Self-Paced Prioritized Curriculum Learning With Coverage Penalty in Deep Reinforcement Learning

Zhipeng Ren, Daoyi Dong, *Senior Member, IEEE*, Huaxiong Li, *Member, IEEE*, and Chunlin Chen, *Member, IEEE*

***Abstract*— In this paper, a new training paradigm is proposed for deep reinforcement learning using self-paced prioritized curriculum learning with coverage penalty. The proposed deep curriculum reinforcement learning (DCRL) takes the most advantage of experience replay by adaptively selecting appropriate transitions from replay memory based on the complexity of each transition. The criteria of complexity in DCRL consist of self-paced priority as well as coverage penalty. The self-paced priority reflects the relationship between the temporal-difference error and the difficulty of the current curriculum for sample efficiency. The coverage penalty is taken into account for sample diversity. With comparison to deep Q network (DQN) and prioritized experience replay (PER) methods, the DCRL algorithm is evaluated on Atari 2600 games, and the experimental results show that DCRL outperforms DQN and PER on most of these games. More results further show that the proposed curriculum training paradigm of DCRL is also applicable and effective for other memory-based deep reinforcement learning approaches, such as double DQN and dueling network. All the experimental results demonstrate that DCRL can achieve improved training efficiency and robustness for deep reinforcement learning.**

***Index Terms*— Coverage penalty, curriculum learning, deep reinforcement learning, self-paced priority.**

I. Introduction

REINFORCEMENT learning helps an agent to learn an optimal policy through the interaction with the environment, and this goal can be achieved by maximizing the future return, which is similar to human behaviors [1], [2]. However, the traditional approaches to solve reinforcement learning problems cannot work well in the high-dimensional real world, and this issue hinders the wide applications of reinforcement learning. Ever since deep learning has made the state-of-the-art achievements in a variety of fields [3]–[8], it provides an alternative way that we can incorporate deep learning with reinforcement learning to obtain effective and flexible representations for the learning agent.

There have been many breakthroughs of applying deep neural network to reinforcement learning in various domains [9]–[13], and one of the most successful frameworks is the deep Q network (DQN) that consists of a convolutional neural network to approximate the value function [14], [15]. DQN makes it possible for a reinforcement learning agent to optimize the policy via online and end-to-end learning. In order to stabilize the learning process, the experience replay mechanism is employed to break the correlations between consecutive samples [16], where uniform sampling is used for the design of experience replay. However, the uniform sampling-based experience replay in the DQN neglects the fact that samples play a diverse role in the learning process and some crucial samples should be selected more frequently than others. On the other hand, prioritized sweeping is one of the most common model-based methods to enhance efficiency with less data and time [17]–[19]. Following this idea, prioritized experience replay (PER) provides a priority-based sample strategy to take full advantage of significant transitions (samples) [20]. However, a critical issue is that deep neural network is sensitive to the change of parameters. Too large change of parameters has a negative effect on the convergence of the training process, which will lead to the adverse volatile consequences in PER [21].

In order to improve the efficiency and robustness of the learning process, a new training paradigm is designed by combining curriculum learning with deep reinforcement learning, and an integrated deep curriculum reinforcement learning (DCRL) algorithm is presented. Curriculum learning is motivated by a student learning strategy in which students get benefit from a meaningful ordered curriculum [22]. Students usually learn many easier courses before they start to learn more complex courses. The combination of curriculum learning and supervised learning has achieved many remarkable achievements in various areas [23]–[25]. In the context of deep reinforcement learning, curriculum learning allows the agent to select different transitions in different learning stages. Transitions are sampled from easiness to hardness regarding different complexities, which makes the training process similar to the education process of students. In addition, to evaluate the complexity of each transition, two criteria

Manuscript received April 28, 2017; revised September 10, 2017, November 26, 2017, and December 16, 2017; accepted January 3, 2018. Date of publication February 1, 2018; date of current version May 15, 2018. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFD0702100, in part by the National Natural Science Foundation of China under Grant 71671086, Grant 61432008, and Grant 71732003, and in part by the Australian Research Council under Grant DP130101658. *(Corresponding authors: Huaxiong Li; Chunlin Chen.)*

Z. Ren, H. Li, and C. Chen are with the Department of Control and Systems Engineering, School of Management and Engineering, Nanjing University, Nanjing 210093, China (e-mail: 2861916772@qq.com; huaxiongli@nju.edu.cn; clchen@nju.edu.cn).

D. Dong is with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2600, Australia (e-mail: daoyidong@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2790981

2162-237X © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

(i.e., self-paced prioritized criterion and coverage penalty criterion) are designed to guarantee both of the sample efficiency and diversity.

Self-paced prioritized criterion implies that this standard is developed by the agent without any requirement of prior knowledge in accordance with self-paced learning [26]–[30]. The relationship between the temporal-difference (TD) error and the curriculum factor is taken into consideration to emphasize the efficiency importance of each transition. Coverage penalty criterion draws on the experience of core notion in [31]–[33] to reduce sample frequency of transitions that have already been used for too many times to guarantee the sample diversity. Generally speaking, the incorporation of self-paced prioritized criterion and coverage penalty criterion reflects both sample efficiency and sample diversity to improve robustness as well as generalization of DCRL.

To test the proposed DCRL algorithm, experimental results on Arcade Learning Environment (ALE) platform are provided with comparison to DQN and PER. The curriculum training method is further combined with memory-based deep reinforcement learning algorithms (e.g., double DQN [34] and dueling network [35]) to verify the generality and practicability of DCRL.

The rest of this paper is organized as follows. Section II introduces several basic concepts about reinforcement learning, the constitution of DQN, as well as the theory of curriculum learning. In Section III, the DCRL algorithm is presented in detail. Experimental results are presented and discussed in Section IV. Concluding remarks are drawn in Section V.

II. Background

In this section, some important concepts in reinforcement learning, DQN, and curriculum learning are briefly introduced in turn.

A. Reinforcement Learning

The formalization of reinforcement learning is based on the model of *Markov decision process*, which consists of a tuple of $\langle S, A, P, R\rangle$ [1], where S is the state space, A is the action space, $P: S\times A\times S \to [0,1]$ is the state transition probability, and R: $S\times A \to \mathbb{R}$ is the reward function.

In the interaction with the environment, the agent forms the state representation $s_t \in S$ at each time step $t \in [0, T]$, where T is the terminal time. It chooses an action $a_t \in A$ according to the policy function $a_t = \pi(s_t)$. Policy π is a mapping from state space S to action space A. After executing action a_t, the agent transits to next state s_{t+1} and gets a scalar reward signal r_t. Thus, we obtain a transition $x_t : (s_t, a_t, r_t, s_{t+1})$ during the time step t. The goal of reinforcement learning is to choose an optimal action a^* at each state s_t so as to maximize the cumulative discounted future rewards of return $R_t = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$, where $\gamma \in [0,1]$ is the discount factor to balance the importance of current rewards and future rewards.

To solve reinforcement learning problems, the optimal state-action value function $Q^*(s,a)$ can be defined as the maximum expectation of return R_t as follows:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\{R_t | s_t = s, a_t = a\}. \tag{1}$$

According to *Bellman Equation*, (1) can be rewritten as

$$Q^*(s,a) = \mathbb{E}\left\{r + \gamma \max_{a'} Q^*(s',a') | s, a\right\} \tag{2}$$

where s' is the next consecutive state and a' is a legal action at s'. Q-learning [36] is one of the widely used TD methods to estimate $Q^*(s,a)$, whose one step updating rule is as follows:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left[r + \gamma \max_{a'} Q_t(s',a') - Q_t(s,a)\right] \tag{3}$$

where $Q_t(s,a)$ denotes the state-action value function at time step t and $\alpha \in [0,1]$ is the learning rate. Under certain conditions [1], the convergence of Q-learning can be guaranteed, i.e., $t \to \infty$, $Q_t(s,a) \to Q^*(s,a)$.

B. Deep Q Network

A lookup table storing all state-action values $Q(s,a)$ is the most usual practice in simple environments. However, when the environment is high-dimensional or the state-action space is continuous, it is impossible to use a tabular form to represent all state-action pairs (s,a). To deal with this problem, a function with parameters θ is usually used for approximating $Q(s,a)$, i.e., $Q(s,a;\theta) \approx Q(s,a)$ [37].

Since deep learning has made major advances in the fields of image recognition [3]–[5], speech recognition [6], [38], [39], as well as natural language processing [7], [40], [41], there are also many trials to apply deep learning in reinforcement learning as a function approximator [9]–[12]. Deep Q network (DQN) is one of the most successful attempts to integrate deep learning with reinforcement learning [14], [15]. Its architecture consists of a convolutional neural network where the input is raw high-dimensional screen images and the output is individual action value. DQN achieves the state-of-the-art results in the Arcade game domain only by end-to-end training without additional prior knowledge. In DQN, in order to use the gradient descent method to update the parameters of the neural network, "right value" $y(s,a)$ of state-action value $Q(s,a)$, which is similar to the true label or value in supervised learning, is bootstrap estimated from the maximum of next state-action value $Q(s',a')$

$$y(s,a) = r + \gamma \max_{a'} Q(s',a';\theta^-) \tag{4}$$

where θ^- denotes the parameters of the target network which is fixed during the computation of $y(s,a)$ and is updated after some training steps. Hence, we obtain TD error δ measured by the deviation between $y(s,a)$ and $Q(s,a)$ as

$$\delta = y(s,a) - Q(s,a) = r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta). \tag{5}$$

Accordingly, the loss function $\text{Loss}(\theta; Q, y)$ that we optimize is given as

$$\text{Loss}(\theta; Q, y) = \frac{1}{2}\left(r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta)\right)^2. \tag{6}$$

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

Differentiate the loss function Loss(θ; Q, y) with respect to the parameters θ of DQN, we have the following gradient:

$$\nabla_\theta \text{Loss} = \left[r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right] \nabla_\theta Q(s, a; \theta). \quad (7)$$

When applying a nonlinear function approximator to an off-policy method like Q-learning, it is supposed to diverge or oscillate due to the strong correlations between consecutive transitions. That is to say, even a small change of the decisions can give rise to severe uncertain variation. Hence, one crucial element leading to the success of DQN is the usage of experience replay [16]. For a DQN agent with experience replay, its replay memory stores transitions that has been experienced for a period of time. Then, the algorithm samples mini-batch data uniformly to update the network parameters. Experience replay not only enhances the efficiency of learning, but also has a great influence on the stability of deep neural network. It allows transitions to be reused for many times to increase sample efficiency, especially for the training on GPU. In addition, experience replay alleviates the violation of independent and identically distributed assumption of training data, which plays an important role in the convergence of learning algorithms.

C. Curriculum Learning

In academic education, only when the students master arithmetic, can they start to study algebra, geometry, and other more difficult courses. If the courses are in a wrong order, the learning process would be more challenging for students. Generally speaking, students learn simpler concepts first, which makes it easy to learn more complex ones later. It is clear that students can benefit from courses organized in a meaningful order.

Motivated by this phenomenon, Bengio *et al.* [22] introduced curriculum learning that applies this kind of human learning strategy in machine learning. Specifically, the learning algorithm divides the training data into different parts according to their learning complexities instead of choosing data randomly from the training set. First of all, the parameters of the learning algorithm are updated only by partial training data containing simple concepts. More and more difficult samples are encountered with the continuation of learning. Eventually, all of the training data are used to fine-tune the parameters.

Curriculum learning has been applied in many supervised and semisupervised learning fields. Curriculum learning for speech recognition in a noisy environment changes the signal-to-noise ratio of audio data stage by stage and successfully improves the noise robustness of recurrent neural network (RNN) [23]. The combination of multimodel and curriculum learning by investigating the difficulty of classifying every unlabeled image in semisupervised learning makes a leap in final results [24]. In addition, curriculum learning for the RNN language model helps the algorithm adapt to the data [25]. All these applications demonstrate the success of curriculum learning for the speedup of the learning process, especially for deep neural networks. In this paper, the curriculum learning scheme is well designed with self-paced prioritized memory and coverage penalty for deep reinforcement learning to achieve improved performance regarding efficiency and robustness.

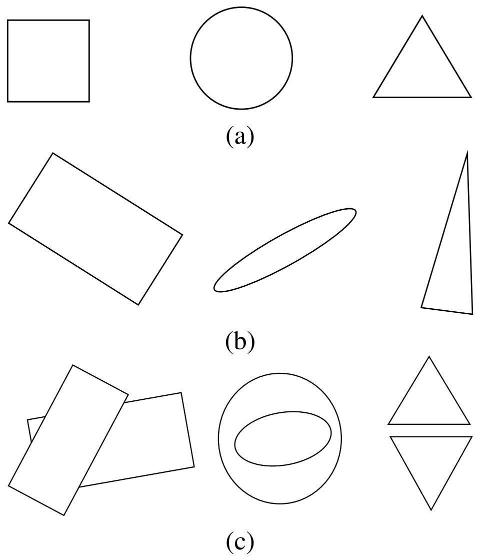


Fig. 1. Examples of shape recognition with different complexities. (a) Square, circle, and equilateral triangle which are very simple to recognize. (b) Some rotational shapes that are more difficult to recognize. (c) Most difficult shapes to recognize.

III. Deep Curriculum Reinforcement Learning

In this section, the framework of DCRL is first introduced. To integrate curriculum learning with deep reinforcement learning, the agent learns the courses of different complexities at different training stages. Then, both self-paced prioritized function and coverage penalty function are designed as criteria to assess the complexity of each transition. Finally, the implementation of the integrated DCRL algorithm is presented.

A. Framework of DCRL

In our learning career, we usually start with easier courses, and then learn more complex courses. It seems that the easy courses lay the foundation for future complex ones. Therefore, our courses become an ordered curriculum sequence according to their complexities. Similarly, considering a given set E containing element e, we can define a curriculum sequence as Definition 1.

Definition 1 (Curriculum Sequence): Considering a given ordered set $E = \{e_m\}_{m=1}^N$, $\forall e_i, e_j \in E$, if $i < j$, then e_i is easier than e_j.

For example, let us consider a shape recognition task, as shown in Fig. 1. We can get a curriculum sequence by dividing the training set into three different parts according to their recognition difficulty as groups [Fig. 1(a)–(c)]. The recognition agent learns to recognize very simple parts correctly at first [Fig. 1(a)]. Then, some difficult parts, such as some rotational, scaled, and inverse shapes, are learned [Fig. 1(b)]. The most difficult parts are used to improve the final performance of the algorithm [Fig. 1(c)].

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

As for deep reinforcement learning, if we draw an analogy between the agent that learns through transitions and the students who learn through courses, sampling from the replay memory is similar with taking some courses from the prescriptive curriculum. Thus, the main idea of DCRL can be described as follows. The agent selects simpler transitions from the replay memory at the beginning of the learning phase and increases the proportion of complex transitions along with the training process. In order to organize the replay memory as a curriculum sequence, transitions should be sorted in a meaningful order according to some criteria determined by a complexity index function.

Definition 2 (Complexity Index Function): Assume a function $\mathrm{CI}(s) \to \mathbb{R}$ defines the complexity index of the sample s in sample set S. For any given two samples s_i and s_j, if $\mathrm{CI}(s_i) < \mathrm{CI}(s_j)$, we say that sample s_i is easier to learn than sample s_j.

With a complexity index function, the training process of DCRL can be divided into two phases, i.e., a curriculum evaluation phase and a parameter updating phase. In the curriculum evaluation phase, the agent selects appropriate transitions based on the curriculum evaluation function. In the parameter updating phase, the network parameters are updated using the stochastic gradient descent method. The framework of DCRL involves the following optimization problems with constraints:

$$\begin{cases}\min_W & f(W,\theta,\lambda)\\ \min_\theta & \sum_{i=1}^{N} w_i\,\mathrm{Loss}(\theta;\,Q_i,y_i)\end{cases}$$
$$\text{s.t. } w_i \in [0,1],\quad \sum_{i=1}^{N} w_i = K \tag{8}$$

where $f(W,\theta,\lambda)$ is the curriculum evaluation function which determines a learning scheme for the agent to select appropriate transitions, $W = [w_1, w_2, \ldots, w_N]$ denotes the weight variables reflecting whether the transition is selected or not, and λ is a variable called curriculum factor to control the difficulty of the selected transitions. $M = \{(Q_i(s,a), y_i(s,a))_{i=1}^{N}\}$ corresponds to the transitions in replay memory, $\mathrm{Loss}(\theta;\,Q_i,y_i)$ is the loss function defined in (6), and K is the size of the sample batch.

In order to solve the above-mentioned problems, we make an assumption that the parameters (W and θ) to be optimized are disjoint, so that one of the two parameters is optimized, while the other is kept fixed in each iteration. These two optimization problems can be eligibly implemented by existing off-the-shelf optimization approaches. Then, we increase the curriculum factor λ so that the agent will be associated with more complex transitions in the next iteration. In that way, we can obtain an optimal model parameter θ^*.

B. Self-Paced Prioritized Memory

Within the framework of DCRL, two critical issues are how to evaluate the complexity of each transition and how to design the explicit criteria. For students, suitable courses in different learning phases can be selected with the guide of subsistent knowledge or experienced teachers. However, for a reinforcement learning agent in an undiscovered environment, no prior knowledge is available for the evaluation of transitions. Moreover, artificially settled standards of transition complexity may be inappropriate for the constantly changing agent while learning. Difficult concepts at the beginning of learning can be understood by the later expert agent effortlessly. Hence, the agent should select transitions from replay memory by itself, which is called self-paced learning [26]–[28]. Self-paced learning implies that the curriculum sequence is determined by the agent without any requirement of prior knowledge. The criterion is changing dynamically in the training process to reflect the gradually obtained information.

On the other hand, the transitions stored in the replay memory have a different importance for the learning agent, and sampling each transition at the same frequency may unavoidably lead to inefficient usage of meaningful transitions. Prioritized sweeping provides a good solution in model-based methods to handle the above uniform sampling problem [17]–[19]. Existing results show that PER can be incorporated with modelfree methods, such as DQN [20]. The core idea of PER is that the priority of transition is measured by the magnitude of TD errors. A couple of variants, including proportional form and rank-based form, are put forward to improve the sample efficiency.

Sampling transitions with a large magnitude of TD error δ may cause adverse consequences due to the following reasons.

1) If the environment is fickle and changeful, the reward is noisy. Noise in the environment hides the real feedback signal and results in unreliable reward.
2) The defined y value is bootstrap estimated from the maximum value of next state-action value $Q(s', a'; \theta^-)$ with parameters in the target network. If the parameters are in an inappropriate scale, it may cause fallibility.
3) The first-order gradient is only reliable in a small local area in the context of deep neural network. Transitions with huge magnitudes of TD errors need smaller step size for the sake of following the curvature of objective function [21].

In order to alleviate the risk of selecting unreliable data, the agent should focus on appropriate transitions in each value iteration. Too easy samples may have little help for improving the current learning ability, while some other samples may be too difficult for the agent to understand. So we define a self-paced prioritized function in DCRL to select appropriate transitions.

Definition 3 (Self-Paced Prioritized Function): Suppose that δ is the TD error of a transition and λ is the curriculum factor that indicates the age of model. $\mathrm{SP}(\delta, \lambda)$ is defined as the self-paced prioritized function that satisfies the following conditions.

1) $\mathrm{SP}(\delta, \lambda) \to [0, 1]$.
2) If $|\delta| > \lambda$, $\mathrm{SP}(\delta, \lambda)$ is a monotonically decreasing function of $|\delta|$. If $|\delta| < \lambda$, $\mathrm{SP}(\delta, \lambda)$ is a monotonically increasing function of $|\delta|$. Otherwise, $\mathrm{SP}(\delta, \lambda)$ is a global maximum.

As a result, each transition stored in a replay memory can be mapped into a scalar number according to a predefined

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

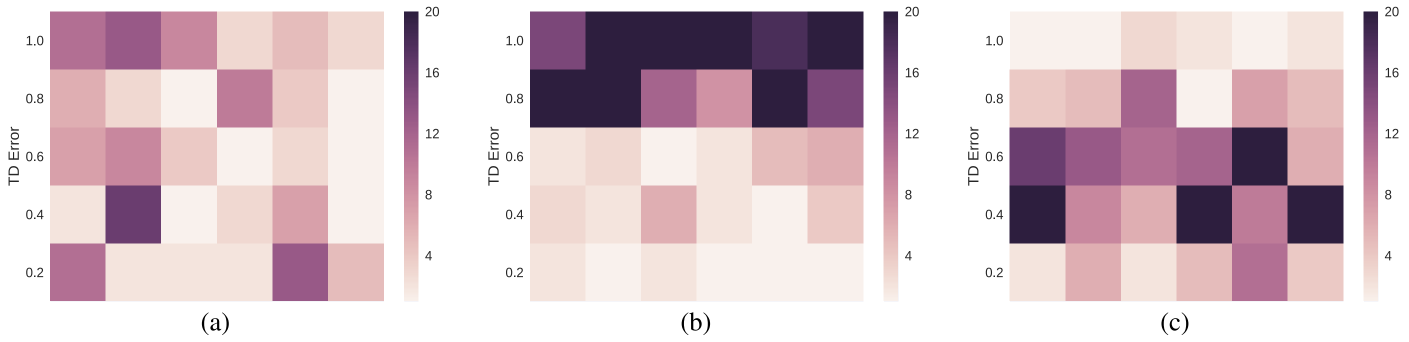


Fig. 2. Heat maps of the coverage number of the three different algorithms. (a) DQN. (b) PER. (c) DCRL. The depth of color in each image represents the size of coverage number.

self-paced prioritized function. We can select transitions that play an effective role in sample efficiency more frequently.

C. Coverage Penalty

In experience replay, when some unnecessary transitions are reused for too many times, unacceptable overtraining situation may happen. Besides, it is clear that the limited size of replay memory makes overtraining even worse [42]. Overtraining reveals a traditional problem in reinforcement learning called the exploration–exploitation tradeoff [43]–[45]. Sufficient exploring in the state-action space is important for preventing the algorithm getting stuck in a bad local minimum, while exploiting the current policy can help the algorithm converge as fast as possible. Accordingly, if the agent is only restricted to some specific transitions, it cannot see the wood for the trees. By contrast, there is no need to manage all aspects of the environment. For the sake of generalization of reinforcement learning agent, we take into account the sample diversity.

Similar situations called overtranslation and undertranslation also happen in machine translation, which implies that some words are translated for multiple times and some words are mistakenly untranslated [46]. To deal with this problem, a coverage mechanism integrated in neural machine translation is introduced in [33]. A new variable called coverage vector is appended into the translation model to keep a record of the attention history in the encoding stage. When the attention model plays a role in the decoding stage, it can help correct the future attention and improve the overall alignment between source sentences and target sentences.

Analogously, we add an additional coverage vector $[\mathrm{cn}_1, \mathrm{cn}_2, \ldots, \mathrm{cn}_N]$ to the replay memory for recording the sample times of each transition. If a transition x_i is sampled to update the model parameters in an iteration, then the corresponding variable coverage number cn_i is updated by $\mathrm{cn}_i = \mathrm{cn}_i + 1$. The more times the transition has been selected, the less the probability of being selected in next iteration is. Thus, a coverage penalty function CP(cn) is defined to accomplish this process.

Definition 4 (Coverage Penalty Function): Suppose that cn is a coverage number keeping track of the sample times. CP(cn) is defined as a coverage penalty function and satisfies the following requirements.

1) $\mathrm{CP}(\mathrm{cn}) \to (0, 1]$.
2) CP(cn) is a monotonically decreasing function of cn.

We choose some coverage numbers from replay memory randomly and divide them into different groups according to their TD errors in a heat map. The heat maps of the three different algorithms (DQN, PER, and DCRL) are displayed in Fig. 2, where the color of the grid reflects the size of the coverage number. These maps disclose the coverage number of each transition. It is clear that the coverage number in Fig. 2(a) is almost the same. But a serious imbalance phenomenon appears in Fig. 2(b), which does harm to the exploration of the agent. DCRL alleviates the imbalance with the help of coverage penalty, as shown in Fig. 2(c) ($\lambda = 0.5$).

D. Implementation

Based on the definitions of the self-paced prioritized function $\mathrm{SP}(\delta, \lambda)$ and the coverage penalty function CP(cn), we can evaluate the complexity of each transition to guarantee the sample efficiency and sample diversity. The complexity index function of transition x_i is depicted as follows:

$$\mathrm{CI}(x_i) = \mathrm{SP}(\delta_i, \lambda) + \eta\, \mathrm{CP}(\mathrm{cn}_i). \tag{9}$$

Thus, the curriculum evaluation function $f(W, \theta, \lambda)$ in a DCRL framework can be specified as

$$f(W, \theta, \lambda) = \sum_{i=1}^{N} \frac{\ln(w_i + 1)}{\mathrm{SP}(\delta_i, \lambda) + \eta\, \mathrm{CP}(\mathrm{cn}_i)}. \tag{10}$$

In (10) and (11), η is an extra factor weighting the pros and cons of priority and penalty.

Here, we give two specific functions of $\mathrm{SP}(\delta, \lambda)$ and CP(cn) that satisfy Definitions 3 and 4, respectively, for the implementation of the proposed DCRL algorithm.

Self-Paced Prioritized Function SP(δ, λ):

$$\mathrm{SP}(\delta, \lambda) = \begin{cases} \exp(|\delta| - \lambda) & |\delta| \le \lambda \\ \dfrac{1}{\log(1-\lambda)} \log(|\delta| - 2\lambda + 1) & \lambda < |\delta| < 2\lambda \\ 0 & |\delta| \ge 2\lambda. \end{cases} \tag{11}$$

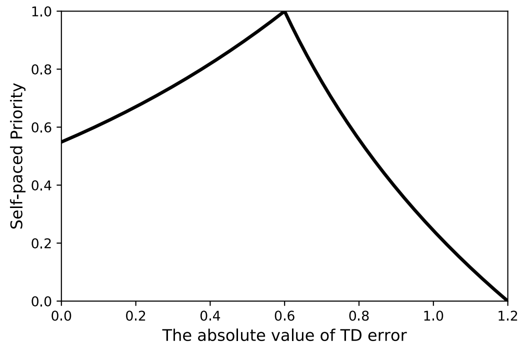
Coverage Penalty Function CP(cn):

$$\mathrm{CP}(\mathrm{cn}) = \exp\left(-\frac{\mathrm{cn}^2}{10}\right). \tag{12}$$

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 DCRL Algorithm

```
Input: Replay Size N, Mini Batch K, Curriculum Factor $\lambda$, Step Size $\mu$ and Balanced Weight $\eta$
    Additional Coverage Number Vector $cn = [cn_1, cn_2, \dots, cn_N]$, Additional Complexity Index Vector $ci = [ci_1, ci_2, \dots, ci_N]$
    Self-paced Prioritized Function $SP(\delta, \lambda)$, Coverage Penalty Function $CP(cn)$
1:  Initialize constant $N, K, \lambda, \mu, \eta$ and Vector $cn = \vec{0}$, $ci = \vec{0}$.
2:  Observe $s_0$ and choose $a_0 \sim \pi(s_0)$
3:  for $t = 1 \to T$ do
4:     Observe $r_t, s_t$
5:     Store transition $(s_{t-1}, a_{t-1}, r_{t-1}, s_t)$ in replay memory with $ci_{t-1} = \max_n ci_n$
6:     for $j = 1 \to K$ do
7:        Sample transition $j \sim P(j) = ci_j / \sum_n ci_n$
8:        Compute TD error $\delta_j = r_j + \gamma \max_{a_{j+1}} Q_{target}(s_{j+1}, a_{j+1}) - Q(s_j, a_j)$
9:        Compute $sp_j = SP(\delta_j, \lambda)$
10:       Update $cn_j$ by $cn_j = cn_j + 1$
11:       Compute $cp_j = CP(cn_j)$
12:       Update $ci_j$ by $ci_j = sp_j + \eta cp_j$
13:    end for
14:    Update $\lambda = \lambda + \mu$
15:    Update weights $\theta$ by stochastic gradient descent.
16:    Copy weights into target network $\theta_{target} \leftarrow \theta$.
17:    Choose action $a_t \sim \pi(s_t)$
18: end for
```


(a)

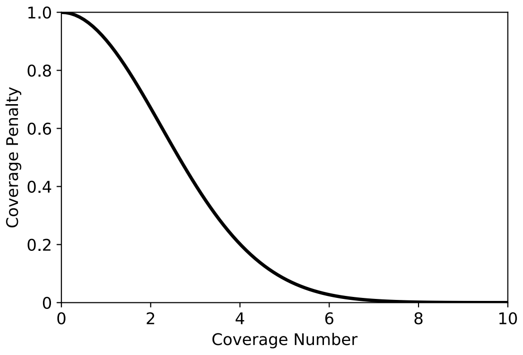

(b)

Fig. 3. Sketch of the designed SP and CP functions. (a) Self-paced prioritized function with $\lambda = 0.6$. (b) Coverage penalty function.

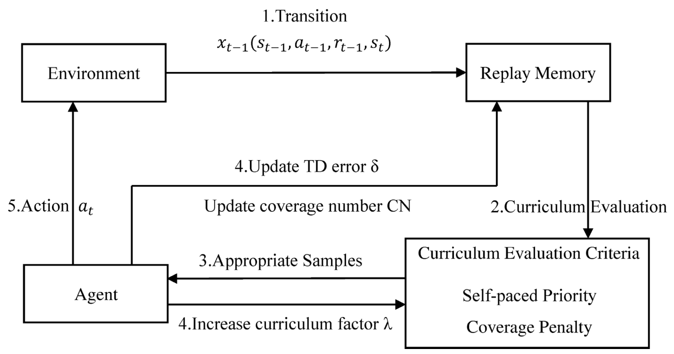


Fig. 4. Procedure of DCRL. 1: agent stores a transition in replay memory. 2. agent scores the complexity of each transition according to curriculum evaluation criteria. 3: agent selects some appropriate transitions to update the network parameters. 4: agent updates the TD error as well as coverage number and increases curriculum factor. 5: agent chooses an action.

An integrated DCRL algorithm is shown in Algorithm 1. Fig. 3 presents a sketch of the designed SP and CP functions. Fig. 4 gives an explanation for the procedure of DCRL. During each value iteration, the agent encounters transition x_{t-1} and stores it in replay memory with coverage number $\mathrm{cn}_{t-1} = 0$ and complexity index $\mathrm{ci}_{t-1} = \max_n \mathrm{ci}_n$ at time step $t - 1$, which ensures that each transition is sampled at least once. Transitions are sampled with probability proportional to their complexity index to compose mini-batch data. Then, new complexity indices are updated according to (10). Finally, we increases the curriculum factor λ so that more difficult transitions are sampled in the next value iteration. This procedure is carried out iteratively until the algorithm converges.

IV. Experiments

To test the proposed DCRL algorithm, experiments are carried out on Arcade games with comparison to two benchmark algorithms (DQN and PER).

A. Setup

The experiments are carried out on the platform ALE to play Atari 2600 games [47]. As shown in Fig. 5, four different

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

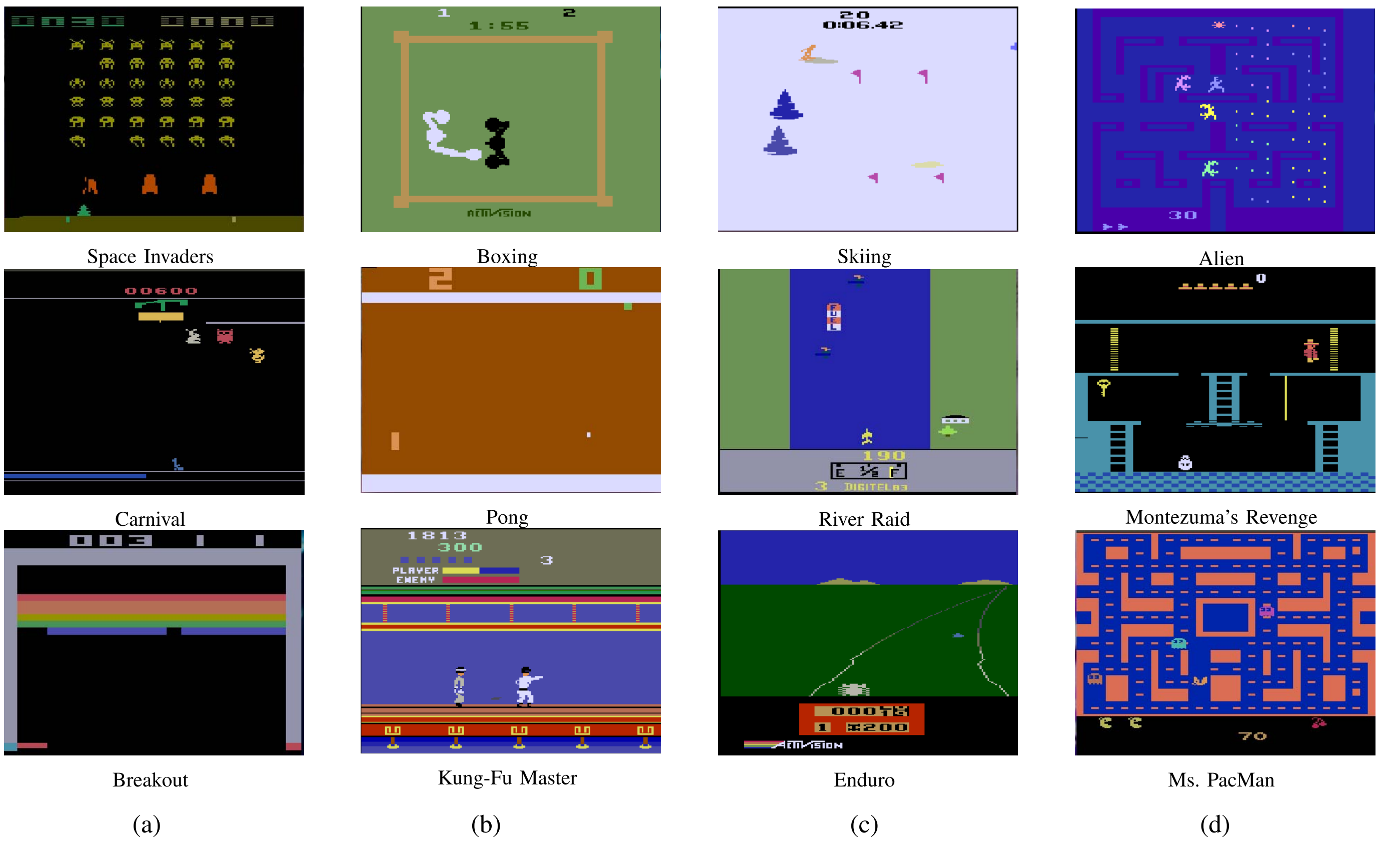


Fig. 5. Four different classes of typical games used in the experiments. (a) Three shooter games: Space Invaders, Carnival, and Breakout. (b) Three adversarial games: Boxing, Pong, and Kung-Fu Master. (c) Three racing games: Skiing, River Raid, and Enduro. (d) Three strategy games: Alien, Montezuma's Revenge, and Ms. PacMan.

classes of games are selected as testing games, including the shooter games, the adversarial games, the racing games, and the strategy games. The shooter games focus on the actions of the agent using some weapons to destroy obstacles [Fig. 5(a)]. The agent in an adversarial game aims to beat its opponent [Fig. 5(b)]. The agent in a racing competition tries to reach the finishing post as soon as possible [Fig. 5(c)]. The agent must make a long-term plan in a strategy game [Fig. 5(d)]. All the experiments are carried out on a computer of ThinkStation P700 with 24xCPU at 2.40 GHz, Nvidia Tesla K20c, Ubuntu 14.04.5 LTS, Python.

Besides DCRL, two baseline algorithms of DQN [15] and PER [20] are also tested for comparison. The sampling methods in DQN and PER can be regarded as special cases of DCRL. DQN treats each transition equally no matter how much the agent can be improved by learning. However, PER selects transitions that are the most complex ones in replay memory to update the parameters.

The architecture of deep neural network and the values of some hyperparameters are the same as those in [15] and [20]. However, we make a minor adjustment of hyperparameters due to high computational requirements and high time cost. The time consumption of training for 50 million frames is not feasible due to hardware and software limitation. Hence, we train for 5 million frames. The agent is warmed up by initial 50 000 frames without training. The hyperparameter β used for importance sampling in PER is no longer linearly annealing but fixed as a constant. We use "sum tree" data structure to store the complexity index of each transition. It is a special binary heap with the property that the value of a parent node is the sum of all values of its children. In addition, we also add some new hyperparameters, including the balanced weight η, the curriculum factor λ, and the step size μ. The values of all additional hyperparameters are selected by performing a grid search on the game Breakout. Please refer to Tables I and II for more details, where Table II only shows the partial results of grid search.

TABLE I
HYPERPARAMETERS ADJUSTMENTS IN NUMERICAL EXPERIMENTS

Altered Hyper-parameters		
Hyper-parameter	Original Value	Altered Value
Training Frames	50 Millions	5 Millions
Importance Sample Variable β	$0.4 \rightarrow 1.0$	0.4

New Additional Hyper-parameters	
Hyper-parameter	Value
Weighted factor η	0.3
Curriculum Factor λ	0.4
Step Size per Epoch μ	0.01

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

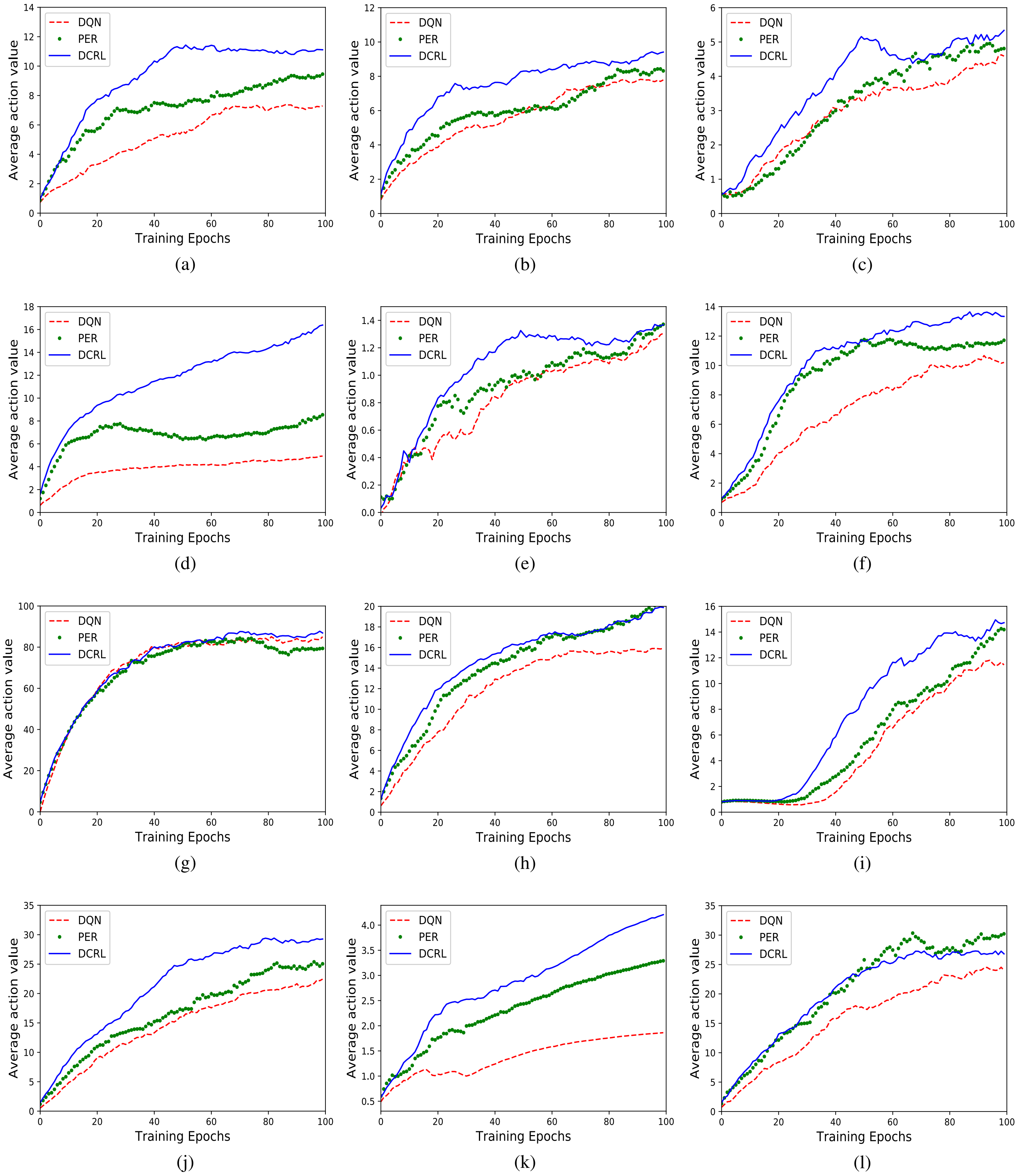
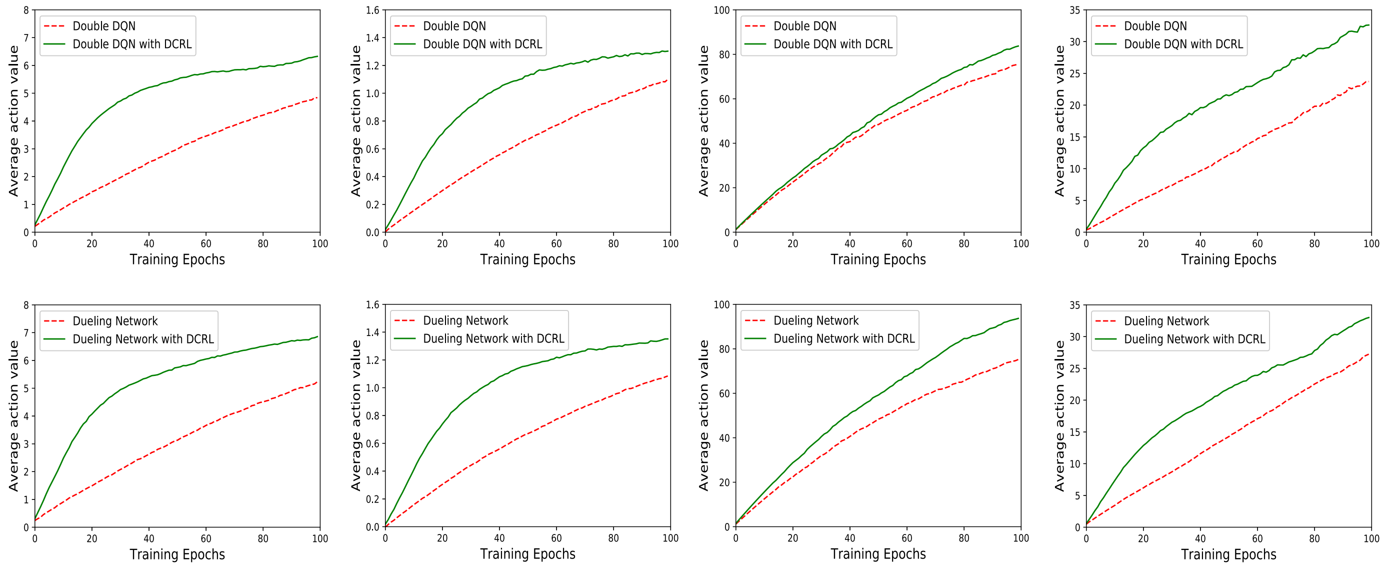


Fig. 6. Performance of DCRL with comparison to DRN and PER regarding the average Q value. (a) Space invaders. (b) Carnival. (c) Breakout. (d) Boxing. (e) Pong. (f) Kung-Fu master. (g) Skiing. (h) River raid. (i) Enduro. (j) Alien. (k) Montezuma's revenge. (l) MS. PacMan.

B. Experimental Results

All the 12 games of four classes, as shown in Fig. 4, are used to test the performance of DCRL, DQN, and PER regarding the average reward per episode and the convergence rate. The learning agents receive rewards from the simulation platform ALE, and we set 10 000 frames as the testing frames. We obtain the average reward per testing episode [47] of DCRL, DQN, and PER for all the 12 games after the end

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.



(a) (b) (c) (d)

Fig. 7. Performance of DCRL with comparison to double DQN and dueling network regarding the average Q value. (a) Breakout. (b) Pong. (c) Skiing. (d) Alien.

TABLE II
GRID SEARCH OF HYPERPARAMETERS IN DCRL

η	λ	μ	Evaluated Q
0.2	0.2	0.008	5.21
0.3	0.3	0.01	5.17
0.2	0.2	0.008	5.10
0.4	0.2	0.01	5.52
0.3	0.4	0.01	5.78
0.3	0.2	0.01	4.98
0.4	0.4	0.01	4.34
0.2	0.4	0.008	5.23

of training. The results are shown as in Table III regarding the average reward with standard deviation. The results show that DCRL outperforms DQN and PER on almost all the tested games. We divide the training phase into 100 epochs and record the average action value of the testing frames after each training epoch. The learning performance regarding the convergence rate is also provided as shown in Fig. 6 for these three algorithms, where the average state-action values converge toward the optimal ones. The experimental results also demonstrate that the learning process of DCRL is faster and more stable than those of DQN and PER.

C. Additional Exploratory Experiments

The proposed DCRL aims to take advantage of the informative transitions in the replay memory and make the agent draw lessons from human curriculum strategy. Hence, some recent memory-based reinforcement learning algorithms can also achieve better results with the help of the proposed curriculum training method of DCRL. In this paper, we further apply DCRL to double DQN [34] and dueling network [35] and present experimental results to verify the generality and practicability of DCRL. The fundamental algorithms are the same as those in [34] and [35]. We incorporated their replay

TABLE III
AVERAGE REWARD PER EPISODE OF DQN, PER, AND DCRL

Game Name	DQN($\pm$std)	PER($\pm$std)	DCRL($\pm$std)
Space Invaders	572($\pm$131)	639($\pm$68)	651($\pm$119)
Carnival	4019.7($\pm$469.1)	4041.7($\pm$406.6)	4066.2($\pm$532.4)
Breakout	87.7($\pm$40.3)	108.6($\pm$50.9)	104.6($\pm$32.0)
Boxing	9.2($\pm$15.0)	11.9($\pm$4.2)	13.6($\pm$9.1)
Pong	5.9($\pm$4.3)	7.2($\pm$5.5)	11.0($\pm$4.6)
Kung-Fu Master	16708($\pm$7998)	15445($\pm$7555)	16371($\pm$5254)
Skiing	-26295($\pm$5815)	-24969($\pm$7310)	-22524($\pm$6296)
River Raid	4172($\pm$622)	5401($\pm$472)	6347($\pm$540)
Enduro	127.0($\pm$78)	144.1($\pm$92)	221.8($\pm$119)
Alien	996($\pm$203)	1113($\pm$129)	1167($\pm$235)
Montezuma's Revenge	0($\pm$0)	0($\pm$0)	0($\pm$0)
MS. PacMan	1217($\pm$252)	1505($\pm$122)	1924($\pm$327)

memory with our DCRL method for comparison. We employ the same two evaluation criteria as mentioned earlier. The values of hyperparameters are the same as those in Table I. As shown in Fig. 7, four groups of experimental results, which belong to the considered four classes of typical games, respectively, show that both of the double DQN and dueling network algorithms attain better performance with the DCRL method.

V. CONCLUSION

In this paper, we investigate the integration of curriculum learning with deep reinforcement learning, and a DCRL

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

algorithm is presented. The learning scheme depends on what the agent has learned from the environment instead of prior knowledge. The relationship between the magnitude of TD error and the current curriculum difficulty is a major evaluation indicator for the sample efficiency. Moreover, coverage penalty reduces the sample frequency of transitions that are used for too many times to alleviate the trend of overtraining. The experimental results demonstrate the success of the proposed DCRL by comparison to DQN and PER algorithms. More experimental results further show that other memory-based deep reinforcement learning algorithms, such as double DQN and dueling network, can also achieve a better performance with the proposed DCRL training paradigm. Our future work will focus on in-depth theoretical research on curriculum learning in deep reinforcement learning and its applications to other continuous control methods, such as deep deterministic policy gradient [48], [49].

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[2] M. L. Littman, "Reinforcement learning improves behaviour from evaluative feedback," *Nature*, vol. 521, no. 7553, pp. 445–451, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. NIPS*, 2012, pp. 1097–1105.

[4] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[5] H. Goh, N. Thome, M. Cord, and J.-H. Lim, "Learning deep hierarchical visual feature coding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2212–2225, Dec. 2014.

[6] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. NIPS*, 2014, pp. 3104–3112.

[8] P. P. Brahma, D. Wu, and Y. She, "Why deep learning works: A manifold disentanglement perspective," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 10, pp. 1997–2008, Oct. 2016.

[9] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *Proc. IJCNN*, 2010, pp. 1–8.

[10] G. Cuccu, M. Luciw, J. Schmidhuber, and F. Gomez, "Intrinsically motivated neuroevolution for vision-based reinforcement learning," in *Proc. ICDL*, Aug. 2011, pp. 1–7.

[11] J. Cheng, J. Yi, and D. B. Zhao, "Neural network based model reference adaptive control for ship steering system," *Int. J. Inf. Technol.*, vol. 11, no. 6, p. 75, 2005.

[12] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *Proc. IJCNN*, Jun. 2012, pp. 1–8.

[13] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 621–634, Mar. 2014.

[14] V. Mnih *et al.* (2013). "Playing atari with deep reinforcement learning." [Online]. Available: http://arxiv.org/abs/1312.5602

[15] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[16] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. learn.*, vol. 8, nos. 3–4, pp. 293–321, 1992.

[17] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learn.*, vol. 13, no. 1, pp. 103–130, 1993.

[18] D. Andre, N. Friedman, and R. Parr, "Generalized prioritized sweeping," in *Proc. Adv. NIPS*, 1998, pp. 1001–1007.

[19] R. Dearden, "Structured prioritized sweeping," in *Proc. Int. Conf. Mach. Learn.*, 2001, pp. 82–89.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. (2016) "Prioritized experience replay." [Online]. Available: https://arxiv.org/abs/1511.05952

[21] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Networks Design*. Boston, MA, USA: PWS Publishing Company, 1996.

[22] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[23] S. Braun, D. Neil, and S.-C. Liu, "A curriculum learning method for improved noise robustness in automatic speech recognition." [Online]. Available: https://arxiv.org/abs/1606.06864

[24] C. Gong *et al.*, "Multi-modal curriculum learning for semi-supervised image classification," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3249–3260, Jul. 2016.

[25] Y. Shi, M. Larson, and C. M. Jonker, "Recurrent neural network language model adaptation with curriculum learning," *Comput. Speech Lang.*, vol. 33, no. 1, pp. 136–154, 2015.

[26] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Proc. Adv. NIPS*, 2010, pp. 1189–1197.

[27] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann, "Self-paced learning with diversity," in *Proc. Adv. NIPS*, 2014, pp. 2078–2086.

[28] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, "Self-paced curriculum learning," in *Proc. AAAI*, 2015, pp. 2694–2700.

[29] D. Meng, Q. Zhao, and L. Jiang. (2015). "What objective does self-paced learning indeed optimize?" [Online]. Available: http://arxiv.org/abs/1511.06049

[30] Q. Zhao, D. Meng, L. Jiang, Q. Xie, Z. Xu, and A. G. Hauptmann. "Self-paced learning for matrix factorization," in *Proc. AAAI*, 2015, pp. 3196–3202.

[31] T. Cohn, C. D. V. Hoang, E. Vymolova, K. Yao, C. Dyer, and G. Haffari. (2016). "Incorporating structural alignment biases into an attentional neural translation model." [Online]. Available: https://arxiv.org/abs/1601.01085

[32] H. Mi, B. Sankaran, Z. Wang, and A. Ittycheriah. (2016). "Coverage embedding models for neural machine translation." [Online]. Available: http://arxiv.org/abs/1605.03148

[33] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. (2016) "Modeling coverage for neural machine translation." [Online]. Available: https://arxiv.org/abs/1601.04811

[34] H. van Hasselt, A. Guez and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, 2016, pp. 2094–2100.

[35] Z. Wang *et al.*, "Dueling network architectures for deep reinforcement learning," in *Proc. ICML*, 2016, pp. 1995–2003.

[36] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[37] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Proc. ICML*, 1995, pp. 30–37.

[38] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2016, pp. 4945–4949.

[39] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2016, pp. 4960–4964.

[40] A. Kumar *et al.* (2015). "Ask me anything: Dynamic memory networks for natural language processing." [Online]. Available: https://arxiv.org/abs/1506.07285

[41] R. G. Reilly and N. Sharkey, *Connectionist Approaches to Natural Language Processing*. Abingdon, U.K.: Routledge, 2016.

[42] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "The importance of experience replay database composition in deep reinforcement learning," in *Proc. NIPS Workshop Deep Rein. Learn.*, 2015, pp. 1–9.

[43] S. Ishii, W. Yoshida, and J. Yoshimoto, "Control of exploitation–exploration meta-parameter in reinforcement learning," *Neural Netw.*, vol. 15, nos. 4–6, pp. 665–687, Jun./Jul. 2002.

[44] P. Abbeel and A. Y. Ng, "Exploration and apprenticeship learning in reinforcement learning," in *Proc. ICML*, 2005, pp. 1–8.

[45] M. Pecka and T. Svoboda, "Safe exploration techniques for reinforcement learning—An overview," in *Proc. MESAS*, 2014, pp. 357–375, doi: 10.1109/TNNLS.2017.2654539.

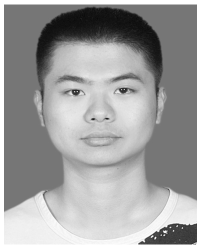
[46] Y. Wu *et al.* (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation." [Online]. Available: https://arxiv.org/abs/1609.08144

[47] 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.

[48] T. P. Lillicrap *et al.* (2015). "Continuous control with deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1509.02971

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.

[49] D. B. Zhao and Y. H. Zhu, "MEC—A near-optimal online reinforcement learning algorithm for continuous deterministic systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 2, pp. 346–356, Feb. 2015.



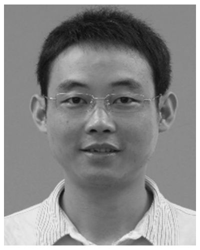
Zhipeng Ren received the B.E. degree in automation from the Department of Control and Systems Engineering, School of Management and Engineering, Nanjing University, Nanjing, China, in 2015, where he is currently pursuing the M.S. degree in control science and engineering.

His current research interests include machine learning, deep learning, and reinforcement learning.

Daoyi Dong (S'05–M'06–SM'11) received the B.E. degree in automatic control and the Ph.D. degree in engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

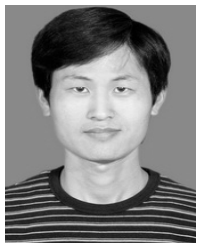
He was with the Institute of Systems Science, Chinese Academy of Sciences, Beijing, China, and the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China. He held visiting positions at Princeton University, Princeton, NJ, USA, RIKEN, Wako-Shi, Japan, and The University of Hong Kong, Hong Kong. He is currently an Associate Professor with the University of New South Wales, Canberra, ACT, Australia. His current research interests include quantum control, multiagent systems, and machine learning.

Dr. Dong received the ACA Temasek Young Educator Award by the Asian Control Association. He was a recipient of the International Collaboration Award and the Australian Post-Doctoral Fellowship from the Australian Research Council. He was a co-recipient of the Guan Zhao-Zhi Award at the 34th Chinese Control Conference, the Best Theory Paper Award at the 11th World Congress on Intelligent Control and Automation, and the Best Student Paper Award at the 2017 Australian and New Zealand Control Conference. He is a Chair of the Technical Committee on Quantum Cybernetics and the IEEE Systems, Man and Cybernetics Society. He serves as an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.



Huaxiong Li (M'11) received the M.E. degree in control theory and control engineering from Southwest University, Nanjing, China, in 2006, and the Ph.D. degree in management science and engineering from Nanjing University, Nanjing, in 2009.

He was a Visiting Researcher with the Department of Computer Science, University of Regina, Regina, SK, Canada, from 2007 to 2008. He held visiting positions at The University of Hong Kong, Hong Kong. He is currently an Associate Professor with the Department of Control and Systems Engineering, School of Management and Engineering, Nanjing University. His current research interests include machine learning, granular computing, and intelligent information processing.



Chunlin Chen (S'05–M'06) received the B.E. degree in automatic control and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2001 and 2006, respectively.

He was with the Department of Chemistry, Princeton University, Princeton, NJ, USA, from 2012 to 2013. He held visiting positions at the University of New South Wales, Canberra, ACT, Australia, and the City University of Hong Kong, Hong Kong. He is currently a Professor and the Head of the Department of Control and Systems Engineering, School of Management and Engineering, Nanjing University, Nanjing, China. His current research interests include machine learning, intelligent control, and quantum control.

Dr. Chen is the Co-Chair of the Technical Committee on Quantum Cybernetics and the IEEE Systems, Man and Cybernetics Society.

Authorized licensed use limited to: TONGJI UNIVERSITY. Downloaded on May 06,2021 at 12:16:08 UTC from IEEE Xplore. Restrictions apply.