Received 4 July 2023, accepted 26 August 2023, date of publication 6 September 2023, date of current version 12 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3312383

RESEARCH ARTICLE

Contrastive Learning Methods for Deep Reinforcement Learning

DI WANG, (Member, IEEE), AND MENGQI HU, (Member, IEEE)
Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60609, USA

Corresponding author: Di Wang (dwang66@uic.edu)

This work was supported by the National Science Foundation under Grant CMMI 1634738.

ABSTRACT Deep reinforcement learning (DRL) has shown promising performance in various application areas (e.g., games and autonomous vehicles). Experience replay buffer strategy and parallel learning strategy are widely used to boost the performances of offline and online deep reinforcement learning algorithms. However, state-action distribution shifts lead to bootstrap errors. Experience replay buffer learns policies with elder experience trajectories, limiting its application to off-policy algorithms. Balancing the new and the old experience is challenging. Parallel learning strategies can train policies with online experiences. However, parallel environmental instances organize the agent pool inefficiently with higher simulation or physical costs. To overcome these shortcomings, we develop four lightweight and effective DRL algorithms, instance-actor, parallel-actor, instance-critic, and parallel-critic methods, to contrast different-age trajectory experiences. We train the contrast DRL according to the received rewards and proposed contrast loss, which is calculated by designed positive/negative keys. Our benchmark experiments using PyBullet robotics environments show that our proposed algorithm matches or is better than the state-of-the-art DRL algorithms.

INDEX TERMS Contrastive learning, deep reinforcement learning, different-age experience, experience replay buffer, parallel learning.

I. INTRODUCTION

Deep neural network provides powerful representation capabilities [1] for reinforcement learning. In the past few years, deep reinforcement learning (DRL) has achieved great success in various areas, such as market strategy [2], [3], robot control [4], [5], and task planning [6], [7], [8]. Through repeated interactions among agents and environments, DRL learns a policy to maximize the expected return, represented as a state or state-action value function. The actor-critic architecture [9] is commonly used in DRL, where the actor network generates actions based on state input, and the critic network approximates the state or state-action value function.

The effectiveness of the experience Replay buffer strategy is proven to boost the performances of deep reinforcement learning algorithms. Experience data is stored in an extra buffer and sampled for model training. The replay buffer can only be adopted by off-policy algorithms (e.g., deep deterministic policy gradient) since it stores data related to older policies. Lee et al. [10] observe that for the off-policy algorithms, state-action distribution shift leads to severe bootstrap error. Kaplanis et al. [11] study the balance between the new and the old experiences. If primarily focusing on recent experiences, the agent can easily forget what to do when it revisits states it has not seen in a while, resulting in catastrophic forgetting and instability. However, by retaining too many old experiences, the agent might focus too much on replaying states that are irrelevant to its current policy, resulting in a sluggish and noisy improvement in its performance. Similarly, Zhang and Sutton [12] prove that using ‘‘different-age’’ experiences directly influences performance. Novati and Koumoutsakos [13] emphasizes the importance of the similarity between the old and most recent policies. Wang et al. [14] analyze the challenges of balancing policy regularization and policy cloning in offline RL methods. In this paper, we try to solve this issue from the perspective of the contrastive learning aspect. The proposed algorithm balances the trade-off between the old and the current policies through policy regularization. Another issue caused by the

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Abdur Razzaque.

unbalanced usage of ''different-age'' policies is the compatibility issue [15].

Parallel learning strategies are effective for the online DRL algorithms, where asynchronous execution on paralleled environment instances [16] collects less correlated online experiences for training purposes. However, paralleled execution of multiple environment instances requires a large overhead of computational resources for environmental simulations or physical environmental resources. Besides, Clemente et al. [17] study the issues of effectiveness of using parallel agents' experiences. On the other hand, a parallel learning strategy can also be used for the offline DRL algorithms to fully use the computation resources, and collected experiences will be stored in a common replay buffer. The issue of experience replay buffer discussed before also exists under this case. Moreover, the parallel learning method provides a promising way to regularize policies by balancing cloned old policies and focusing on current ones. We try to illustrate this critical issue deeply and propose four possible combination methods. Based on our experiments, policy regularization and this balance between old and current policies can improve the performance of RL. These results match the conclusions of existing works [11], [12], while we use different methods in this paper.

This research proposes a contrast learning enhanced DRL algorithm to address these issues. Contrast learning is an unsupervised learning approach to distinguishing the positive examples of similar objects and negative examples of dissimilar objects [18]. The contrastive examples are stored in a query dictionary [19]. The representation vectors of target objects (called a query) should closely approximate the positive examples (called positive keys) and differentiate negative examples (called negative keys). It is demonstrated that contrast learning can boost data efficiency in image recognition where the performance of the proposed contrast predictive model outperforms other semi-supervised and supervised learning methods using only 1% of the dataset [20]. The contrast information extracted from experience trajectories can stabilize the observation data, de-correlate the updating process, and thus improve the model performances. However, there are three challenges to integrating DRL with contrast learning: (i) How to use contrastive learning to balance ''different-age'' experiences, especially the elder experience and the newest experiences? (ii) How to generate positive and negative samples? (iii) How to design an efficient contrast loss function for the training?

To address these challenges, we proposed four contrast deep reinforcement learning (CDRL) methods: instance-actor, parallel-actor, instance-critic, and parallel-critic.

- In the instance-actor method, the target actor network generates negative keys, and the actor network generates a query using the same mini-batch sample data from the experience replay buffer. The target actor network uses the most recently experienced data to generate positive keys.
- In the parallel-actor method, an extra actor network is trained to generate positive/negative keys with the most recently experienced data. The critic network is used to classify the positive and negative keys based on value estimations.

The instance-critic and parallel-critic methods are similar to the above two methods. More details are provided in Section IV. To propose a proper contrast loss function, we measure the distances between the query and positive/negative keys using the Jensen-Shannon divergence and add importance weights to adjust the loss function values based on the quality of positive/negative keys. We hypothesize that the quality of positive/negative keys is high when the distance between the positive and negative keys is significant. A Monte Carlo method is adopted to calculate the expectation of the contrast loss function.

We implement the CDRL algorithms based on a deep deterministic policy gradient (DDPG), since many advanced algorithms are developed from DDPG, like twin delayed deep deterministic (TD3). The performance of our proposed CDRL algorithms is compared with six state-of-the-art reinforcement learning algorithms, including DDPG [21], D4PG [22], TD3 [23], PPO [24], SAC [25], and A2C [16]. The simulation results on PyBullet robotics environments show that the proposed CDRL can boost performances and convergence speed. The Pybullet environments are widely used in research of DRL. They are free but a little bit different from the famous MuJoCo environments.

In summary, our main contributions are:

1) To the best of our knowledge, the integration of contrast learning and DRL in balancing different-age experiences is less studied. Contrast learning can enhance the actor/critic network representation ability and knowledge learned from different-age trajectories. The format of the parallel learning method is not novel. However, it serves as a great cutting point for the combination of contrastive learning and reinforcement learning, and this research aspect is less studied.
2) To study contrast learning from different viewpoints, we propose four contrast learning methods, each of which adopts a different way to generate positive/negative examples with different merits.
3) A novel contrast loss function is proposed, considering the quality of positive/negative keys.

The organization of the remainder of this paper is as follows: Section II illustrates the existing related work. Section III describes the foundation of reinforcement learning and contrast learning. In Section IV, we show details of our proposed CDRL algorithms. Section V demonstrates the advantages of our proposed method over the state-of-the-art. In Section VI, we conclude our proposed approach and discuss future studies.

II. RELATED WORK

In the past few years, extensive studies have focused on boosting the performance of DRL with various actor-critic algorithms. Schulman et al. [26] propose the trust region policy optimization algorithm for improving the policy monotonically. Kronecker-factored trust region is introduced by Wu et al. [27] to calculate the natural gradient efficiently. Silver et al. [21] introduce the deterministic policy gradient algorithm to replace the stochastic policy gradient algorithm with the experience replay buffer strategy. Fujimoto et al. [23] approximate the actual state-action value function with the minimum values of two critic networks' estimations. Millán-Arias et al. [28] learn the environmental dynamics with extra provided advice. Wang [29] combine Hebbian learning with DRL to boost data efficiency.

Experience replay buffer and parallel learning are hot research topics in DRL. Kong et al. [30] study the unbalanced utilization of the experience replay buffer. A half-normal sampling probability window is proposed to sample online experiences with higher probabilities. Li et al. [31] propose to use the trajectory experience replay buffer to overcome the disadvantages of offline algorithms. Similarly, Yang et al. [32] propose an episodic memory as complementary to the experience replay buffer. Luu and Yoo [33] propose a Hindsight Goal Ranking (HGR) sampling algorithm for the experience replay buffer.

Schmitt et al. [34] enable the participating agents to share their experience with a standard replay module. Similar to A3C, Nair et al. [35] propose a distributed learning architecture for deep Q-network [36], which includes parallel actors, parallel learners, a distributed neural network, and a distributed experience replay buffer. Grounds and Kudenko [37] propose a parallel SARSA algorithm where agents are trained separately and share weights periodically. Horgan et al. [38] propose using multiple actors with a shared neural network to interact with environments. A learner is employed to sample data and update the parameters of the shared neural network. Nair et al. [35] deploy multiple DQN agents on parallel environment instances to collect massive experience into the experience replay buffer.

Contrast learning provides a new unsupervised learning framework, which attracts increasing attention. He et al. [39] propose a dynamic dictionary constructor with the momentum contrast strategy, which could decouple the dictionary size from the mini-batch size and maintain consistency. Wu et al. [40] put all positive/negative keys in a memory bank. Hadsell et al. [19] propose two loss functions for similar and dissimilar pairs and restrict the dissimilar pairs within a margin. Wu et al. [40] propose a non-parametric softmax loss function to maximize the distinction between instances. Besides, the noise-contrastive estimation [41] is used to approximate the softmax value. Moreover, Bell and Bala [18] focus on learning similarity metrics with contrast learning and evaluate the performances of embedding by training with contrastive loss, object classification softmax loss, and both. Tian et al. [42] propose a contrastive multiview coding method that captures information shared between multiple sensory views. Chopra et al. [43] propose a contrast learning method with a similarity metric to address the data unbalance issue. Srinivas et al. [44] prove the success of contrastive unsupervised representations in improving the performance of DRL by finding better feature vectors of input images. Like [45], Zhu et al. utilize inputting images to generate positive and negative keys, restricting the proposed algorithm to environments with video inputs. Unlike our paper, our work focuses on efficient parallel learning methods to balance different-age experiences instead of finding powerful feature vectors on image inputs.

III. BACKGROUND

A. ACTOR-CRITIC REINFORCEMENT LEARNING

Reinforcement learning is an efficient approach to learning the decision-making process through agent-environment interactions. Given the environment state $s_t \in \mathcal{S}$ at time step t, the agent chooses actions $a_t \in \mathcal{A}$ following a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state space and the action space, respectively. After executing the actions a_t, the agent receives immediate reward r_t from the environment and transits into the next state s_{t+1} based on the transition function satisfying $p(s_{t+1} \mid s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1} \mid s_t, a_t)$. Then the agent generates the next action a_{t+1} and repeats the above process until the decision process is terminated. The objective function of DRL is the expectation of return represented as $J(\pi) = \mathbb{E}[R_t^\gamma \mid \pi]$, where the return is denoted in Eq. (1).

$$R_t^\gamma = \sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i), 0 < \gamma < 1 \tag{1}$$

Here, we take the deterministic policy gradient algorithm as an example for demonstration [46]. State-action value, or named as Q value, is the foundation of DDPG. Given s_t, a_t, π and the discounted state distribution ρ^π, Q value can be represented as in Eq. (2).

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi}[R_t^\gamma \mid s_t, a_t] \tag{2}$$

where $\rho^\pi(s') := \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_1(s) p(s \rightarrow s', t, \pi) ds$ [21]. According to the chain rule, Q value can be calculated by Eq. (3).

$$\begin{aligned} Q^\pi(s_t, a_t) = \mathbb{E}_{s\sim\rho^\pi}(&r(s_t, a_t) \\ &+ \gamma \mathbb{E}_{a_{t+1}\sim\pi}[Q^\pi(s_{t+1}, a_{t+1})]) \end{aligned} \tag{3}$$

Experience replay buffer is utilized in DDPG. β is the elder stochastic behavior policy sampled from the replay buffer. $\rho^\beta(s') := \int_S \sum_{t=1}^{\infty} \gamma^{t-1} p_1(s) p(s \rightarrow s', t, \beta) ds$. The actor network and the critic network are parameterized by ϕ and θ, respectively. The performance objective and its gradient can be written as

$$J_\beta(\pi_\phi) = \int_S \rho^\beta(s) Q^\pi(s, \pi_\phi(s)) ds \tag{4}$$

$$\bigtriangledown_\theta J_\beta(\pi_\phi) \approx \mathbb{E}_{s\sim\rho^\beta}[\bigtriangledown_\theta \pi_\phi(s) \bigtriangledown_a Q^\pi(s, a) \mid_{a=\pi_\phi(s)}] \tag{5}$$

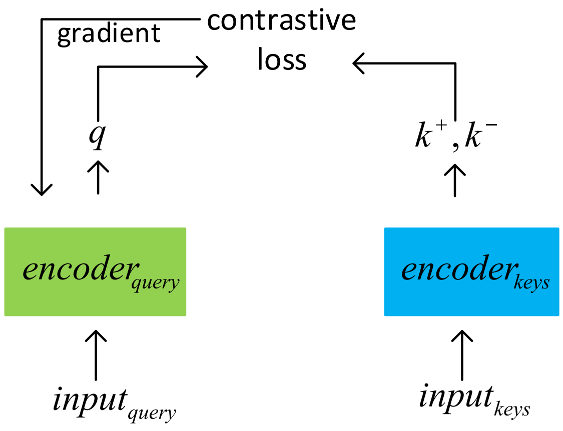


FIGURE 1. Contrastive learning mechanism (The parameters of the query encoder are optimized based on the contrastive loss. The parameters of the key encoder are optimized with soft update strategy or trained separately.).

The loss function of the critic network can be approximated as in Eq. (6).

$$L(\theta) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta}[(y_t - Q_\theta(s_t, a_t \mid \theta))^2] \tag{6}$$

where $y_t = r(s_t, a_t) + \gamma Q_{\theta'}(s_{t+1}, \mu_{\theta'}(s_{t+1}) \mid \theta')$, and target critic network and target actor network are parameterized by θ' and θ', respectively.

B. CONTRAST LEARNING

Contrast learning is to build an encoded keys dictionary $\{k_1^-, k_2^-, \ldots, k_M^-, k^+\}$. When an encoded query representation vector q comes, one positive key representation vector, k^+, matches q. The remaining M keys $\{k_1^-, k_2^-, \ldots, k_M^-\}$ are negative keys k^-. A contrastive loss is a function $L'(q, k^+, k^-)$ whose value is low when q is similar to its matched k^+ and dissimilar to k^-. As illustrated in Fig. 1, the parameters of the query encoder are trained to minimize the contrastive loss. The main challenges of contrastive learning are the selection of key encoders and the contrast loss construction. In the traditional application of contrast learning in vision tasks, the positive key inputs are generated easily by a data augmentation strategy like picture rotation. The negative key inputs indicate the picture of other categories. Thus, the quality of the key dictionary is guaranteed. However, in DRL, the environment states and the actions may be continuous. The quality of the keys dictionary can not be taken for granted. Moreover, the size of the ideal dictionary is enormous. According to these observations, we propose a new contrast loss function considering the quality of the key dictionary.

C. CONTRAST LOSS FUNCTION

One challenge of contrastive learning is to find a proper loss function. Unimodal loss functions, like mean squared error, are not very useful because representation vectors can be represented as a high-dimensional conditional probability distribution. The contrast loss is designed to minimize $D(P(q \mid input_q), P(k^+ \mid input_{k^+}))$ and maximize $D(P(q \mid input_q), P(k^- \mid input_{k^-}))$, where D is the distance estimator. $input_q$, $input_{k^+}$, and $input_{k^-}$ are the corresponding encoder inputs. Using the Jensen-Shannon divergence as a distance estimator, we propose a new contrast loss function for its application in DRL, as shown in Eqs. (7-8).

$$L' = \sum_{i=1}^{M} (\exp(D(k^+, q)) - \exp(D(k_i^-, q)))w_i \tag{7}$$

$$w_i = \frac{\exp(D(k^+, k_i^-))}{\sum_{j=1}^{M} \exp(D(k^+, k_j^-))} \tag{8}$$

where w_i is an importance weight parameter measuring the quality of the pair of keys (k^+, k_i^-). If the distance between k^+ and k_i^- is small, the key pair provides less information. If the above distance is large, the key pair provides more information. From the viewpoint of the expectation, L' can be rewritten as (9).

$$L' = \mathbb{E}_{p_{k^+,k^-}}(\exp(D(k^+, q)) - \exp(D(k^-, q))) \tag{9}$$

where p_{k^+,k^-} refers to the quality probability of the sampled keys dictionary. The Jensen-Shannon divergence operator $D(a, b)$ is calculated according to (10-11).

$$D(a, b) = \frac{1}{2} KL\left(a \Big|\Big| \frac{a+b}{2}\right) + \frac{1}{2} KL\left(b \Big|\Big| \frac{a+b}{2}\right) \tag{10}$$

$$KL(a||b) = -\sum_{x \in \mathcal{X}} a(x) \log \frac{a(x)}{b(x)} \tag{11}$$

Thus, the actor's parameters ϕ are updated according to the gradient calculated in (12), and the critic's parameters θ are updated according to the gradient calculated in (13).

$$\nabla_{\phi} \widetilde{J}(\phi) = \nabla_{\phi} J(\phi) + \beta_1 \nabla_{\phi} L'(\phi) \tag{12}$$

$$\nabla_{\theta_i} \widetilde{L}(\theta) = \nabla_{\theta} L(\theta) + \beta_2 \nabla_{\theta} L'(\theta) \tag{13}$$

where β_1 and β_2 are hyperparameters. In this paper, we set $\beta_1 = 0.5$ and $\beta_2 = 0.5$.

We referenced and tested several popular loss functions. Based on our experiments, the JSD-based loss function is better than the others. A potential reason is that trajectories in the replay buffer are collected with different policies and may differ from the current policy. The probability distributions of some policies may have less overlapped areas. The JSD-based loss function is symmetric and is designed to solve this headache. As for the training process, we do not pass the gradient through the parallel networks. Of course, this method is feasible. In this paper, we add extra sample batches in each time step. Compared with the baseline, we use the "bigger" batch size. However, this operation will not influence the complexity largely.

IV. CONTRASTIVE DEEP REINFORCEMENT LEARNING METHODS

This section discusses the four CDRL methods. The differences lie in the approaches to generating positive and negative keys.

Algorithm 1 Pseudocode of the Instance-Actor Method

```
1: Initialize critic networks $Q_{\theta_1}(s, a \mid \theta_1)$, target critic networks $Q_{\theta_1'}(s, a \mid \theta_1')$ with $\theta_1' \leftarrow \theta_1$, actor network $\pi_\phi(s)$, target actor network $\pi_{\phi'}(s)$ with $\phi' \leftarrow \phi$, experience replay buffer with size $N$.
2: for episode $k = 1$ to $K$ do
3:     for $t = 1$ to $T_k$ do
4:         Observe state $s_t$, and generate action $a_t = \pi_\phi(s_t) + \eta_t$ where $\eta_t$ denotes a random process.
5:         Execute action $a_t$, receive reward $r_t$, next state $s_{t+1}$.
6:         Add one transition $(s_t, a_t, r_t, s_{t+1})$ to replay buffer.
7:         Sample a minibatch of $b_1$ transitions $[s_t, a_t, r_t, s_{t+1}]_{b_1}$. Calculate the query vector $\pi_\phi(s_{b_1})$ and the negative keys $\pi_{\phi'}(s_{b_1})$.
8:         Fetch the recent experienced minibatch of $b_2$ transitions $[s_t]_{b_2}$. Calculate the positive keys $\pi_{\phi'}(s_{b_2})$.
9:         Update critic with the loss function in Eq. (6).
10:        Calculate the contrast loss with the query vector, positive and negative keys according to Eq. (9). Update actor with the performance function with the modified loss function Eq. (12).
11:        Update the target actor and the target critic networks with the soft update trick.
12:    end for
13: end for
```

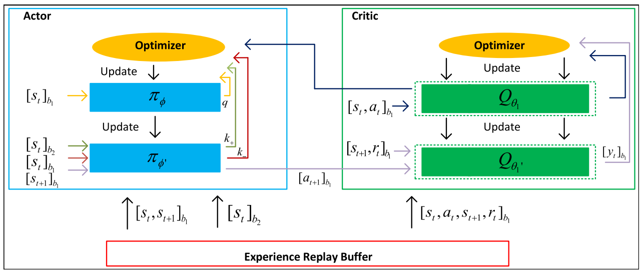


FIGURE 2. Instance-actor contrastive learning method.

A. INSTANCE-ACTOR METHOD

In this method, the actor network is considered a query encoder, and the target actor network is considered a key encoder. As shown in Fig. 2, in each training time step, we sample a batch of data $[s_t, a_t, r_t, s_{t+1}]_{b_1}$ from the replay buffer and $[s_t]_{b_2}$ indicates the most recent experienced trajectories. Actor network π_ϕ takes $[s_t]_{b_1}$ as inputs and generates query. Target actor network $\pi_{\phi'}$ takes $[s_t]_{b_1}$ as inputs and generates the negative key. Because the target actor network utilizes the temporal difference error strategy and the delay updating strategy, the target actor network parameters are updated behind the actor network parameters. To keep exploring new areas in the action space, the output of the actor network and the target network should be different, which prohibits falling into near-optimal actions. Thus, we maximize the distance between the query and the negative keys according to Eq. (12). On the other hand, the target actor network $\pi_{\phi'}$ takes the most recent experienced trajectories as inputs and generates the positive key. Since the policy keeps being trained iteratively, the current policy should be better than the elder policy resulting in the recently experienced state having a higher possibility of being visited by the optimal policy than the elder state. Thus we should minimize the distance between the query and the positive key to learn the knowledge embedded in online policy and mitigate the side effects of the off-policy. The pseudocode is presented in Algorithm 1.

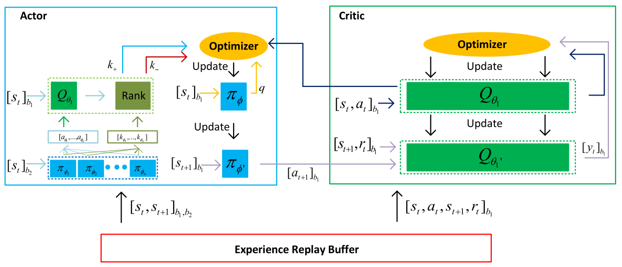


FIGURE 3. Parallel-actor contrastive learning method.

B. PARALLEL-ACTOR METHOD

In this method, one actor network is the query encoder, and the remaining n actor networks are the key encoders. As shown in Fig. 3, in each training time step, we sample a batch of data $[s_t, a_t, r_t, s_{t+1}]_{b_1}$ from the replay buffer. Actor network π_ϕ takes $[s_t]_{b_1}$ as inputs and generates query. Meanwhile, we maintain extra n actor networks $\pi_{\phi_1}, \ldots, \pi_{\phi_n}$. These actor networks take most recent experienced trajectories $[s_t]_{b_2}$ as inputs and generate the dictionary $\{k^+, k^-\}$. A ranker sorts the keys based on the value estimates provided by the critic network Q_{θ_1}. The key with the highest Q value is the positive key k^+, and the remaining keys are classified as the negative keys k^-. By minimizing the distance between the query and the positive key and maximizing the distance between the query and the negative keys as in Eq. (12), the actor network π_ϕ directly learns the representations of online policies from different peers. Since the n actor networks are trained separately based on the same experience replay buffer, we raise the data efficiency by times. Moreover, we exploit the difference in peers' training trajectories. The positive key labels are currently the best policy, and the negative keys are labeled as bad policy examples. DDPG algorithm selects actions deterministically without evaluations over the action spaces. Integrating this information through contrastive learning leads to raising the exploration ability. The pseudocode is presented in Algorithm 2.

C. INSTANCE-CRITIC METHOD

This method adopts the critic network as the query encoder and the target critic network as the keys encoder. As shown in Fig. 4, in each training time step, we sample a batch of data $[s_t, a_t, r_t, s_{t+1}]_{b_1}$ from the replay buffer and $[s_t, a_t]_{b_2}$ denotes the most recent experienced data. Critic network θ takes $[s_t, a_t]_{b_1}$ as inputs and generates query. Target critic network

Algorithm 2 Pseudocode of the Parallel-Actor Method

```
1: Initialize critic networks $Q_{\theta_1}(s, a \mid \theta_1)$, target critic networks $Q_{\theta_1'}(s, a \mid \theta_1')$ with $\theta_1' \leftarrow \theta_1$, actor network $\pi_\phi(s)$, target actor network $\pi_{\phi'}(s)$ with $\phi' \leftarrow \phi$, experience replay buffer with size $N$.
2: Initialize $n$ extra actor networks $\pi_{\phi_i}(s)$ i=1,2,... n.
3: for episode $k = 1$ to $K$ do
4:     for $t = 1$ to $T_k$ do
5:         Observe state $s_t$, and generate action $a_t = \pi_\phi(s_t) + \eta_t$ where $\eta_t$ denotes a random process.
6:         Execute action $a_t$, receive reward $r_t$, next state $s_{t+1}$.
7:         Add one transition $(s_t, a_t, r_t, s_{t+1})$ to replay buffer.
8:         Sample a minibatch of $b_1$ transitions $[s_t, a_t, r_t, s_{t+1}]_{b_1}$. Calculate the query vector $\pi_\phi(s_{b_1})$.
9:         Fetch the recent experienced minibatch of $b_2$ transitions $[s_t]_{b_2}$. Calculate $n * b_2$ keys $\pi_{\phi_i}(s_{b_2})$, i=1,2,..n. and corresponding $n * b_2$ actions $a_{n*b_2}$.
10:        Rank keys according to the estimations of Q value $Q_{\theta_1}.(s_{b_2}, a_{n*b_2})$. $b_2$ Keys with the highest Q values are viewed as the positive keys, while the remaining serve as the negative keys.
11:        Update critic with the loss function in Eq. (6).
12:        Calculate the contrast loss with the query vector, positive and negative keys according to Eq. (9). Update actor with the performance function in Eq. (12).
13:        Update the extra $n$ actors with the performance function in Eq. (5).
14:        Update the target actor and the target critic networks with the soft update trick.
15:     end for
16: end for
```

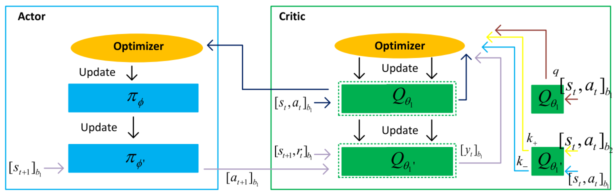


FIGURE 4. Instance-critic contrastive learning method.

θ' takes $[s_t, a_t]_{b_2}$ as inputs and generates the positive key. Meanwhile, target critic network θ' takes $[s_t, a_t]_{b_1}$ as inputs and generates the negative keys. Because the target critic network utilizes the delay updating strategy, the parameters of the target critic network are updated behind the parameters of the critic network. Fujimoto et al. [23] prove that more updates of the critic network will provide more accurate estimations. The output of the critic network and the target critic network should be different, which overcomes underestimation and overestimation to some extent. Thus, we maximize

Algorithm 3 Pseudocode of the Instance-Critic Method

```
1: Initialize critic networks $Q_{\theta_1}(s, a \mid \theta_1)$, target critic networks $Q_{\theta_1'}(s, a \mid \theta_1')$ with $\theta_1' \leftarrow \theta_1$, actor network $\pi_\phi(s)$, target actor network $\pi_{\phi'}(s)$ with $\phi' \leftarrow \phi$, experience replay buffer with size $N$.
2: for episode $k = 1$ to $K$ do
3:     for $t = 1$ to $T_k$ do
4:         Observe state $s_t$, and generate action $a_t = \pi_\phi(s_t) + \eta_t$ where $\eta_t$ denotes a random process.
5:         Execute action $a_t$, receive reward $r_t$, next state $s_{t+1}$.
6:         Add one transition $(s_t, a_t, r_t, s_{t+1})$ to replay buffer.
7:         Sample a minibatch of $b_1$ transitions $[s_t, a_t, r_t, s_{t+1}]_{b_1}$. Calculate the query vector $Q_{\theta_1}(s_{b_1}, a_{b_1})$ and the negative keys $Q_{\theta'}(s_{b_1}, a_{b_1})$.
8:         Fetch the recent experienced minibatch of $b_2$ transitions $[s_t, a_t]_{b_2}$. Calculate the positive keys $Q_{\theta'}(s_{b_2}, a_{b_2})$.
9:         Update actor with the loss function in Eq. (5).
10:        Calculate the contrast loss with the query vector, positive and negative keys according to Eq. (9). Update critic with the performance function with the modified loss function Eq. (13).
11:        Update the target actor and the target critic networks with the soft update trick.
12:     end for
13: end for
```

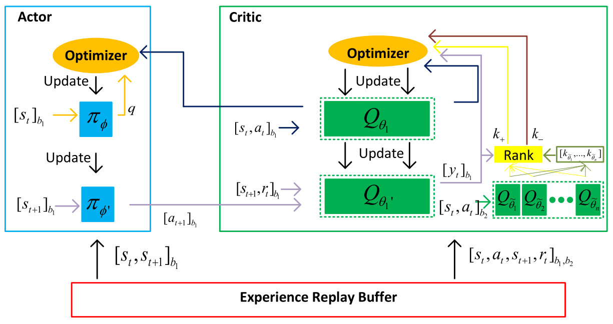


FIGURE 5. Parallel-critic contrastive learning method.

the distance between the query and the negative keys according to Eq. (13). On the other hand, since the policy keeps being trained iteratively, the current policy should be better than the elder policy, resulting in the recently experienced trajectories being more important than the elder trajectories. As proved in [15], the learned knowledge from the recent trajectories will solve the compatibility issue among the actor and critic networks. Thus we should minimize the distance between the query and the positive key. The pseudocode is presented in Algorithm 3.

D. PARALLEL-CRITIC METHOD

This method adopts one of the critic networks as a query encoder and extra critic networks as key encoders. As shown

TABLE 1. Summary of four contrastive learning methods.

Methods	Emphasize	Use Parallel-learning Framework	Complexity	Data Efficiency
Instance-actor CDRL	Policy Improvement	No	Similar to DDPG	Higher than DDPG
Parallel-actor CDRL	Policy Improvement	Yes	Larger than DDPG	Much higher than DDPG
Instance-critic CDRL	Policy Evaluation	No	Similar to DDPG	Higher than DDPG
Parallel-critic CDRL	Policy Evaluation	Yes	Larger than DDPG	Much higher than DDPG

Algorithm 4 Pseudocode of the Parallel-Critic Method

```
1: Initialize critic networks $Q_{\theta_1}(s, a \mid \theta_1)$, target critic networks $Q_{\theta_1'}(s, a \mid \theta_1')$ with $\theta_1' \leftarrow \theta_1$, actor network $\pi_\phi(s)$, target actor network $\pi_{\phi'}(s)$ with $\phi' \leftarrow \phi$, experience replay buffer with size $N$.
2: Initialize $n$ extra critic networks $Q_{\widetilde{\theta_i}}$ i=1,2,...n.
3: for episode $k = 1$ to $K$ do
4:     for $t = 1$ to $T_k$ do
5:         Observe state $s_t$, and generate action $a_t = \pi_\phi(s_t) + \eta_t$ where $\eta_t$ denotes a random process.
6:         Execute action $a_t$, receive reward $r_t$, next state $s_{t+1}$.
7:         Add one transition $(s_t, a_t, r_t, s_{t+1})$ to replay buffer.
8:         Sample a minibatch of $b_1$ transitions $[s_t, a_t, r_t, s_{t+1}]_{b_1}$. Calculate the query vector $Q_{\theta_1}(s_{b_1}, a_{b_1})$.
9:         Fetch the recent experienced minibatch of $b_2$ transitions $[s_t, a_t, r_t, s_{t+1}]_{b_2}$. Calculate $n{*}b_2$ keys $Q_{\widetilde{\theta_i}}$, i=1,2,..n.
10:        Rank keys according to the $MSE(Q_{\widetilde{\theta_i}}, y_t)$, i=1,2,..n). $b_2$ Keys with the lowest MSE values are viewed as the positive keys, while the remaining serve as the negative keys.
11:        Update actor with the loss function in Eq. (5).
12:        Calculate the contrast loss with the query vector, positive and negative keys according to Eq. (9). Update critic with the performance function with the modified loss function Eq. (13).
13:        Update the extra $n$ critic with the performance function in Eq. (6).
14:        Update the target actor and the target critic networks with the soft update trick.
15:    end for
16: end for
```

in Fig. 5, we sample a batch of data $[s_t, a_t, r_t, s_{t+1}]_{b_1}$ from the replay buffer, and $[s_t, a_t, r_t, s_{t+1},]_{b_2}$ denotes the most recent experienced data. Critic network Q_{θ_1} takes $[s_t, a_t]_{b_1}$ as inputs and generates query. Meanwhile, we maintain n critic networks $Q_{\widetilde{\theta_1}}, \ldots, Q_{\widetilde{\theta_n}}$. These n critic networks take $[s_t, a_t]_{b_2}$ as inputs and generate the dictionary $\{k^+, k^-\}$. A ranker sorts the keys based on the mean square errors $(y_t - Q_{\widetilde{\theta_{1,\ldots,n}}})^2$. The key with the lowest error is the positive key, and the remaining is the negative one. Similarly, we minimize the distance between the query and the positive key while maximizing the distance between the query and the negative keys. Since the n critics are trained separately with the same experience replay buffer, we increase the data efficiency by times. Moreover, we exploit the difference in peers' training trajectories. The positive key labels are currently the most accurate Q value, and the negative keys are labeled as less accurate Q values. The idea of multiple critic networks is similar to TD3, which has proven performance improvement. However, the difference is that the most recently experienced trajectories are utilized instead of randomly sampled from the buffer. Moreover, Zhang et al. [47] find the underestimation issue in TD3 in which two critic networks are utilized. They propose to use three critic networks to find the balanced weights. However, our proposed methods learn all critic networks' representations. The pseudocode is presented in Algorithm 4.

Table 1 summarizes the differences among different proposed methods. Policy improvement and policy evaluation are two essential parts of DRL methods. Based on [48], better policy evaluations can always lead to better policies. Parallel-actor and parallel-critic methods maintain extra numbers of actor networks and critic networks in the parallel-learning framework. Besides, the introduction of the ranker increases the complexity. Furthermore, the inaccurate estimated Q value brings accumulated error in the ranker of the parallel-actor method in the training process. On the other hand, the parallel-critic method takes the stable y value in the ranker.

The complexity of Q learning is intractable for the continuous state-action spaces [49]. In this paper, we analyze the time complexity of our proposed methods simply. Assume d is the length of the episode and m is the training episodes. DDPG visits each data point in the replay buffer multiple times. s denotes the sampling times of each data point at each time step. The complexity of DDPG is $m^2 * d^2 * s$. The complexity of the instance-actor and the instance-critic methods are $m^2 * d^2 * s + m * d * b_2$, where b_2 is the batch size of the recent experience trajectories. The complexity of the parallel-actor and the parallel-critic methods are $(n+1) * m^2 * d^2 * s + m * d * b_2 + m * d * b_2 * log(b_2)$. n extra parallel actor networks or critic networks are maintained, and a sequencing operation is required in the ranker operator.

Our proposed contrastive learning methods have higher data efficiency than DDPG. Assume the batch size of DDPG is b_1. The total number of positive and negative keys is n times of b_1. In instance-actor and instance-critic methods, $b_2 = n{*}b_1$. For each time step, extra n times experience trajectories will be utilized for the training purpose. In parallel-actor and parallel-critic methods, $b_2 = b_1$. For each time step, an extra $2 * n$ times experience trajectories will be utilized

TABLE 2. The maximum mean rewards of five proposed methods in four environments.

Environment	Instance-actor CDRL	Parallel-actor CDRL	Instance-critic CDRL	Parallel-critic CDRL	Combined All
Ant	1292.49	1330.04	1273.85	1384.64	1426.67
HalfCheetah	1783.11	1913.99	1874.53	2010.44	2105.12
InvDoublePen	7421.20	8088.74	7342.73	8080.37	8232.75
InvPenSwingup	827.65	834.69	802.05	841.32	845.17

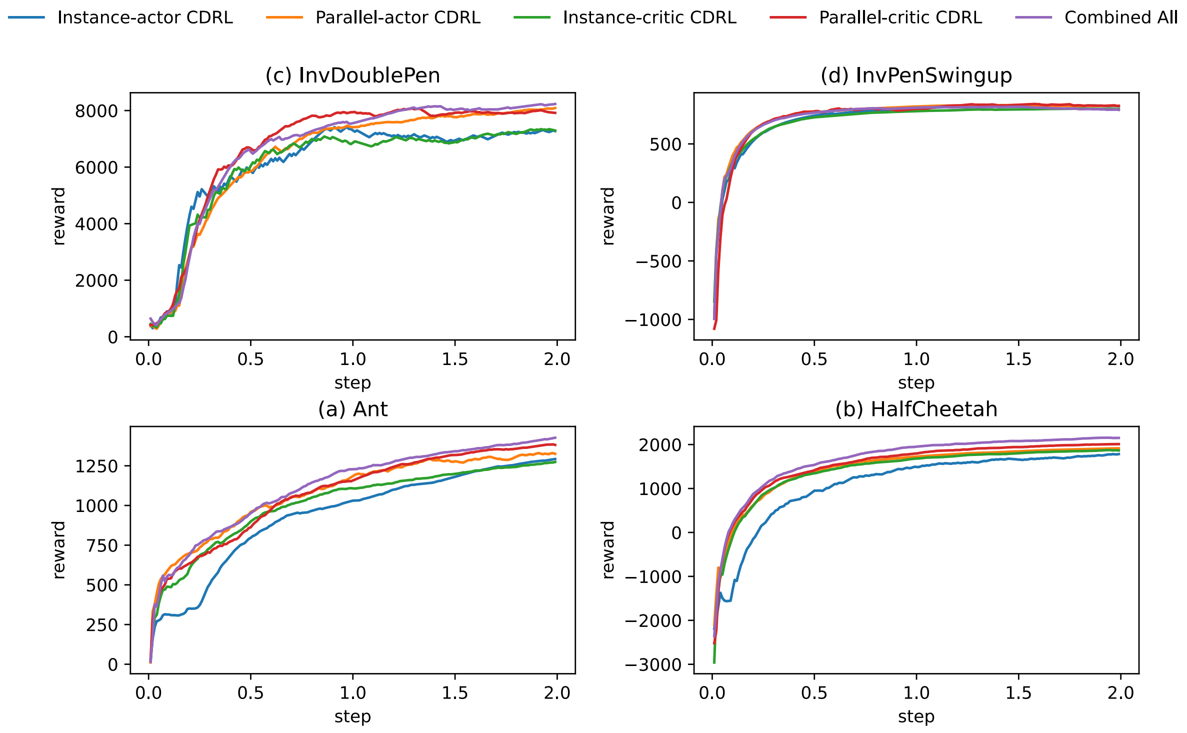


FIGURE 6. Performance comparisons among five proposed methods in four environments.

for the training purpose because extra n neural networks are trained.

Similar to [21] and [39], the convergence of proposed methods can be proved. According to Eq. 7, when our proposed methods converge, positive keys and negative keys are similar. Namely, $D(k^+, q) \approx D(k_i^-, q)$. The proposed contrastive loss approaches zero. Then we can refer to the convergence proof of the DDPG method.

V. SIMULATION RESULTS

Our proposed algorithms are tested at six Pybullet environments (Ant, HalfCheetah, InvDoublePen, InvPenSwingup, Hopper, Walker2d). Six benchmarks are from Maxim Lapan [46], involving two popular online methods (A2C, PPO) and four popular offline methods (SAC, DDPG, D4PG, TD3). Implementation details of benchmarks can be found at [46]. To match the basic settings of benchmarks, the actor and critic networks comprise two fully connected neural network layers with 400 and 300 hidden units. Here, the 300 hidden-unit layer indicates the logic layer. For simplification, we decrease the dimension from 300 to 20 with another mean operator layer, which calculates the mean value of every 15 dimensions. The sensitive analysis of these hyper-parameters will be discussed later. The experience replay buffer and batch size are 50,000 and 128, respectively. We set the number of keys to 4 times. Thus, in instance-actor and instance-critic methods, $b_2 = 4 * 128$. Based on our experiments, simply increasing the batch size from 128 to $4 * 128$ in the DDPG method can not boost the performance of received rewards and convergence, which matches the conclusion of [50]. In parallel-actor and parallel-critic methods, 4 extra networks are trained, and $b_2 = 128$.The sensitive analysis of the number of keys will be discussed later. The Adam optimizer is used for both actor and critic networks. The ReLU activation operator is used in all layers. The learning rate is 0.001.

The figure 6 shows the received mean rewards of five proposed CDRL methods in the training process of four environments. The total training time steps are $2e^6$. To plot the figure, we run the testing every $1e^4$ time steps. The combined-all CDRL algorithm means the combination of instance-actor, parallel-actor, instance-critic and parallel-critic CDRL methods. These curves clearly show that the combined-all CDRL method receives the maximum mean rewards and highest convergency speed in these four environments. Parallel-critic CDRL is better than parallel-actor CDRL in received rewards and convergency speed, since inaccurate policy estimations in the training process of parallel-actor CDRL may

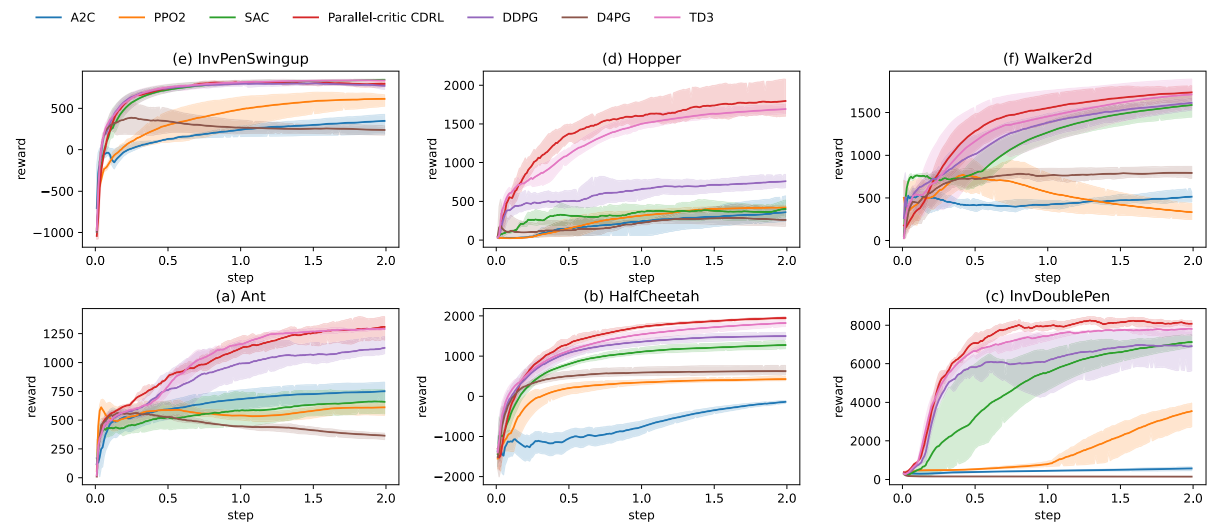


FIGURE 7. Performance comparisons among Parallel-critic CDRL and other benchmarks in six environments.

TABLE 3. The maximum mean rewards of Parallel-critic CDRL and other benchmarks in six environments.

Environment	A2C [16]	PPO [24]	SAC [25]	Parallel-critic CDRL	DDPG [21]	D4PG [22]	TD3 [23]
Ant	828.82	681.68	765.86	1397.50	1224.19	608.54	1302.70
HalfCheetah	-106.08	505.52	1398.56	1981.32	1591.40	768.47	1881.01
InvDoublePen	2546.86	3956.51	7503.41	8138.85	7808.91	2363.96	8008.10
InvPenSwingup	425.97	694.72	823.66	842.49	824.24	584.58	839.43
Hopper	569.77	449.98	522.05	2049.36	845.89	526.52	1632.63
Walker2D	618.37	965.43	1448.18	1759.25	1529.33	868.56	1666.12

introduce extra errors. In the parallel-critic CDRL method, the y values are less biased. With the assistance of a parallel-learning framework, parallel-critic and parallel-actor receive larger maximum mean rewards and convergence speeds than instance-critic and instance-actor methods. The convergence of the instance-actor CDRL method is a little bit worse than the others since the instance-actor CDRL utilizes the target actor network to generate positive/ negative keys, and the target actor network is conservative. Table 2 shows the maximum mean rewards of five proposed CDRL methods in four environments. For example, in the Ant environment, Combined-all CDRL is 3.04% larger than parallel-critic CDRL, 7.27% larger than parallel-actor CDRL, 10.38% larger than instance-actor CDRL, and 11.99% larger than instance-critic CDRL in received maximum mean rewards respectively. Similarly, in the HalfCheetah environment, parallel-critic CDRL is 5.04% larger than parallel-actor CDRL, 7.25% larger than instance-critic CDRL, and 12.75% larger than instance-actor CDRL in received maximum mean rewards respectively.

The figure 7 shows the received mean rewards of five proposed CDRL methods in the training process of four environments. The total training time steps are $2e^6$. To plot the figure, we run ten times of testing every $1e^4$ time steps. Typically, online methods require more training steps than offline methods. But $2e^6$ is sufficient for the training purposes of offline methods. These curves clearly show that the parallel-critic CDRL method receives the maximum mean rewards and highest convergency speed among six benchmarks in these six environments. Table 3 shows the maximum mean rewards of the proposed parallel-critic CDRL method and six other benchmarks in six environments. For example, in the HalfCheetah environment, parallel-critic CDRL is 5.33% larger than TD3, 24.50% larger than DDPG, and 41.67% larger than SAC in maximum mean rewards, respectively. In the Walker2D environment, parallel-critic CDRL is 5.59% larger than TD3, 15.03% larger than DDPG, and 21.48% larger than SAC in maximum mean rewards, respectively. In the InvDoublePen environment, our proposed parallel-critic CDRL method converges around $7e^5$ time steps, which is faster than the other benchmarks.

As mentioned, implementation details of benchmarks can be found at [46]. For fairness, we set the hidden units as 300 and then compress the dimension to 20 with the mean operator. Figure 8 shows the sensitive analysis of hidden units in the HalfCheetah environment of the parallel-critic CDRL method. Decreasing the number of hidden units from 300 to 100 causes the maximum mean rewards to drop from 2010.44 to 1833.28, while increasing the dimension from 20 to 40 leads to the maximum mean rewards rising from 2010.44 to 2163.32. In conclusion, reasonably higher dimension hidden units involve more contrastive information. However, high dimensions, like increasing the dimension from 20 to 100, will increase the computation cost hugely, which is not recommended. Figure 9 shows the

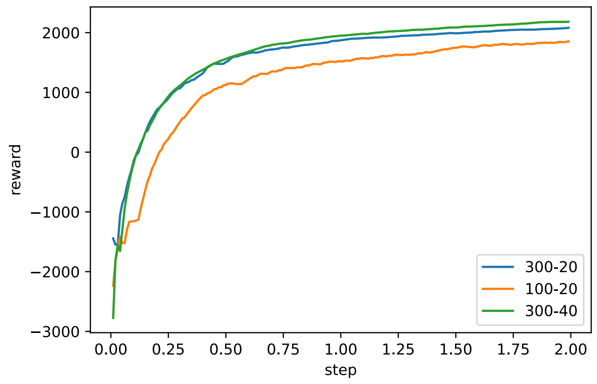


FIGURE 8. Sensitivity analysis of hidden units in HalfCheetah environment.

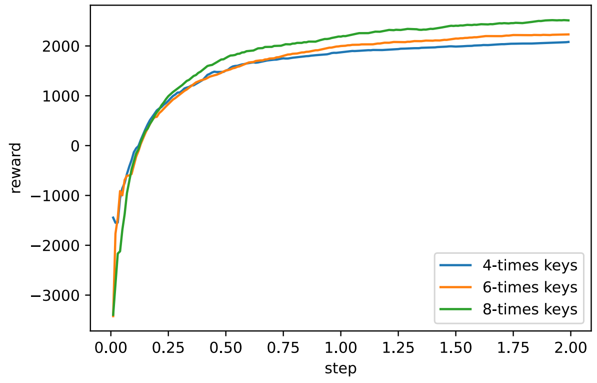


FIGURE 9. Sensitivity analysis of the total number of keys in HalfCheetah environment.

sensitive analysis of the total number of keys in the HalfCheetah environment. The received maximum mean reward of parallel-critic CDRL with 8 times keys is 2321.52, 6.59% larger than the method with 6 times keys, and 15.47% larger than the method with 4 times keys.

The advantages of the proposed contrast deep reinforcement learning algorithm can be summarized below.

- Firstly, the mean maximum rewards of CDRL match or are vastly better than the state-of-art algorithms. Besides, the convergence speed of CDRL matches or is higher than the others, and the standard deviation of the mean rewards of DRL matches or is less than the others.
- Secondly, with the help of the contrast learning structure, CDRL balances and learns from different-age experiences. By utilizing online experiences, CDRL can overcome the shortcomings of off-policy algorithms.
- Thirdly, unlike most parallel learning frameworks, CDRL is more data-efficient and resource-saving without parallel execution of environment instances.
- Lastly, positive and negative keys provide extra information to assist the training process of DRL.

VI. CONCLUSION

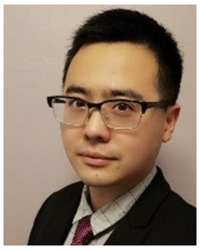
The integration of deep learning and reinforcement learning brings great success. The success of the experience replay buffer strategy and parallel learning strategy has been proven recently. The experience replay buffer stores data generated by elder policies that differ from the current policy. The state-action distributions shift leads to bootstrap error. If the most recently experienced trajectories in the buffer keep being sampled, off-policy algorithms will be degraded to online policy algorithms. The issue of balancing different-age experiences is challenging. The parallel learning strategy can regularize policies via different experiences from parallel environmental instances. However, these parallel environmental instances require higher simulation or physical costs. This paper introduces contrast learning into deep reinforcement learning with four different contrast learning methods: instance-actor, parallel-actor, instance-critic, and parallel-critic methods. Negative and positive keys provide extra information about different-age experiences. Our proposed algorithms can have the advantages of both online and offline policies. Besides, a new contrast loss is proposed based on the quality of positive/ negative keys. Experiments prove that our proposed algorithm can match or perform better than the state-of-the-art DRL algorithms with six environments.

In the future, we will extend our contrast learning method to multi-agent reinforcement learning algorithms. Compared with the single-agent system, multi-agent deep reinforcement learning is more complicated. Besides, the contrast learning among different agents will bring many advancements. Moreover, the graph convolutional neural network [51] is a promising alternative to the contrast learning method.

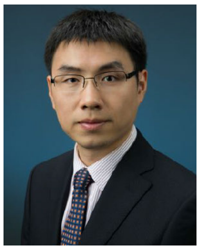
REFERENCES

[1] A. Amanlou, A. A. Suratgar, J. Tavoosi, A. Mohammadzadeh, and A. Mosavi, "Single-image reflection removal using deep learning: A systematic review," *IEEE Access*, vol. 10, pp. 29937–29953, 2022.
[2] Z. Shahbazi and Y.-C. Byun, "Machine learning-based analysis of cryptocurrency market financial risk management," *IEEE Access*, vol. 10, pp. 37848–37856, 2022.
[3] T. Sun, D. Huang, and J. Yu, "Market making strategy optimization via deep reinforcement learning," *IEEE Access*, vol. 10, pp. 9085–9093, 2022.
[4] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Reinforcement learning-based cascade motion policy design for robust 3D bipedal locomotion," *IEEE Access*, vol. 10, pp. 20135–20148, 2022.
[5] M. Rothmann and M. Porrmann, "A survey of domain-specific architectures for reinforcement learning," *IEEE Access*, vol. 10, pp. 13753–13767, 2022.
[6] D. Wang, M. Hu, and Y. Gao, "Multi-criteria mission planning for a solar-powered multi-robot system," in *Proc. Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, Aug. 2018, p. V02AT03A026.
[7] D. Wang, M. Hu, and J. D. Weir, "Simultaneous task and energy planning using deep reinforcement learning," *Inf. Sci.*, vol. 607, pp. 931–946, Aug. 2022.
[8] L. Yun, D. Wang, and L. Li, "Explainable multi-agent deep reinforcement learning for real-time demand response towards sustainable manufacturing," *Appl. Energy*, vol. 347, Oct. 2023, Art. no. 121324.
[9] J. Khalid, M. A. M. Ramli, M. S. Khan, and T. Hidayat, "Efficient load frequency control of renewable integrated power system: A twin delayed DDPG-based deep reinforcement learning approach," *IEEE Access*, vol. 10, pp. 51561–51574, 2022.
[10] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble," in *Proc. Conf. Robot Learn.*, 2022, pp. 1702–1712.
[11] C. Kaplanis, C. Clopath, and M. Shanahan, "Continual reinforcement learning with multi-timescale replay," 2020, *arXiv:2004.07530*.
[12] S. Zhang and R. S. Sutton, "A deeper look at experience replay," 2017, *arXiv:1712.01275*.
[13] G. Novati and P. Koumoutsakos, "Remember and forget for experience replay," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4851–4860.

[14] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," 2022, *arXiv:2208.06193*.
[15] D. Wang and M. Hu, "Deep deterministic policy gradient with compatible critic network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 4332–4344, Aug. 2023.
[16] V. Mnih, Ad. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[17] A. V. Clemente, H. N. Castejón, and A. Chandra, "Efficient parallel methods for deep reinforcement learning," 2017, *arXiv:1705.04862*.
[18] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 1–10, Jul. 2015.
[19] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 1735–1742.
[20] O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord, "Data-efficient image recognition with contrastive predictive coding," 2019, *arXiv:1905.09272*.
[21] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[22] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, T. B. Dhruva, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," 2018, *arXiv:1804.08617*.
[23] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*.
[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, *arXiv:1801.01290*.
[26] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[27] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5279–5288.
[28] C. C. Millán-Arias, B. J. T. Fernandes, F. Cruz, R. Dazeley, and S. Fernandes, "A robust approach for continuous interactive actor-critic algorithms," *IEEE Access*, vol. 9, pp. 104242–104260, 2021.
[29] D. Wang, "Meta reinforcement learning with Hebbian learning," in *Proc. IEEE 13th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2022, pp. 0052–0058.
[30] S.-H. Kong, I. M. A. Nahrendra, and D.-H. Paek, "Enhanced off-policy reinforcement learning with focused experience replay," *IEEE Access*, vol. 9, pp. 93152–93164, 2021.
[31] S. Li, O. Li, G. Liu, S. Ding, and Y. Bai, "Trajectory based prioritized double experience buffer for sample-efficient policy optimization," *IEEE Access*, vol. 9, pp. 101424–101432, 2021.
[32] D. Yang, X. Qin, X. Xu, C. Li, and G. Wei, "Sample efficient reinforcement learning method via high efficient episodic memory," *IEEE Access*, vol. 8, pp. 129274–129284, 2020.
[33] T. M. Luu and C. D. Yoo, "Hindsight goal ranking on replay buffer for sparse reward environment," *IEEE Access*, vol. 9, pp. 51996–52007, 2021.
[34] S. Schmitt, M. Hessel, and K. Simonyan, "Off-policy actor-critic with shared experience replay," 2019, *arXiv:1909.11583*.
[35] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu, and D. Silver, "Massively parallel methods for deep reinforcement learning," 2015, *arXiv:1507.04296*.
[36] W. Meng, Q. Zheng, L. Yang, P. Li, and G. Pan, "Qualitative measurements of policy discrepancy for return-based deep Q-network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4374–4380, Oct. 2020.
[37] M. Grounds and D. Kudenko, "Parallel reinforcement learning with linear function approximation," in *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*. Berlin, Germany: Springer, 2005, pp. 60–74.
[38] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," 2018, *arXiv:1803.00933*.
[39] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2019, *arXiv:1911.05722*.
[40] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3733–3742.
[41] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 297–304.
[42] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," 2019, *arXiv:1906.05849*.
[43] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2005, pp. 539–546.
[44] A. Srinivas, M. Laskin, and P. Abbeel, "CURL: Contrastive unsupervised representations for reinforcement learning," 2020, *arXiv:2004.04136*.
[45] J. Zhu, Y. Xia, L. Wu, J. Deng, W. Zhou, T. Qin, T.-Y. Liu, and H. Li, "Masked contrastive representation learning for reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3421–3433, Mar. 2023.
[46] M. Lapan, *Deep Reinforcement Learning Hands-On: Apply Modern RL Methods, With Deep Q-Networks, Value Iteration, Policy Gradients, TRPO, AlphaGo Zero and More*. Birmingham, U.K.: Packt Publishing Ltd, 2018.
[47] Z. Zhang, Z. Pan, and M. J. Kochenderfer, "Weighted double Q-learning," in *Proc. Twenty-Sixth Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3455–3461.
[48] D. Wang, "Reinforcement learning for combinatorial optimization," in *Encyclopedia of Data Science and Machine Learning*. Hershey, PA, USA: IGI Global, 2023, pp. 2857–2871.
[49] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning," in *Proc. AAAI*, 1993, pp. 99–107.
[50] F. He, T. Liu, and D. Tao, "Control batch size and learning rate to generalize well: Theoretical and empirical evidence," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.
[51] J. Skarding, B. Gabrys, and K. Musial, "Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey," *IEEE Access*, vol. 9, pp. 79143–79168, 2021.



DI WANG (Member, IEEE) received the B.S. degree in electrical engineering from Fuzhou University, China, in 2014, and the M.S. degree in electrical engineering from Tianjin University, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL, USA. His current research interests include disassembly planning, multi-agent systems, intelligent control, and energy schedules in the smart city.



MENGQI HU (Member, IEEE) received the Ph.D. degree in industrial engineering from Arizona State University, Tempe, AZ, USA, in 2012. He is currently an Assistant Professor with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago. He has published more than 30 journal articles, such as the IEEE Transactions on Evolutionary Computation, *Information Sciences*, and *Applied Energy*. His current research interests include multi-agent decision-making and reinforcement learning with applications in autonomous vehicles and smart grids. He serves as an Associate Editor for *Swarm and Evolutionary Computation* journal.