQN with best episode replay memory (Ours-BE) and HE-EMDQN without episode adjusted exploration (Ours-No EAE) on representative games. The scores are smoothed using moving average over 4 epochs with the same random seed.

difference when all of them are incorporated in our HE-EMDQN (Fig. 4). In other word, by making high efficient use of episodic memory, our algorithm do make difference in improving same efficiency for DRL.

VI. CONCLUSION

In this paper, we propose a new sample-efficient reinforcement learning architecture which introduces a new EM module and incorporates episodic thought in some key components of DRL efficiently: exploration, experience replay and loss function. Experiments in Atari games show our algorithm is significantly more sample-efficient than both DQN and EMDQN. We also do extra experiments to analysis how these EM related improvement measures influence the sample efficiency respectively. In conclusion, by making high efficient use of episodic memory, our algorithm do make difference in improving same efficiency for DRL. In the future, we will try to: 1) apply our work to continuous motion environments and practical applications in communication; 2) extend our EM module with other biologically inspired technique to update the module and evaluate new states accurately.

REFERENCES

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.

[2] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017.

[3] A. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 2017, no. 19, pp. 70–76, Jan. 2017.

[4] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*, vol. 2. New York, NY, USA: Springer, 2018.

[5] L. Lv, S. Zhang, D. Ding, and Y. Wang, "Path planning via an improved DQN-based learning policy," *IEEE Access*, vol. 7, pp. 67319–67330, 2019.

[6] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.

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

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[9] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, and T. Ewalds, "Alphastar: Mastering the real-time strategy game starcraft II," *DeepMind Blog*, p. 2, Jan. 2019.

[10] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC model-free reinforcement learning," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 881–888.

[11] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[12] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow," *Trends Cognit. Sci.*, vol. 23, no. 5, pp. 408–422, May 2019.

[13] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1–7.

[14] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2015, *arXiv:1511.06581*. [Online]. Available: http://arxiv.org/abs/1511.06581

[15] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: http://arxiv.org/abs/1511.05952

[16] J. Liu, X. Tao, and J. Lu, "QoE-oriented rate adaptation for DASH with enhanced deep Q-learning," *IEEE Access*, vol. 7, pp. 8454–8469, 2019.

[17] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2017, pp. 1–8.

[18] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare, "Safe and efficient off-policy reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1054–1062.

[19] S. W. Kennerley and M. E. Walton, "Decision making and reward in frontal cortex: Complementary evidence from neurophysiological and neuropsychological studies," *Behav. Neurosci.*, vol. 125, no. 3, p. 297, 2011.

[20] P. Andersen, R. Morris, D. Amaral, T. Bliss, and J. O'Keefe, *The Hippocampus Book*. London, U.K.: Oxford Univ. Press, 2006.

[21] D. Marr, D. Willshaw, and B. McNaughton, "Simple memory: A theory for archicortex," in *From the Retina to the Neocortex*. New York, NY, USA: Springer, 1991, pp. 59–128.

[22] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory," *Psychol. Rev.*, vol. 102, no. 3, p. 419, 1995.

[23] M. Lengyel and P. Dayan, "Hippocampal contributions to control: The third way," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 889–896.

[24] C. Blundell, B. Uria, A. Pritzel, Y. Li, A. Ruderman, J. Z. Leibo, J. Rae, D. Wierstra, and D. Hassabis, "Model-free episodic control," 2016, *arXiv:1606.04460*. [Online]. Available: http://arxiv.org/abs/1606.04460

[25] A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell, "Neural episodic control," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 2827–2836.

[26] Z. Lin, T. Zhao, G. Yang, and L. Zhang, "Episodic memory deep Q-networks," 2018, *arXiv:1805.07603*. [Online]. Available: http://arxiv.org/abs/1805.07603

[27] S. Hansen, A. Pritzel, P. Sprechmann, A. Barreto, and C. Blundell, "Fast deep reinforcement learning using online adjustments from the past," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10567–10577.

[28] S. Y. Lee, C. Sungik, and S.-Y. Chung, "Sample-efficient deep reinforcement learning via episodic backward update," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2110–2119.
[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[30] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[31] K. J. Young, R. S. Sutton, and S. Yang, "Integrating episodic memory into a reinforcement learning agent using reservoir sampling," 2018, *arXiv:1806.00540*. [Online]. Available: http://arxiv.org/abs/1806.00540
[32] D. Nishio and S. Yamane, "Random projection in neural episodic control," 2019, *arXiv:1904.01790*. [Online]. Available: http://arxiv.org/abs/1904.01790
[33] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," 2018, *arXiv:1810.02274*. [Online]. Available: http://arxiv.org/abs/1810.02274
[34] Z. Zhang, J. Chen, Z. Chen, and W. Li, "Asynchronous episodic deep deterministic policy gradient: Toward continuous control in computationally complex environments," *IEEE Trans. Cybern.*, early access, Dec. 31, 2019, doi: 10.1109/TCYB.2019.2939174.
[35] D. Nishio and S. Yamane, "Faster deep Q-learning using neural episodic control," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2018, pp. 486–491.
[36] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemp. Math.*, vol. 26, nos. 189–206, p. 1, 1984.
[37] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, "# exploration: A study of count-based exploration for deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2753–2762.
[38] J. O'Neill, B. Pleydell-Bouverie, D. Dupret, and J. Csicsvari, "Play it again: Reactivation of waking experience and memory," *Trends Neurosci.*, vol. 33, no. 5, pp. 220–229, May 2010.
[39] R. A. Adcock, A. Thangavel, S. Whitfield-Gabrieli, B. Knutson, and J. D. E. Gabrieli, "Reward-motivated learning: Mesolimbic activation precedes memory formation," *Neuron*, vol. 50, no. 3, pp. 507–517, May 2006.
[40] A. M. Do, A. V. Rupert, and G. Wolford, "Evaluations of pleasurable experiences: The peak-end rule," *Psychonomic Bull. Rev.*, vol. 15, no. 1, pp. 96–98, Feb. 2008.
[41] A. R. Mahmood, H. P. van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3014–3022.
[42] C. M. A. Pennartz, R. Ito, P. F. M. J. Verschure, F. P. Battaglia, and T. W. Robbins, "The hippocampal–striatal axis in learning, prediction and goal-directed behavior," *Trends Neurosci.*, vol. 34, no. 10, pp. 548–559, Oct. 2011.
[43] F. S. He, Y. Liu, A. G. Schwing, and J. Peng, "Learning to play in a day: Faster deep reinforcement learning by optimality tightening," 2016, *arXiv:1611.01606*. [Online]. Available: https://arxiv.org/abs/1611.01606
[44] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: http://arxiv.org/abs/1606.01540

DUJIA YANG received the B.S. degree in microelectronics from the Hefei University of Technology, Hefei, China, in 2013. He is currently pursuing the Ph.D. degree with the Wireless Information Network Laboratory, University of Science and Technology of China, Hefei. His current research interests are in the areas of wireless communications, big data analysis, and deep learning.

XIAOWEI QIN received the B.S. and Ph.D. degrees from the Department of Electrical Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2000 and 2008, respectively. Since 2014, he has been a member of staff with the Key Laboratory of Wireless Optical Communications of Chinese Academy of Sciences, USTC. His research interests include optimization theory, service modeling in future heterogeneous networks, and big data in mobile communication networks.

XIAODONG XU received the B.S. and Ph.D. degrees from the Department of Electrical Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2000 and 2007, respectively. He is currently a member of staff with the Key Laboratory of Wireless Optical Communications of Chinese Academy of Sciences, USTC. His current research interests include communication signal processing, blind signal processing, and machine learning in wireless communication.

CHENSHENG LI received the B.S. degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, in 2015, where he is currently pursuing the Ph.D. degree with the Wireless Information Network Laboratory. His research interests include deep learning in non Euclidean domain, graph neutral networks, and machine learning in mobile networks.

GUO WEI received the B.S. degree in electronic engineering from the University of Science and Technology of China (USTC), Hefei, China, in 1983, and the M.S. and Ph.D. degrees in electronic engineering from the Chinese Academy of Sciences, Beijing, China, in 1986 and 1991, respectively. He is currently a Professor with the School of Information Science and Techno logy, USTC. His current research interests include wireless and mobile communications, wireless multimedia communications, ultra wideband communication systems, and wireless information networks.