SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2024, Vol. 67, Iss. 8, 182203:1–182203:15
https://doi.org/10.1007/s11432-023-4016-0

Skill enhancement learning with knowledge distillation

Naijun LIU, Fuchun SUN*, Bin FANG & Huaping LIU

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, *China*

Received 28 May 2023/Revised 14 September 2023/Accepted 4 December 2023/Published online 22 July 2024

Abstract Skill learning through reinforcement learning has significantly progressed in recent years. However, it often struggles to efficiently find optimal or near-optimal policies due to the inherent trial-and-error exploration in reinforcement learning. Although algorithms have been proposed to enhance skill learning efficacy, there is still much room for improvement in terms of skill learning performance and training stability. In this paper, we propose an algorithm called skill enhancement learning with knowledge distillation (SELKD), which integrates multiple actors and multiple critics for skill learning. SELKD employs knowledge distillation to establish a mutual learning mechanism among actors. To mitigate critic overestimation bias, we introduce a novel target value calculation method. We also perform theoretical analysis to ensure the convergence of SELKD. Finally, experiments are conducted on several continuous control tasks, illustrating the effectiveness of the proposed algorithm.

Keywords skill learning, enhancement learning, reinforcement learning, knowledge distillation

1 Introduction

Skill learning through reinforcement learning (RL) [1], especially deep reinforcement learning [2, 3], has achieved exciting results in various fields such as video games [4], navigation [5], door opening [6], block-stacking [7], grasping [8], and some other complex manipulation tasks [9, 10]. While RL algorithms have demonstrated success in various applications, they often struggle to efficiently discover optimal or near-optimal policies. Achieving high-performing skills and stable training still remains challenging due to the inherent trial-and-error exploration in RL. Recently, this has become a critical issue in the skill-learning research community.

To address these challenges, scholars have proposed several common methods. One promising approach is maximum entropy reinforcement learning [11], which maximizes the policy entropy to promote policy exploration, thus improving the skill performance and training stability [12]. Maximizing entropy is a common regularization technique used in reinforcement learning. Another simple yet effective method is the reinforcement and imitation learning method [13], in which the skill policy is initialized via imitation learning with expert demonstrations and then further trained through reinforcement learning. Extensive research has been devoted to Q-learning and actor-critic-based reinforcement learning paradigms, aiming to improve skill training performance by mitigating critic overestimation bias. For instance, various approaches such as double Q-learning techniques [14, 15], using two critic approximators [12, 16], or employing an ensemble of critics [17, 18] have been utilized. Policy distillation [19] is another popular method in the teacher-student learning paradigm, wherein the knowledge from a teacher model is distilled into a student model. Unfortunately, many skill-learning domains lack strong teacher models or demonstrations. Thus, the learned skill performance and the training stability of methods like imitation learning or policy distillation are often limited by the capabilities of the teacher model and the quality of demonstrations.

Mutual learning is a concept rooted in human cognitive psychology [20], which entails a collective effort amongst students to collaborate and discover solutions. In contrast to the conventional teacher-student

* Corresponding author (email: fcsun@tsinghua.edu.cn)

framework, where students passively acquire knowledge from teachers, mutual learning necessitates intellectual collaboration from diverse perspectives [21,22]. Motivated by this concept, our research endeavors to replicate the human capacity to learn from one another in the skill-learning field. We investigate the potential of mutual learning in skill enhancement learning without the need for teacher models.

In this paper, we focus on improving skill learning performance and training stability in the case of unavailable teacher models. We propose skill enhancement learning with knowledge distillation (SELKD), which includes multiple actors and multiple critics. A mutual learning mechanism is established through knowledge distillation among the actors. To introduce diversity, each actor and critic is randomly initialized. During the skill training process, each actor is trained by optimizing its RL objective and extracting beneficial knowledge from the best-performing peer actor through knowledge distillation.

Acknowledging the inherent critic overestimation bias, we draw inspiration from [23] and introduce a new target value calculation mechanism called in-target max-min to mitigate this issue in SELKD. To generate an accurate target value, in-target max-min selects a random subset of critics to perform a minimum operation, followed by a maximum operation over all actors. The critic is then trained based on the resulting target value. However, our experimental results and conclusions differ from the conclusion reported in [23].

Our proposed method is additionally inspired by the policy distillation method [19]. However, unlike policy distillation, SELKD does not rely on teacher models to guide policy learning but rather enhances policy performance by mutual learning. SELKD shares similarities with the dual policy distillation (DPD) method [24] concerning the absence of teacher models. However, significant differences exist as all critics in SELKD are shared among all actors, unlike the actor-critic appearing in pairs in DPD. Furthermore, we design a unique method for calculating target values in SELKD to alleviate critic overestimation bias.

This paper makes the following contributions.

• We propose the SELKD algorithm, which integrates multiple actors and multiple critics to form a mutual learning mechanism through knowledge distillation among the actors. In addition, an ensemble of critics is utilized to guide the updated direction for the actors. To the best of our knowledge, this is the first work that utilizes mutual learning through knowledge distillation to enhance skill learning performance and improve training stability.

• To overcome the common issue of the critic overestimation bias in RL, we propose an improved target value calculation mechanism called in-target max-min. This mechanism selects a random subset of critics to perform a minimum operation, followed by a maximum operation over all actors to obtain the target value. The critic is then trained based on the resultant target value. In-target max-min mechanism effectively alleviates the critic overestimation bias in SELKD.

• We provide theoretical proof for the convergence of our algorithm and experimentally validate its effectiveness through a series of experiments. Our results illustrate that our proposed algorithm significantly outperforms its counterparts regarding skill performance and training stability. Furthermore, a complete set of ablation studies verifies the validity of each proposed component in our algorithm.

The rest of this paper is organized as follows. The related work is discussed in Section 2. The preliminaries required for developing our proposed algorithm are summarized in Section 3. The details of the SELKD algorithm and the convergence analysis of SELKD are provided in Sections 4 and 5, respectively. The experiments are introduced in Section 6. The discussion is given in Section 7. Finally, Section 8 concludes this paper.

2 Related work

Numerous approaches have been proposed to improve the skill learning performance or training stability in reinforcement learning. One such type of method is entropy-based reinforcement learning, including maximum entropy reinforcement learning [25], G-learning [26], and soft actor-critic (SAC) [12], which encourages policy exploration in the learning process. Additionally, the maximizing entropy mechanism exhibits the property of stabilizing the policy training process [12]. Currently, the maximum entropy is a widely used regularization technique in reinforcement learning.

Another type of method involves utilizing demonstration data. In off-policy reinforcement learning, demonstration data are stored in a replay buffer and then sampled during policy learning [27]. Demonstrations are also utilized for initial policy pre-training through behavior cloning [28]. Additionally, some studies modify the distribution of initial states to align them with demonstration states [13, 29].

Nonetheless, acquiring high-quality expert demonstrations is a costly endeavor. We explore enhancing skill learning performance without the need for demonstrations.

Addressing the issue of critic overestimation bias in Q-learning or actor-critic-based reinforcement learning is crucial for improving skill learning performance and training stability [30]. Double Q-learning [14] and double deep Q-network (double DQN) [15] address this bias by utilizing two critic functions instead of one, with each function used to independently estimate the value of the critic. Twin delayed deep deterministic policy gradient (TD3) [16] and SAC [12] further mitigate the overestimation bias by evaluating the minimum value between two critics. Maxmin Q-learning [17] involves setting N critic functions and selecting their minimum value. Meanwhile, randomized ensembled double Q-learning (REDQ) [18] achieves high performance by utilizing an ensemble of critic functions and performing in-target minimization with a randomly selected subset of them. Our method also suffers from the critic overestimation bias. However, the integration of multiple actors and multiple critics in our algorithm limits the applicability and practicality of existing methods mentioned above for addressing this bias. Consequently, we propose a novel target value calculation method called in-target max-min to alleviate the issue of critic overestimation bias.

Our work is also related to another class of methods called policy distillation [19, 31], which forms a typical teacher-student learning framework to enhance the learned policy performance by continuously learning from a teacher model in the policy training process. Since good teacher policy models are usually not available, some works explore distilling learning from non-perfect teacher models, which can be regarded as a student-student learning framework, such as Dual Policy Distillation [24], P2PDRL [32]. In addition, some work also explores learning from a set of peers [23], which forms a student-students learning framework similar to mutual learning. Our approach is similar to DPD [24] in that we do not use a teacher model, but it differs in that our method integrates multiple actors that exceed two, as we discovered through experimentation that two actors are not sufficient for mutual learning. Another notable difference from DPD is that our method employs multiple critics, with each actor associated with an ensemble of critics. Unlike [23, 24], the actors and critics in our algorithm are unpaired.

Our approach is also relevant to ensemble reinforcement learning. Previous studies have explored the use of ensemble critics to improve skill performance. For instance, Average-DQN [33] reduces overestimation bias by averaging multiple critic functions, while random ensemble mixture (REM) [34] randomly selects a subset of critic functions. Maxmin Q-learning selects the minimum critic value among all critic functions, while SUNRISE [35] reweights target critic value based on uncertainty estimates from an ensemble of critics. Aggressive Q-learning with ensembles (AQE) [36] averages all the critic functions, excluding the topmost critic function, to obtain the target value function. In contrast, other studies have focused on using an ensemble of actors to enhance the trained skill performance. Ensemble proximal policy optimization (EPPO) [37] derives an ensemble policy by averaging the outputs from sub-policies. Automatic model selection using validation temporal-difference (TD) error (AVTD) method [38] trains several actors, where each actor applies a different regularization strategy and dynamically selects the action with the smallest validation TD error. Determinantal point process-based neural network sampler (DNS) [39] samples a subset of neural networks for backpropagation at every training step. Actor-critic ensemble [40] uses a critic ensemble to select the best action from proposals of multiple actors. In contrast, our approach involves ensembling multiple actors and multiple critics, where all actors perform additional mutual learning based on knowledge distillation.

In addition, although our proposed SELKD algorithm has multiple actors like the multi-agent reinforcement learning (MARL) algorithm [41, 42], SELKD has fundamental differences from MARL. In SELKD, the state of each actor is independent and unrelated to the states of other actors, whereas, in MARL, the state of each agent is influenced by the other agents. Additionally, the state transition function in SELKD solely considers the current actor's state and action, while in MARL, it also incorporates the states and actions of other agents.

3 Preliminaries

In this section, we introduce traditional RL, SAC, REDQ, and the notations used in this paper.

3.1 Traditional RL

Consider a finite-horizon Markov decision process (MDP), defined by a tuple $M = \{\mathcal{S}, \mathcal{A}, p, p_0, \gamma, r\}$, where $\mathcal{S} = \{s\}$ is the space of the environment states, and $\mathcal{A} = \{a\}$ is the action space. $\gamma \in (0, 1]$ is a discount factor. $p(s'|s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the dynamics model, also called the state transition function, which denotes the probability of an agent moving to state s' after taking action a in state s. $p_0 : \mathcal{S} \to \mathbb{R}$ represents the distribution of the initial state s_0, while $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ serves as the reward function. In off-policy RL, the transition tuples $\{(s, a, r, s')\}$ are stored into a replay buffer $\mathcal{D}$ and sampled to train the skill policy, thereby enabling the agent to learn from a broader set of experiences. The expectation of the cumulative reward obtained by a policy π can be defined as $\eta(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$. The objective of reinforcement learning is to maximize $\eta(\pi)$.

3.2 SAC

SAC [12] is an off-policy actor-critic method based on the maximum entropy reinforcement learning framework [11], which includes an actor (π_θ) and two critics ($Q_{\phi_j}, j = 1, 2$). The parameters of the actor and critics are updated through alternating soft policy evaluation and soft policy improvement. The parameter ϕ_j of each critic is updated by minimizing the following soft Bellman residual:

$$L_{\phi_j} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(Q_{\phi_j}(s, a) - y \right)^2 \right], \tag{1}$$

where $y = r + \min_{j=1,2} Q_{\phi_j}(s', a') - \alpha \log \pi_\theta(a'|s')$ is the target value, a' is the next action sampled from policy actor $\pi_\theta(\cdot|s')$. The parameter θ of the policy actor π_θ is updated by maximizing the following objective:

$$L_\theta = \mathbb{E}_{s\sim\mathcal{D}} \left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s) \right], \ \tilde{a}_\theta(s) \sim \pi_\theta(\cdot|s), \tag{2}$$

where α ($\alpha > 0$) is the temperature parameter that determines the relative importance of the entropy term compared to the reward, thereby controlling the stochasticity of the optimal policy.

3.3 REDQ

REDQ [18] is also an off-policy actor-critic method based on the max-min reinforcement learning framework. In REDQ, an ensemble of critic functions is used to model the critic. One key feature of REDQ is the in-target minimization, which samples a subset of critic functions to create the target value used for training the critic networks. The target value y is calculated as

$$y = r + \gamma \left(\min_{j\in\mathcal{M}} Q_{\phi_{\text{targ},j}}\left(s', \tilde{a}'\right) - \alpha \log \pi_\theta\left(\tilde{a}' \mid s'\right) \right), \tag{3}$$

where $\mathcal{M}$ represents a subset of critic functions. The parameters of each critic function ($\phi_j, j = 1, 2, \ldots, N$) are updated using a similar approach to (1). The gradient used to update policy parameter θ is defined as follows:

$$\nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{s\sim\mathcal{B}} \left(\frac{1}{N} \sum_{j=1}^{N} Q_{\phi_j}\left(s, \tilde{a}_\theta(s)\right) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s)|s\right) \right), \ \tilde{a}_\theta(s) \sim \pi_\theta(\cdot|s), \tag{4}$$

where N is the total number of critic functions, and $s \sim \mathcal{B}$ denotes that state s is sampled from a batch of transition tuples $\mathcal{B}$.

4 Method

In this section, we present our SELKD algorithm that enables skill enhancement learning in the policy training process. The overview of the SELKD algorithm is illustrated in Figure 1.

Our proposed algorithm SELKD is a novel model-free algorithm that can be combined with any standard off-policy algorithm, such as DDPG [43], TD3 [16], or SAC [12]. For the sake of concreteness, we build SELKD on SAC in this paper. An experimental study is also presented in Section 7 to discuss

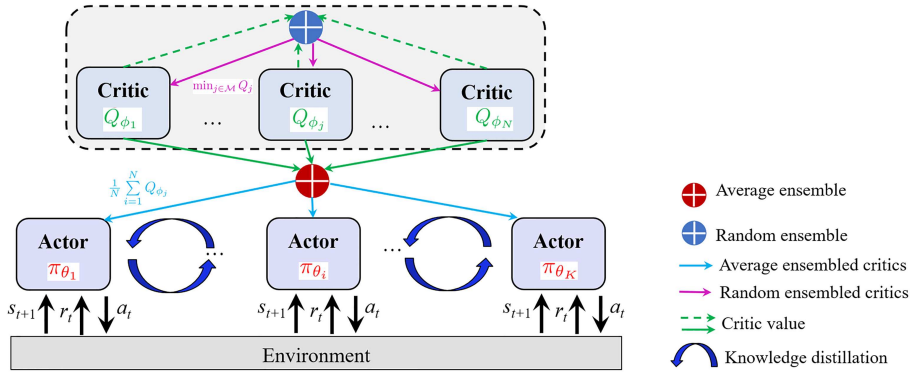

Figure 1 (Color online) Overview of the SELKD algorithm. SELKD integrates K actors $\pi_{\theta_i}, i = 1, 2, \ldots, K$, and N critics $Q_{\phi_j}, j = 1, 2, \ldots, N$. All actors interact with the same environment with different initializations. In the skill training process, each actor is trained by optimizing its own RL objective and extracting beneficial knowledge from the best-performing peer actor through knowledge distillation. Each critic is updated with a target value calculated with a random subset of critics, followed by a maximum operation over all actors.

the performance of the SELKD built on SAC and TD3. Formally, SELKD includes N critics $\{Q_{\phi_j}\}_{j=1}^{N}$, K actors $\{\pi_{\theta_i}\}_{i=1}^{K}$, and only one replay buffer $\mathcal{D}$.

The individual critics Q_{ϕ_j} $(j = 1, 2, \ldots, N)$ are randomly and independently initialized and updated by minimizing the Mean squared error (MSE) $J_{Q_{\phi_j}}$ as follows:

$$J_{Q_{\phi_j}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\frac{1}{2}(Q_{\phi_j}(s,a) - y)^2\right], \tag{5}$$

where y is the target value function given by

$$y = r + \gamma \max_{i=1,\ldots,K}\left(\min_{j\in\mathcal{M}} Q_{\phi_{\text{target},j}}(s', \tilde{a}_i') + \alpha \log(\pi_{\theta_i}(\cdot|s'))\right), \ \tilde{a}_i' \sim \pi_{\theta_i}(\cdot|s'). \tag{6}$$

To reduce overestimation bias, similar to REDQ [18], the target value y is calculated with a minimization over a random subset $\mathcal{M}$ of the N critics and followed by a maximum operation over all actors $\{\pi_{\theta_i}\}_{i=1}^{K}$. The size of the subset $\mathcal{M}$ is M and kept fixed. Our proposed target calculation method is an extension of the target calculation methods used in SAC [12], Maxmin Q-learning [17], and REDQ [18]. Specifically, if $K = 1$, our method reduces to REDQ, while if $K = 1, M = N$, it reduces to Maxmin Q-learning, and for $K = 1, M = N = 2$, it reduces to SAC. In our algorithm, $K \geqslant 2$ offers an additional benefit in establishing a mutual learning mechanism among actors.

The individual actors π_{θ_i} $(i = 1, 2, \ldots, K)$ are randomly and independently initialized, and optimized by maximizing the expected return and the entropy of the policy actor. This optimization is mathematically represented as

$$\max_{\theta_i} \mathbb{E}_{s\sim\mathcal{D}}\left(\frac{1}{N}\sum_{i=1}^{N} Q_{\phi_j}(s, \tilde{a}_{\theta_i}(s)) - \alpha \log \pi_{\theta_i}(\tilde{a}_{\theta_i}(s)|s)\right), \ \tilde{a}_{\theta_i}(s) \sim \pi_{\theta_i}(\cdot|s). \tag{7}$$

To distill knowledge from the best-performing actor to the other peer actors, we introduce an additional objective $J_{\pi_{\theta_i}}$ expressed as follows:

$$J_{\pi_{\theta_i}} = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\text{KL}}\left(\pi_{\theta_i}(\cdot|s)||\pi_{\hat{\theta}_i}(\cdot|s)\right)\right], \tag{8}$$

where $\pi_{\hat{\theta}}$ represents the best-performing actor, which is estimated using Monte Carlo (MC) rollouts. $D_{\text{KL}}(\pi_{\theta_i}||\pi_{\hat{\theta}_i})$ is the KL divergence between the distribution of π_{θ_i} and $\pi_{\hat{\theta}_i}$.

Consequently, the overall objective function for optimizing actor π_{θ_i} is given by

$$\max_{\theta_i} \mathbb{E}_{s\sim\mathcal{D}}\left(\frac{1}{N}\sum_{j=1}^{N} Q_{\phi_j}(s, \tilde{a}_{\theta_i}(s)) - \alpha \log \pi_{\theta_i}(\tilde{a}_{\theta_i}(s)|s) - J_{\pi_{\theta_i}}\right), \ \tilde{a}_{\theta_i}(s) \sim \pi_{\theta_i}(\cdot|s). \tag{9}$$

Algorithm 1 SELKD

Initialize K actors π_{θ_i} with parameters θ_i $(i = 1, \ldots, K)$, N critic Q_{θ_j} with parameters ϕ_j $(j = 1, \ldots, N)$, empty a replay buffer $\mathcal{D}$, set target parameters $\phi_{\text{target},j} \leftarrow \phi_j$ for $j = 1, 2, \ldots, N$.
repeat
  for $i = 1, \ldots, K$ **do**
    Take one action $a_t \sim \pi_{\theta_i}(\cdot|s_t)$, observe reward r_t, new state s_{t+1};
    Store transition tuple into replay buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$;
  end for
  for G updates **do**
    for $j = 1, \ldots, N$ **do**
      Sample a mini-batch $\mathcal{B} = \{(s, a, r, s')\}$ from $\mathcal{D}$;
      Sample a set $\mathcal{M}$ of M distinct indices from $1, 2, \ldots, N$;
      Compute the target y:

$$y = r + \gamma \max_{i=1,\ldots,K} \left(\min_{j \in \mathcal{M}} Q_{\phi_{\text{target},j}}(s', \tilde{a}'_i) + \alpha \log(\pi_{\theta_i}(\cdot|s')) \right), \ \tilde{a}'_i \sim \pi_{\theta_i}(\cdot|s');$$

      Update critic parameters ϕ_j with gradient descent using

$$\nabla_{\phi_j} \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s') \in \mathcal{B}} (Q_{\phi_j}(s, a) - y)^2;$$

      Update target networks with $\phi_{\text{target},j} \leftarrow \rho \phi_{\text{target},j} + (1 - \rho)\phi_j$;
    end for
  end for
  for $i = 1, \ldots, K$ **do**
    Sample a mini-batch $\mathcal{B} = \{(s, a, r, s')\}$;
    Compute distillation loss $J_{\pi_{\theta_i}}$:

$$J_{\pi_{\theta_i}} = \mathbb{E}_{s \sim \mathcal{D}} \left[D_{\text{KL}} \left(\pi_{\theta_i}(\cdot|s) || \pi_{\hat{\theta}_i}(\cdot|s) \right) \right],$$

    where $\pi_{\hat{\theta}_i}$ is the best-performing actor approximated with Monte Carlo return;
    Update actor parameters θ_i with gradient ascent using

$$\nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{s \sim \mathcal{B}} \left(\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_j}(s, \tilde{a}_{\theta_i}(s)) - \alpha \log \pi_{\theta_i}(\tilde{a}_{\theta_i}(s)|s) - J_{\pi_{\theta_i}} \right), \ \tilde{a}_{\theta_i}(s) \sim \pi_{\theta_i}(\cdot|s);$$

  end for
until Convergence;
return π_{θ_i}, for $i = 1, \ldots, K$.

The training procedure is detailed in Algorithm 1. SELKD algorithm begins with each actor π_{θ_i} $(i = 1, 2, \ldots, K)$ interacting with the environment to collect transition tuples $\{(s_t, a_t, r_t, s_{t+1})\}$. Each actor π_{θ_i} $(i = 1, 2, \ldots, K)$, at time-step t, receives a state s_t and outputs a normal distribution of actions corresponding to the state s_t. An action a_t is then sampled from this normal distribution: $a_t \sim \pi_{\theta_i}(\cdot|s_t)$. The sampled action a_t is then executed in the environment, resulting in the actor π_{θ_i} transitioning to a new state s_{t+1} and receiving a reward r_t. The transition tuples $\{(s_t, a_t, r_t, s_{t+1})\}$ are stored in the replay buffer $\mathcal{D}$. The replay buffer $\mathcal{D}$ is assigned a size of 1.0×10^5. When $\mathcal{D}$ reaches its maximum capacity, the oldest transition tuples in $\mathcal{D}$ are replaced with the most recently collected ones. In SELKD, all actors utilize a single, shared replay buffer $\mathcal{D}$. The experienced transition tuples from different actors are stored in the same replay buffer. This ensures the diversity of the off-policy training data and improves the sample efficiency of SELKD. The learning process starts by sampling a batch of transition tuples $\mathcal{B} = \{(s, a, r, s')\}$ from the replay buffer $\mathcal{D}$. The actors and critics are trained alternatively. The update-to-data (UTD) ratio [18] is set to be G. To train each critic Q_{ϕ_j} $(j = 1, 2, \ldots, N)$, the target value y is calculated using (6). All critics are optimized using the MSE error shown in (5). To train each actor $\pi_{\theta_i} (i = 1, 2, \ldots, K)$, the distillation loss is computed with (8). The actor parameters θ_i are then optimized via gradient ascent using (9). Finally, the SELKD algorithm converges. After training, the best-performing actor is selected to be executed in the environment.

5 Convergence analysis of SELKD

In this section, we show that SELKD is convergent in a finite tabular MDP setting. In this MDP setting, we maintain N tabular critics $Q^j, j = 1, 2, \ldots, N$. As different actors own the same action space $\mathcal{A}$, at

Algorithm 2 Tabular SELKD

Initialize $\{Q_j(s,a), s\in\mathcal{S}, a\in\mathcal{A}\}_{j=1}^{N}$;
repeat
  for $t=0,1,\ldots$ **do**
    Choose $a_t\in\mathcal{A}$ based on $\{Q^j(s_t,a_t)\}_{j=1}^{N}$, observe r_t, s_{t+1};
    Randomly choose a subset $\mathcal{M}$ of size M from $\{1,\ldots,N\}$;
    $y_t = r_t + \gamma \max_{a'\in\mathcal{A}} \min_{j\in\mathcal{M}} Q^j\left(s_{t+1},a'\right)$;
    for $j=1,\ldots,N$ **do**
      $Q^j(s_t,a_t)\leftarrow Q^j(s_t,a_t)+\alpha\left(y_t-Q^j(s_t,a_t)\right)$;
    end for
  end for
until Convergence.

each time-step t, we perform an update by setting target value y_t as follows:

$$y_t = r_t + \gamma \max_{a'\in\mathcal{A}} \min_{j\in\mathcal{M}} Q^j\left(s_{t+1},a'\right), \tag{10}$$

which is a simplified version of (6). The critic Q^j is updated with respect to target value y_t and learning rate $\alpha_t(s_t,a_t)$:

$$Q^j(s_t,a_t) = Q^j(s_t,a_t) + \alpha_t(s_t,a_t)(y_t - Q^j(s_t,a_t)). \tag{11}$$

The tabular SELKD algorithm is shown in Algorithm 2. For clarity, we use $G=1$.

We first introduce Lemma 1 and apply it to prove the convergence of the SELKD algorithm. The proof borrows from the proof of convergence of TD3 [16] and SARSA [44].

Lemma 1. Consider a stochastic process $\left(\zeta_t,\Delta_t,F_t\right), t\geqslant 0$ where $\zeta_t,\Delta_t,F_t: X\rightarrow\mathbb{R}$ satisfy the equation:

$$\Delta_{t+1}\left(x_t\right) = \left(1-\zeta_t\left(x_t\right)\right)\Delta_t\left(x_t\right) + \zeta_t\left(x_t\right)F_t\left(x_t\right),$$

where $x_t\in X$ and $t=0,1,2,\ldots$. Let P_t be a sequence of increasing σ-fields such that ζ_0 and Δ_0 are P_0-measurable, and ζ_t,Δ_t, and F_{t-1} are P_t-measurable, $t=1,2,\ldots$. Assume that the following holds:

(1) The set X is finite;

(2) $\zeta_t\left(x_t\right)\in[0,1], \sum_t \zeta_t\left(x_t\right)=\infty, \sum_t\left(\zeta_t\left(x_t\right)\right)^2<\infty$ with probability 1 and $\forall x\neq x_t: \zeta(x)=0$;

(3) $\left\|\mathbb{E}\left[F_t\mid P_t\right]\right\|\leqslant\kappa\left\|\Delta_t\right\|+c_t$ where $\|\cdot\|$ denotes the maximum norm, $\kappa\in[0,1)$, and c_t converges to 0 with probability 1;

(4) $\mathrm{Var}\left[F_t\left(x_t\right)\mid P_t\right]\leqslant C\left(1+\kappa\left\|\Delta_t\right\|\right)^2$, where C is some constant.

Then the sequence Δ_t converges to 0 with probability 1.

The proof of Lemma 1 can be found in [44].

Theorem 1 (Convergence of SELKD)**.** Consider a finite MDP and assume for all $s\in\mathcal{S}$ and $a\in\mathcal{A}$, $\gamma\in[0,1]$, $\mathrm{Var}[r_t(s,a)]<+\infty$, $\sum_{t=0}^{+\infty}\alpha_t(s,a)=+\infty$, $\sum_{t=0}^{+\infty}\alpha_t^2(s,a)<+\infty$ with $\alpha_t(s,a)\in[0,1]$. Then $\forall j\in[1,N]$, $Q^j(s,a)$ in SELKD learning converges to optimal value $Q^*(s,a)$, as defined by the Bellman optimality, with probability 1.

Proof. We apply Lemma 1 with $P_t=\{Q^1,\ldots,Q^N,s_0,a_0,\alpha_0,r_1,s_1,a_1,\ldots,s_t,a_t\}$, $X=\mathcal{S}\times\mathcal{A}$, $\Delta_t=Q^j(s_t,a_t)-Q^*(s_t,a_t)$, $\zeta_t=\alpha_t$. First, note that conditions (1), (2), and (4) of Lemma 1 hold with the conditions given by Theorem 1, respectively. Letting $Q_t^j(s_t,a_t)$ be the j-th critic at iteration update step t, we have

$$Q_{t+1}^j(s_t,a_t) = Q_t^j(s_t,a_t) + \alpha_t(s_t,a_t)(y_t - Q_t^j(s_t,a_t)),$$

where $y_t=\max_{a'}\min_{j\in\mathcal{M}}Q^j(s_{t+1},a')$. We can get

$$\begin{aligned}
Q_{t+1}^j(s_t,a_t)-Q^*(s_t,a_t) =& Q_t^j(s_t,a_t)-Q^*(s_t,a_t)+\alpha_t(s_t,a_t)\left(y_t-Q_t^j(s_t,a_t)+Q^*(s_t,a_t)-Q^*(s_t,a_t)\right),\\
\Delta_{t+1}(s_t,a_t) =& \Delta_t(s_t,a_t)+\alpha_t(s_t,a_t)\left(y_t-Q_t^j(s_t,a_t)+Q^*(s_t,a_t)-Q^*(s_t,a_t)\right)\\
=& \left(1-\alpha_t(s_t,a_t)\right)\Delta_t(s_t,a_t)+\alpha_t(s_t,a_t)\left(y_t-Q^*(s_t,a_t)\right)\\
=& \left(1-\alpha_t(s_t,a_t)\right)\Delta_t(s_t,a_t)+\alpha_t(s_t,a_t)F_t(s_t,s_t),
\end{aligned}$$

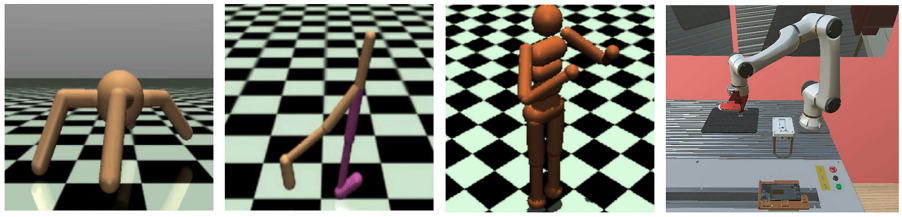

Figure 2 (Color online) OpenAI Gym tasks and a 3C assembly task that are used to benchmark our method. (a) Ant; (b) Walker2d; (c) Humanoid; (d) 3C assembly task.

where we have defined $F_t(s_t, a_t)$ as

$$\begin{aligned} F_t\left(s_t, a_t\right) &= y_t - Q^*(s_t, a_t) \\ &= r_t + \max_{a'} \min_{j \in \mathcal{M}} Q^j(s', a') - Q^*(s_t, a_t) \\ &= r_t + \max_{a'} \min_{j \in \mathcal{M}} Q^j(s', a') - Q^*(s_t, a_t) + \gamma Q^j(s_{t+1}, a^*) - \gamma Q^j(s_{t+1}, a^*) \\ &= \max_{a'} \min_{j \in \mathcal{M}} Q^j(s', a') - \gamma Q^j(s_{t+1}, a^*) + F_t^Q, \end{aligned}$$

and $F_t^Q = r_t + \gamma Q_t^j(s_{t+1}, a^*) - Q^*(s_t, a_t)$ denotes the value of F_t under standard Q-learning and $c_t = \max_{a'} \min_{j \in \mathcal{M}} Q^j(s', a') - Q^*(s_t, a_t) - \gamma Q^j(s_{t+1}, a^*)$. As $\mathbb{E}[F_t^Q \mid P_t] \leqslant \gamma \left\|\Delta_t\right\|$ is a well-known result, then condition (3) of Lemma 1 holds if it can be shown that c_t converges to 0 with probability 1. Let $y_t = \max_{a'} \min_{j \in \mathcal{M}} Q^j(s_t, a')$ and $\Delta_t^{ij}(s_t, a_t) = Q_t^i(s_t, a_t) - Q_t^j(s_t, a_t)$, where c_t converges to 0, if ${\Delta_t}^{ij}(s_t, a_t)$ converges to 0. We have

$$\begin{aligned} \Delta_{t+1}^{ij}(s_t, a_t) &= Q_t^i(s_t, a_t) + \alpha_t(s_t, a_t)(y_t - Q_t^i(s_t, a_t)) - Q_t^j(s_t, a_t) - \alpha_t(s_t, a_t)(y_t - Q_t^j(s_t, a_t)) \\ &= Q_t^i(s_t, a_t) - Q_t^j(s_t, a_t) - \alpha_t(s_t, a_t)(Q_t^i(s_t, a_t) - Q_t^j(s_t, a_t)) \\ &= (1 - \alpha_t(s_t, a_t))\Delta_t^{ij}(s_t, a_t). \end{aligned}$$

Clearly, Δ_t^{ij} will converge to 0, which demonstrates that we have satisfied condition 3 of Lemma 1, implying that for $\forall j \in [1, 2, \ldots, N], Q_t^j\left(s_t, a_t\right)$ converges to $Q^*\left(s_t, a_t\right)$, thus proving Theorem 1.

6 Experiments

The goal of our experimental evaluations is to assess the skill learning performance and the training stability of our proposed algorithm in comparison to prior learning algorithms on three challenging continuous control tasks from the OpenAI Gym benchmark [45], as well as a designed 3C assembly task, shown in Figure 2. For each of the three tasks from the OpenAI Gym, the agent must be trained to maximize accumulated rewards. In the case of the 3C assembly task, the agent must be trained to accurately place an FPC (flexible printed circuit) part at a desired assembly position, which is a necessary and common task in 3C assembly scenarios.

We compare our algorithm SELKD against SAC [12], REDQ [18], DPD [24], REM [34], Average-DQN [33], AQE [36], DNS [39], AVTD [38], and TASAC [23]. For a fair comparison, we reimplemented all the baseline algorithms within the same code framework, where all network architectures, optimizers, batch size, replay buffer size, discount factor, and learning rate settings remained identical. A comparison of hyperparameter settings among SELKD and other baseline algorithms is summarized in Table 1. For example, SEKD is implemented with default parameters of $G = 20$, $N = 10$, $M = 2$, and $K = 5$ for all environments. REDQ uses $G = 20$, $N = 10$, and $M = 2$. In addition, implementing the unique component(s) of each baseline algorithm is based on the setting in the relevant paper.

Specifically, after each training epoch, we conducted ten episodes using the current policy actor(s) and recorded the performance as the undiscounted sum of rewards for the OpenAI Gym tasks, and success rates for the 3C assembly task.

Table 1 Comparison of hyperparameter settings among SELKD and other baseline algorithms

Hyperparameters	SELKD(Our)	SAC	REDQ	DPD	Average-DQN	REM	AVTD	AQE	DNS	TASAC
learning rate (α)	3.0×10^{-4}									
Discount factor (γ)	0.99									
Optimizer	Adam [46]									
Mini-batch size ($\lvert\mathcal{B}\rvert$)	256									
Replay buffer size ($\mathcal{D}$)	1.0×10^5									
Hidden layers of critic	(256, 256)									
Hidden layers of actor	(256, 256)									
UTD ratio (G)	20	1	20	20	20	20	20	20	20	20
Number of critic(s) (N)	10	2	10	4	10	10	10	10	10	2
Size of subset of critics (M)	2	–	2	–	–	–	–	–	–	–
Number of actor(s) (K)	5	1	1	2	1	1	1	1	1	2

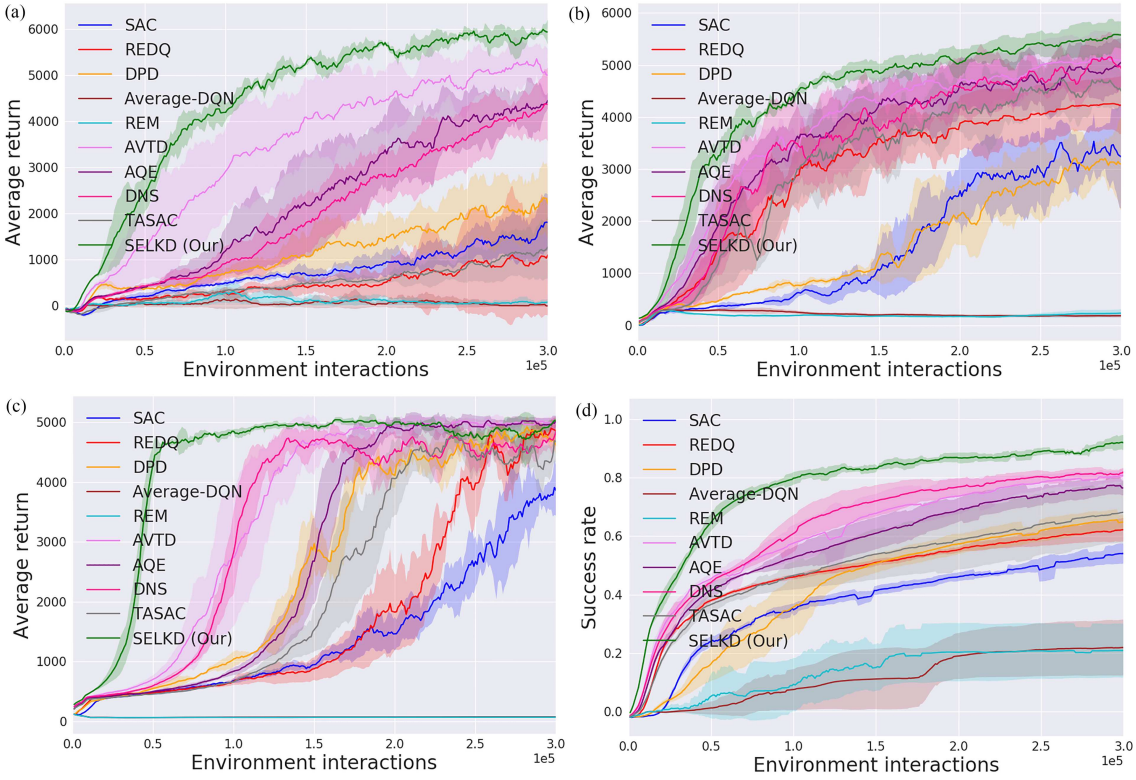

Figure 3 (Color online) Learning curves for the OpenAI Gym tasks and the 3C assembly task. The shaded region represents half a standard deviation of the average evaluation over five trials. SELKD outperforms all the baseline algorithms in all test tasks. (a) Ant; (b) Walker2d; (c) Humanoid; (d) 3C assembly task.

6.1 Comparative evaluation

Figure 3 shows the learning curves for SAC, REDQ, DPD, Average-DQN, REM, AQE, DNS, AVTD, TASAC, and our proposed algorithm SELKD. We train five instances of each algorithm with different random seeds, conducting ten evaluation trials every 1000 environment interaction time-steps. The solid curves correspond to the mean values, and the shaded region represents the standard derivations over the five instances. The final trained skill performance on all test tasks is summarized in Table 2.

The results show that, overall, SELKD outperforms the baseline algorithms across all test tasks in terms of learning speed and final performance. In particular, the REM and Average-DQN algorithms are ineffective in learning skill policies for the Ant, Walker2d, and Humanoid tasks. The success rate of skill policies learned with REM and Average-DQN on the 3C assembly task is less than 30%.

The standard deviation of learning curves for multiple random seeds throughout the training process is utilized to evaluate the skill training stability. For clarity, we also plot the standard deviation of the

Table 2 Average return (Ant, Walker2d, Humanoid) and success rate (3C assembly task) over 10 trials at 0.3 million environment interaction time-steps[a)]

Method	Ant	Walker2d	Humanoid	3C assembly task
SAC [12]	1800.26 ± 197.11	2857.03 ± 449.66	4049.53 ± 235.64	0.54 ± 0.03
REDQ [18]	1271.40 ± 445.76	4155.59 ± 674.79	5014.75 ± 290.80	0.62 ± 0.03
DPD [24]	2641.93 ± 429.44	2775.84 ± 409.08	4613.84 ± 213.01	0.65 ± 0.05
Average-DQN [33]	42.54 ± 104.66	187.20 ± 35.06	73.75 ± 4.00	0.22 ± 0.09
REM [34]	204.32 ± 106.47	230.99 ± 33.94	68.93 ± 3.89	0.21 ± 0.10
AVTD [38]	4278.17 ± 771.89	5165.20 ± 501.53	5021.68 ± 329.81	0.81 ± 0.04
AQE [36]	4601.68 ± 618.97	5078.57 ± 564.35	5050.11 ± 191.04	0.77 ± 0.04
DNS [39]	4244.70 ± 475.12	4607.91 ± 600.16	5059.68 ± 240.75	0.82 ± 0.03
TASAC [23]	1256.23 ± 190.79	4374.47 ± 605.69	4208.07 ± 301.07	0.68 ± 0.08
SELKD(Our)	$\mathbf{5460.14 \pm 105.17}$	$\mathbf{5411.97 \pm 276.48}$	$\mathbf{5125.52 \pm 121.22}$	$\mathbf{0.93 \pm 0.02}$

a) The maximum value for each task is in bold.

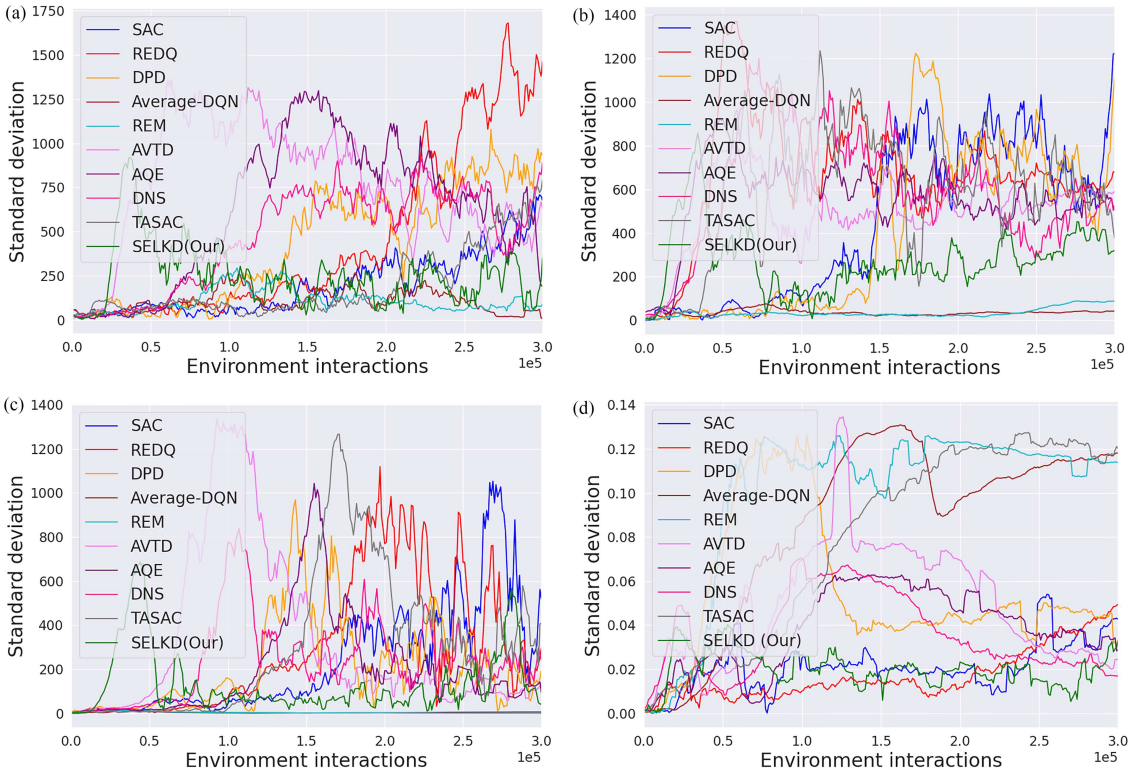

Figure 4 (Color online) Standard deviation of the learning curves for the OpenAI Gym tasks and the 3C assembly task. (a) Ant; (b) Walker2d; (c) Humanoid; (d) 3C assembly task.

whole training process to show the training stability, as is shown in Figure 4. From Figure 4, it can be seen that the REM and Average-DQN algorithms performed poorly in all test tasks, so it is unnecessary to compare REM and Average-DQN with the other algorithms regarding the training stability. During the initial training period, the policy performances of most algorithms are poor and exhibit high variance due to the random initialization of each actor and critic. Throughout the training process, except for the initial stage, the SELKD algorithm consistently demonstrates low standard deviation, indicating excellent training stability. Furthermore, a histogram depicting the average standard deviation of learning curves in the second half of the entire training process (environment interaction time-steps ranging from 1.5×10^5 to 3.0×10^5) is illustrated in Figure 5. Compared to the other baseline algorithms, except REM and average-DQN, the SELKD algorithm demonstrates a generally smaller standard deviation in the later stages of skill training. The results provide additional validations for the strong skill training stability of the SELKD algorithm. Based on these results, the proposed SELKD algorithm demonstrates the best performance and superior training stability in all test tasks.

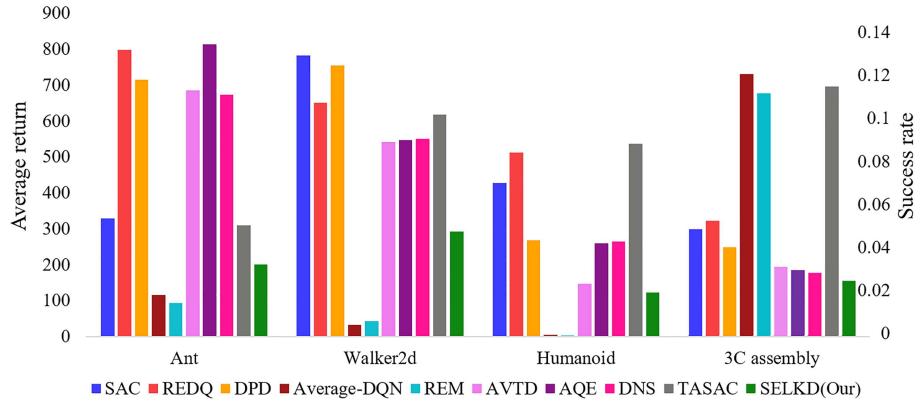

Figure 5 (Color online) Average standard deviation for learning curves during the second half of the overall training process. Our algorithm (SELKD(Our)) exhibits greater stability throughout the skill training process.

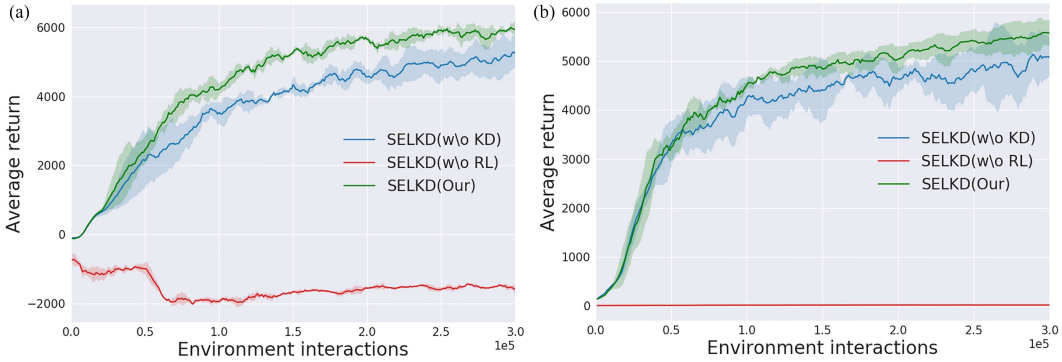

Figure 6 (Color online) Ablation study results for knowledge distillation and reinforcement learning mechanisms. The comparisons indicate that knowledge distillation and reinforcement learning are both beneficial for skill enhancement learning in SELKD. (a) Ant; (b) Walker2d.

6.2 Ablation study

In this subsection, we further examine which particular component of SELKD is important. To achieve this, we perform ablation studies to understand the contribution of each component, including knowledge distillation, reinforcement learning, target valuation formation, and the number of actors.

Firstly, we compare the full model with two variants of SELKD which correspond to pure knowledge distillation and pure reinforcement learning (Subsection 6.2.1). The two baselines use the same setup as the full model SELKD. Additionally, we conduct ablation studies to understand the impact of actors (Subsection 6.2.2) and the impact of the target value formation (Subsection 6.2.3). The significance of each component shows similar effects across all the test tasks. To make the content of this section more concise, we present the experimental results of the representative tasks, namely the Ant task and the Walker2d task.

6.2.1 *Pure reinforcement learning/pure knowledge distillation*

One variant of our algorithm is SELKD(w\o KD) (pure reinforcement learning). In SELKD(w\o KD), only one actor is selected to be trained using both reinforcement learning and knowledge distillation learning, while the other actors only engage in reinforcement learning. Subsequently, all actors are tested with MC rollouts, and the knowledge from the best-performing actor is distilled to the other actors.

Another variant of our algorithm is SELKD(w\o RL) (pure knowledge distillation). In SELKD(w\o RL), one actor is chosen to be trained via both reinforcement learning and knowledge distillation learning throughout the entire training process, while the other actors are trained using knowledge distillation with the best-performing actor. The results are illustrated in Figure 6. Figure 6 depicts that both the reinforcement learning and knowledge distillation mechanisms play crucial roles in enhancing policy learning performance in our algorithm.

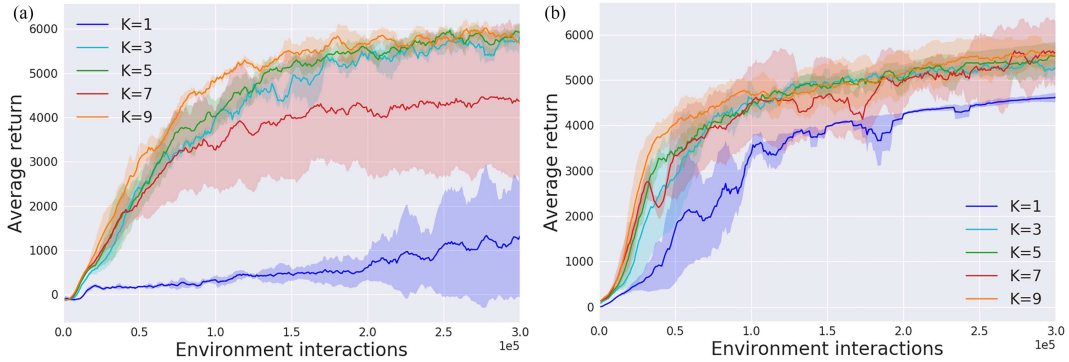

Figure 7 (Color online) Ablation study results for the number of actors K. The trained policy performance is satisfactory with $K = 5$. (a) Ant; (b) Walker2d.

6.2.2 *Number of actors*

We investigate the impact of the number of actors (K). Specifically, five different experiment settings with $K = 1, K = 3, K = 5, K = 7, K = 9$ were implemented, and the corresponding results are presented in Figure 7. According to the findings, the policy performance is at their best when $K = 9$ and satisfactory when $K = 5$. Due to the time and computational constraints, we did not investigate the effects of involving more than 9 actors ($K = 9$) on policy performance. Therefore, our experiments use a default setting of 5 actors ($K = 5$).

6.2.3 *Target value formation*

Inspired by [23], we propose four different target value calculation formations, which are the following.

Case 1. In-target min-min (denoted as min-min):

$$y = r + \gamma \min_{i=1,\ldots,K} \left(\min_{j \in \mathcal{M}} Q_{\phi_{\text{target},j}}(s', \tilde{a}'_i) + \alpha \log(\pi_{\theta_i}(\cdot|s')) \right), \ \tilde{a}'_i \sim \pi_{\theta_i}(\cdot|s').$$

Case 2. In-target min-max (denoted as min-max):

$$y = r + \gamma \min_{i=1,\ldots,K} \left(\max_{j \in \mathcal{M}} Q_{\phi_{\text{target},j}}(s', \tilde{a}'_i) + \alpha \log(\pi_{\theta_i}(\cdot|s')) \right), \ \tilde{a}'_i \sim \pi_{\theta_i}(\cdot|s').$$

Case 3. In-target max-min (denoted as max-min):

$$y = r + \gamma \max_{i=1,\ldots,K} \left(\min_{j \in \mathcal{M}} Q_{\phi_{\text{target},j}}(s', \tilde{a}'_i) + \alpha \log(\pi_{\theta_i}(\cdot|s')) \right), \ \tilde{a}'_i \sim \pi_{\theta_i}(\cdot|s').$$

Case 4. In-target max-max (denoted as max-max):

$$y = r + \gamma \max_{i=1,\ldots,K} \left(\max_{j \in \mathcal{M}} Q_{\phi_{\text{target},j}}(s', \tilde{a}'_i) + \alpha \log(\pi_{\theta_i}(\cdot|s')) \right), \ \tilde{a}'_i \sim \pi_{\theta_i}(\cdot|s').$$

Figure 8 illustrates the learning curves for the above four target value formation cases. The results demonstrate that the in-target max-min formation achieves the best performance. Therefore, we chose the in-target max-min as our default target value calculation method.

7 Discussion

7.1 Backbone algorithm

In Section 4, our approach SELKD employs the off-policy algorithm SAC [12] as its backbone algorithm. Another well-known off-policy algorithm is TD3 (twin delayed deep deterministic policy gradient) [16]. However, in most control task cases, SAC shows better performance than TD3 with the reason that SAC encourages exploration by utilizing an entropy regularization. We also conducted experiments to analyze

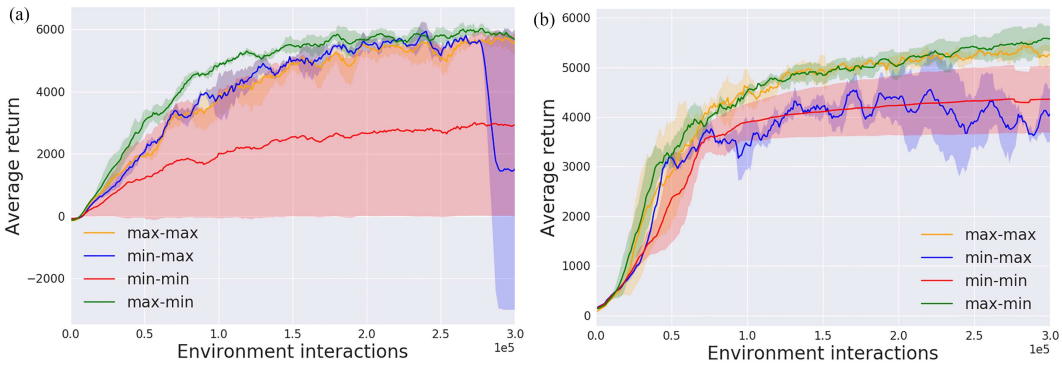

Figure 8 (Color online) Ablation study results for different target value formations. The in-target max-min formation achieves the best performance. (a) Ant; (b) Walker2d.

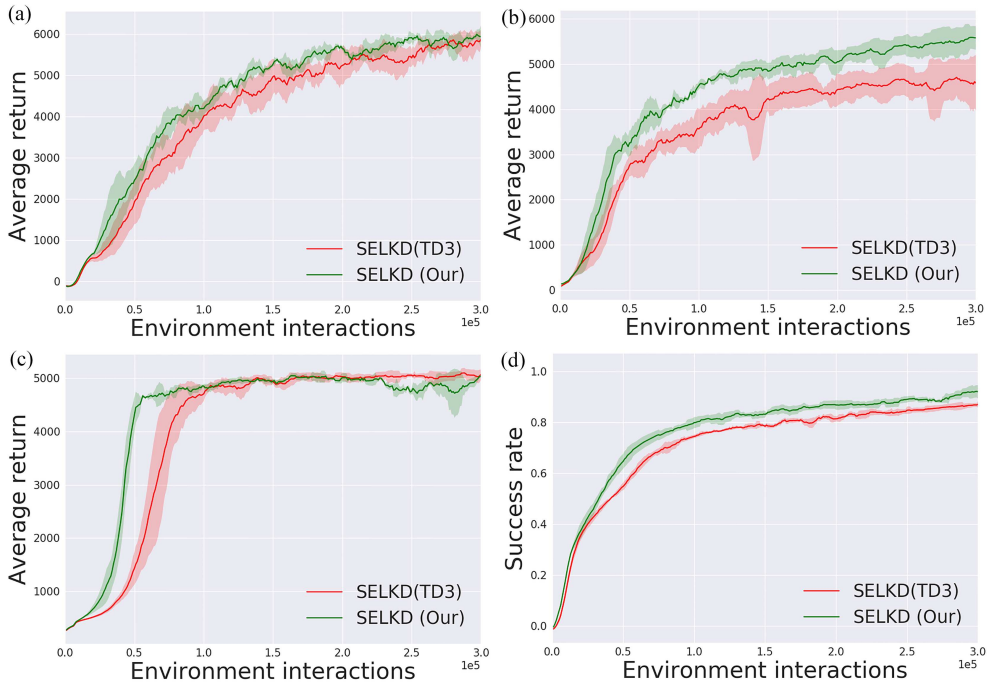

Figure 9 (Color online) Learning curves of the skills trained with SAC backbone (SELKD(Our)) and TD3 backbone (SELKD(TD3)). (a) Ant; (b) Walker2d; (c) Humanoid; (d) 3C assembly task.

the performance of our proposed algorithm built on SAC (denoted as SELKD(our)) and TD3 (denoted as SELKD(TD3)). The results are shown in Figure 9.

As depicted in Figure 9, SELKD(Our) performs better than SELKD(TD3) in the Ant, Walker2d, and 3C assembly tasks. However, in the Humanoid task, SELKD(TD3) is slightly superior to SELKD(Our), consistent with the conclusion that SAC outperforms TD3 in most cases.

7.2 Computational complexity

A key differentiating factor between the SELKD algorithm and other baseline algorithms is the utilization of multiple actors ($K \geqslant 2$), where each actor undergoes individual optimization through RL and mutual learning with knowledge distillation. The SELKD algorithm offers significant advantages in enhancing skill performance and improving training stability but also presents challenges in terms of computational complexity. In this subsection, we elucidate the computational complexity from two perspectives: space cost analysis and training complexity analysis. To maintain clarity and concision in the analysis, we compare our algorithm SELKD with the representative algorithm REDQ [18].

(1) Space cost analysis: Let n_a and n_c denote the parameter sizes of one actor and one critic, respectively. Ignoring parameters with low quantities, the total parameter size of the SELKD algorithm can

be estimated as $O(Kn_a + Nn_c)$, where K represents the number of actors and N denotes the number of critics. In contrast, the total parameter size of REDQ can be estimated as $O(n_a + Nn_c)$. It is evident that our approach requires more parameters than the baseline algorithm.

(2) Training complexity analysis: As the number of actor parameters increases, there are more computational FLOPs (floating-point operations per second) inevitably needed in the reinforcement learning process. Additionally, each actor engages in knowledge distillation learning, resulting in additional computational FLOPs being required. Nevertheless, parallel optimization of all actors can reduce the required wall-clock time.

8 Conclusion

In this paper, we proposed an algorithm called SELKD that integrates multiple actors and multiple critics. We introduced a novel target value calculation method in SELKD to alleviate critic overestimation bias. The SELKD employs knowledge distillation to establish a mutual learning mechanism among actors. We performed a theoretical analysis to ensure convergence of the proposed SELKD algorithm. Finally, our experiments on several continuous control tasks demonstrate that our proposed method achieves superior performance in terms of skill learning performance and training stability compared to baseline algorithms. However, achieving these promising performance requires an increasing number of actors, resulting in more parameters needed. In light of this, future work can explore methods to decrease the overall parameter size, such as by sharing a portion of the neural network backbone across multiple actors. Additionally, future research can focus on enhancing long-horizon skill learning, complex task skill learning, and improving the scalability of the learning framework for real-world control tasks.

Acknowledgements This work was supported by "New Generation Artificial Intelligence" Key Field Research and Development Plan of Guangdong Province (Grant No. 2021B0101410002), National Science and Technology Major Project of the Ministry of Science and Technology of China (Grant No. 2018AAA0102900), and National Natural Science Foundation of China (Grant Nos. U22A2057, 62133013).

References

1 Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge: MIT Press, 2018
2 Ibarz J, Tan J, Finn C, et al. How to train your robot with deep reinforcement learning: lessons we have learned. Int J Robotics Res, 2021, 40: 698–721
3 Luo F-M, Xu T, Lai H, et al. A survey on model-based reinforcement learning. Sci China Inf Sci, 2024, 67: 121101
4 Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518: 529–533
5 Liu N J, Cai Y H, Lu T, et al. Real-sim-real transfer for real-world robot control policy learning with deep reinforcement learning. Appl Sci, 2020, 10: 1555
6 Gu S X, Holly E, Lillicrap T, et al. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: Proceedings of IEEE International Conference on Robotics and Automation, 2017. 3389–3396
7 Haarnoja T, Pong V, Zhou A, et al. Composable deep reinforcement learning for robotic manipulation. In: Proceedings of IEEE International Conference on Robotics and Automation, 2018. 6244–6251
8 Levine S, Finn C, Darrell T, et al. End-to-end training of deep visuomotor policies. J Mach Learn Res, 2016, 17: 1334–1373
9 Fazeli N, Oller M, Wu J, et al. See, feel, act: hierarchical learning for complex manipulation skills with multisensory fusion. Sci Robot, 2019, 4: eaav3123
10 Liu N J, Lu T, Cai Y H, et al. Manipulation skill learning on multi-step complex task based on explicit and implicit curriculum learning. Sci China Inf Sci, 2022, 65: 114201
11 Ziebart B D. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. Dissertation for Ph.D. Degree. Pittsburgh: Carnegie Mellon University, 2010
12 Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of International conference on machine learning, 2018. 1861–1870
13 Zhu Y, Wang Z, Merel J, et al. Reinforcement and imitation learning for diverse visuomotor skills. 2018. ArXiv:1802.09564
14 Hasselt H. Double Q-learning. In: Proceedings of Advances in Neural Information Processing Systems, 2010
15 van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: Proceedings of AAAI Conference on Artificial Intelligence, 2016
16 Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proceedings of International Conference on Machine Learning, 2018. 1587–1596
17 Lan Q, Pan Y, Fyshe A, et al. Maxmin Q-learning: controlling the estimation bias of Q-learning. In: Proceedings of International Conference on Learning Representations, 2020
18 Chen X Y, Wang C, Zhou Z J, et al. Randomized ensembled double Q-learning: learning fast without a model. In: Proceedings of International Conference on Learning Representations, 2021
19 Rusu A A, Colmenarejo S G, Gülçehre Ç, et al. Policy distillation. In: Proceedings of International Conference on Learning Representations, 2016. 1–13
20 Dillenbourg P. Collaborative Learning: Cognitive and Computational Approaches. New York: Elsevier Science, 1999
21 Littman M L. Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of Machine Learning Proceedings, 1994. 157–163

22 Hadfield-Menell D, Russell S J, Abbeel P, et al. Cooperative inverse reinforcement learning. In: Proceedings of Advances in Neural Information Processing Systems, 2016
23 Joshi T, Kodamana H, Kandath H, et al. TASAC: a twin-actor reinforcement learning framework with a stochastic policy with an application to batch process control. Control Eng Pract, 2023, 134: 105462
24 Lai K H, Zha D, Li Y, et al. Dual policy distillation. In: Proceedings of International Joint Conference on Artificial Intelligence, 2020. 3146–3152
25 Haarnoja T, Tang H, Abbeel P, et al. Reinforcement learning with deep energy-based policies. In: Proceedings of International Conference on Machine Learning, 2017. 1352–1361
26 Fox R, Pakman A, Tishby N. Taming the noise in reinforcement learning via soft updates. In: Proceedings of Conference on Uncertainty in Artificial Intelligence, 2016. 202–211
27 Nair A, McGrew B, Andrychowicz M, et al. Overcoming exploration in reinforcement learning with demonstrations. In: Proceedings of IEEE International Conference on Robotics and Automation, 2018. 6292–6299
28 Torabi F, Warnell G, Stone P. Behavioral cloning from observation. In: Proceedings of International Joint Conference on Artificial Intelligence, 2018. 4950–4957
29 Popov I, Heess N, Lillicrap T, et al. Data-efficient deep reinforcement learning for dexterous manipulation. 2017. ArXiv:1704.03073
30 Kumar A, Gupta A, Levine S. DisCor: corrective feedback in reinforcement learning via distribution correction. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 18560–18572
31 Czarnecki W M, Pascanu R, Osindero S, et al. Distilling policy distillation. In: Proceedings of International Conference on Artificial Intelligence and Statistics, 2019. 1331–1340
32 Zhao C, Hospedales T. Robust domain randomised reinforcement learning through peer-to-peer distillation. In: Proceedings of Asian Conference on Machine Learning, 2021. 1237–1252
33 Anschel O, Baram N, Shimkin N. Averaged-DQN: variance reduction and stabilization for deep reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2017. 176–185
34 Agarwal R, Schuurmans D, Norouzi M. An optimistic perspective on offline reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2020. 104–114
35 Lee K, Laskin M, Srinivas A, et al. SUNRISE: a simple unified framework for ensemble learning in deep reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2021. 6131–6141
36 Wu Y, Chen X, Wang C, et al. Aggressive Q-learning with ensembles: achieving both high sample efficiency and high asymptotic performance. In: Proceedings of Advances in Neural Information Processing Systems, 2022
37 Yang Z, Ren K, Luo X, et al. Towards applicable reinforcement learning: improving the generalization and sample efficiency with policy ensemble. In: Proceedings of International Joint Conference on Artificial Intelligence, 2022
38 Li Q, Kumar A, Kostrikov I, et al. Efficient deep reinforcement learning requires regulating overfitting. In: Proceedings of International Conference on Learning Representations, 2022
39 Sheikh H, Frisbee K, Phielipp M. DNS: determinantal point process based neural network sampler for ensemble reinforcement learning. In: Proceedings of International Conference on Machine Learning, 2022. 19731–19746
40 Huang Z, Zhou S, Zhuang B, et al. Learning to run with actor-critic ensemble. 2017. ArXiv:1712.08987
41 Wang H, Yu Y, Jiang Y. Review of the progress of communication-based multi-agent reinforcement learning (in Chinese). Sci Sin Inform, 2022, 52: 742–764
42 Li J C, Wu F, Shi H B, et al. A collaboration of multi-agent model using an interactive interface. Inf Sci, 2022, 611: 349–363
43 Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning. In: Proceedings of International Conference on Learning Representations, 2016
44 Singh S, Jaakkola T, Littman M L, et al. Convergence results for single-step on-policy reinforcement-learning algorithms. Machine Learn, 2000, 38: 287–308
45 Brockman G, Cheung V, Pettersson L, et al. OpenAI Gym. 2016. ArXiv:1606.01540
46 Kingma D P, Ba J. Adam: a method for stochastic optimization. In: Proceedings of International Conference on Learning Representations, 2015