ERL-TD: Evolutionary Reinforcement Learning Enhanced with Truncated Variance and Distillation Mutation

Qiuzhen Lin[1], Yangfan Chen[1], Lijia Ma[1], Wei-Neng Chen[2], Jianqiang Li[3*]

[1]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[2]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[3]National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen, China
lijq@szu.edu.cn

Abstract

Recently, an emerging research direction called Evolutionary Reinforcement Learning (ERL) is proposed, which combines evolutionary algorithm into reinforcement learning (RL) for tackling the tasks of sequential decision making. However, the recently proposed ERL algorithms often suffer from two challenges: the inaccuracy of policy estimation caused by the overestimation bias in RL and the insufficience of exploration caused by inefficient mutations. To alleviate these problems, we propose an Evolutionary Reinforcement Learning algorithm enhanced with Truncated variance and Distillation mutation, called ERL-TD. We utilize multiple Q-networks to evaluate state-action pairs, so that multiple networks can provide more accurate evaluations for state-action pairs, in which the variance of evaluations can be adopted to control the overestimation bias in RL. Moreover, we propose a new distillation mutation to provide a promising mutation direction, which is different from traditional mutation generating a large number of random solutions. We evaluate ERL-TD on the continuous control benchmarks from the OpenAI Gym and DeepMind Control Suite. The experiments show that ERL-TD shows excellent performance and outperforms all baseline RL algorithms on the test suites.

Introduction

Reinforcement Learning (RL) is a branch of machine learning that focuses on teaching an agent how to make decisions through trial and error interactions with an environment. It is inspired by the way that human learns from feedback and rewards, which has been applied in various domains (Grinsztajn, Furelos-Blanco, and Barrett 2022; Zhao et al. 2022; Rey, Hammad, and Saberi 2023), including robotics (Luo et al. 2023), game playing (Vinyals et al. 2019), autonomous driving (Sallab et al. 2017), and resource management (Ni et al. 2021). Although RL demonstrates impressive capabilities in solving the tasks of sequential decision-making, it still suffers from weak exploration and the sensitivity of hyperparameters. On the other hand, Evolutionary Algorithm (EA) is a kind of computational intelligence methods, which simulates the natural evolutionary process by maintaining a population and iteratively searching for superior solutions. In each iteration, individuals with high fitness are selected to produce offspring through genetic crossover and mutation, while individuals with low fitness are eliminated. EA shows advantages such as strong search capability, robustness, and stable convergence. Even though EA has achieved some success in training deep neural networks for RL (Such et al. 2018), it is obviously less sample-efficient than the RL methods like Deep Q-Learning (Mnih et al. 2015).

Since RL and EA have complementary advantages, a natural idea is to integrate them for designing better policy optimization algorithms. In this exciting direction, there are many novel ideas emerging in recent years (Gangwani and Peng 2018; Khadka and Tumer 2018). A first work is Evolutionary Reinforcement Learning (ERL) proposed in (Khadka and Tumer 2018), which is an innovative framework integrating Genetic Algorithm (GA) (Mitchell 1998) with Deep Deterministic Policy Gradien (DDPG) (Lillicrap et al. 2015). This method maintains an RL agent and multiple actor networks, which coexist and interact with each other. By this way, it facilitates the sharing of knowledge among populations, which allows efficient information transfer and accelerates the learning process. However, the integration of EA and RL is still rudimentary. As a result, many variants of ERL (Sigaud 2022) are subsequently proposed, which are improved from two aspects: enhancing the communication between RL policy and EA policies (Khadka et al. 2019; Li et al. 2022), and improving the crossover and mutation operators (Gangwani and Peng 2018; Bodnar, Day, and Lió 2020). However, the above ERL algorithms still suffer from low efficiency and poor stability, which are mainly induced by inaccuracy of Q-values evaluation and randomness of evolution. Generally, Q-values are used to guide the direction of actions in RL, while evolutionary operators will generate promising candidates in EA. Thus, without accurate Q-values estimation in RL and efficient evolutionary operators in EA, the population of actor networks would generate a large number of meaningless experiences, leading to inefficiency and poor stability.

Here, we discuss ERL from the perspectives of Q-values estimation and population evolution. As it is described in (Thrun and Schwartz 1993), off-policy RL often suffers from overestimation bias, which also exists in ERL. To address this issue, we use the mean of multiple networks minus the exponential of their variance as the target to update Q-networks. Since each network has a different initializa-

*Corresponding author.

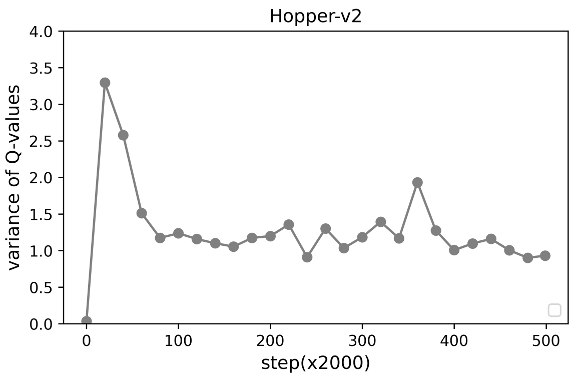


Figure 1: This curve represents the changes of the variance of Q-values in multiple Q-networks, which are obtained by sampling 500 states randomly from the buffer.

tion state, the variance of Q-values for untrained state-action pairs is larger than that for trained state-action pairs. Therefore, the variance of Q-values estimation can roughly represent its uncertainty in RL. Thus, it can be used to control the overestimation bias in RL. However, as shown in Figure 1, even training with the same experience pool for many times, the Q-values of multiple networks are still different and their variance won't converge to 0, as they have different initialization states. Thus, if the target consistently uses the mean of multiple networks minus the exponential of their variances, there will exist underestimation bias that may lead the networks to get trapped in local optima. To solve this issue, we use the mean of multiple networks as the update target after their variance becomes stable, otherwise we reduce overestimation by using the mean of multiple networks minus the exponential of their variance as the target. Moreover, it is also important in ERL to control the direction of mutation in policy networks. All policy networks should explore near the policy that is considered optimal by the current Q-networks. This is because we cannot verify whether the Q-values estimation is correct or not if the experience pool doesn't contain the actions that are considerred optimal by the current Q-networks. Thus, ERL-TD distills the population by the existing best policy before mutation, so the policy generated by distillation mutation won't be overly random. Finally, we summarize the contributions of this work:

- We propose a truncated variance Bellman backup to control the overestimation bias of Q-values in ERL, which can improve the accuracy of Q-values estimation in ERL.
- We propose a distillation mutation, which can provide an excellent direction for mutation and generate more meaningful experiences, so as to enhance the efficiency and stability of ERL.
- The proposed method ERL-TD sets a new state-of-the-art in terms of data efficiency and asymptotic performance compared to the existing ERL algorithms on the Gym environments. Also, ERL-TD achieves superior performance on the DMC environments.

Background

ERL is a framework integrating EA into RL, which leverages EA to enhance the exploration of RL. To validate our performance on the Gym environments, we integrate EA into Soft Actor-Critic (SAC) (Haarnoja et al. 2018) to form ERL-TD. Moreover, to validate our performance in a pixel-input environment, we combine ERL-TD with Data-regularized Q (DrQ) (Kostrikov, Yarats, and Fergus 2020), which can be run on the DMC environments. Thus, in this section, we introduce some background of our used RL and EA separately.

Reinforcement Learning RL is a branch of machine learning that deals with decision-making problems. The goal of RL is to learn an optimal policy that maximizes the cumulative discounted rewards. Formally, RL problems are often modeled as Markov Decision Processes (MDP). The MDP is defined by the tuple (S, A, P, R, γ). The goal of the agent is to learn an optimal policy π^* that maximizes the expected cumulative reward (also called the return) from the starting state. The return is defined as: $G_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where $0 \leq \gamma \leq 1$ is the discount factor and T is the maximum episode horizon. The interactive process of RL can be summarized as a dynamic cycle of observation, action, and learning. To improve its exploration, SAC integrates the ideas of maximum entropy and off-policy learning in RL. The core idea of SAC is to optimize a stochastic policy by maximizing both the expected cumulative reward and the entropy of the policy distribution. This entropy regularization encourages exploration and helps to prevent premature convergence to suboptimal policies. Moreover, the high dimensionality of pixel-input is one of the key challenges in RL. To enhance the performance in pixel-input environments, DrQ introduces the techniques of data augmentation based on SAC. Specifically, DrQ applies edge padding and random cropping to the input images, which are commonly used in computer vision.

Evolutionary Algorithm EA (Bäck and Schwefel 1993) is a kind of optimization algorithms as inspired by natural selection, which maintains a population and runs iterative search for solving complex problems (Fogel 2006; Spears et al. 1993). Generally, EA starts with creating an initial population of candidate solutions (also called individuals). Each individual in the population is evaluated and assigned a fitness value based on how well it solves the problem. Individuals with higher fitness values have a higher chance of being selected for reproduction, which will generate offspring. In ERL, each individual of EA represents a policy network in RL, where the crossover and mutation are implemented as changes to the neural network weights (Floreano, Dürr, and Mattiussi 2008; Lüders et al. 2017; Risi and Togelius 2015; Stanley and Miikkulainen 2002).

Motivation

The off-policy RL suffers from the problem of Q-values overestimation bias. The issue of overestimation can be constructed as Jensen's inequality (Thrun and Schwartz 1993). Specifically, we use Q^{appro} to denote the Q-values evaluated by the function approximation and Q^{true} to denote the true Q-value. Suppose that Q^{appro} is equal to Q^{true} corrupted by a noise term $U(a)$, where $\forall a\ \mathbb{E}_U[U(a)] = 0$,

$$\begin{aligned} \max_a Q^{appro}(s,a) &= \max_a \mathbb{E}_U[Q^{true}(s,a) + U(a)] \\ &\leq \mathbb{E}_U\left[\max_a\{Q^{true}(s,a) + U(a)\}\right] \end{aligned} \tag{1}$$

In practice, the stochasticity of state transitions, the randomness of rewards, and the spontaneous errors of function approximation can generate noise. The overestimation bias caused by the $U(a)$ propagates backward and accumulates over the learning process. To further clarify the overestimation phenomenon, we randomly sample over 500 states from the buffer on the Hopper-v2 environment. Starting from the sampled states, we used the accumulated discount rewards of the current policy as the true Q-values and the mean evaluations of the Q-networks as the estimated Q-values. The results, as shown in Figure 2, indicate that the overestimation of Q-values seriously impairs the evaluation process of RL. Furthermore, the overestimation phenomenon does not disappear, despite the Q-values estimation becoming gradually stable.

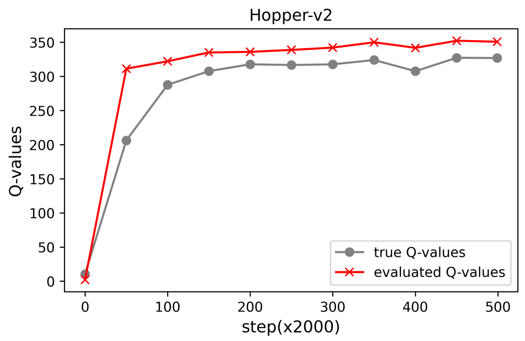


Figure 2: The true Q-values (grey) are estimated by using the average discounted return over 500 states sampled from the replay buffer, following the current policy. The evaluated Q-values (red) are estimated by current Q-networks.

In ERL, the process of evolution mainly involves crossover and mutation, the randomness of which is a crucial issue. To search for solutions, traditional mutations in EA introduce Gaussian noise in the solution space, while crossover involves exchanging fragments of elite solutions. However, if crossover and mutation are not controlled, the policy network will diverge in an unpredictable direction.

Related Work

Recently, integrating EA into RL has emerged as a promising method. A number of ERL algorithms (Ma et al. 2022; Zhou et al. 2023; Li et al. 2023) have been proposed to enhance the performance of RL. The first ERL framework was proposed by (Khadka and Tumer 2018), which utilizes the EA population to provide diverse data for training the DDPG algorithm (Lillicrap et al. 2015). Periodically, the RL agent is reinserted into the EA population to inject gradient information. This method inherits the credit assignment capability of RL and effectively explores different sets of policies. In parallel, CEM-RL (Pourchot and Sigaud 2019) integrates Cross-Entropy Method (CEM) into Twin Delayed Deep Deterministic policy gradient algorithm (TD3) (Fujimoto, Hoof, and Meger 2018). CEM is an optimization algorithm used to solve stochastic optimization problems, which iteratively updates a population of candidate solutions by cross-entropy based on the given objective function. In the setting of CEM-RL, TD3 provides the gradient of Temporal Difference (TD) errors for half of the individuals in the population. After that, a number of ERL variants are subsequently proposed, which focus on enhancing the communication between the RL and EA policies or/and improving the crossover and mutation operators. For example, Collaborative Evolutionary Reinforcement Learning (CERL) (Khadka et al. 2019) extends a single RL agent to multiple agents with different hyperparameter settings. At the same time, all learners use a shared replay buffer to achieve high sample efficiency. The entire process is bound by the EA to explore the parameter space and integrate the best policy. Although the above mentioned ERL methods are innovative, their used crossover and mutation operators in EA are destructive. Thus, Proximal Distilled Evolutionary Reinforcement Learning (PDERL) (Bodnar, Day, and Lió 2020) proposes the distillation crossover and proximal mutation to alleviate the policy collapse at the parameter level. Q-filtered behavior distillation crossover merges two parent policies in the phenotype space to form a child policy, while proximal mutation ensures that there is no significant difference between the new policy and the old policy. Similarly, to address the catastrophic destruction of traditional crossover and mutation in neural network parameter, Genetic Policy Optimization (GPO) (Gangwani and Peng 2018) employs imitation learning for policy crossover in the state space and utilizes policy gradient methods for mutation. State space crossover effectively combines two parent policies into an offspring or sub-policy, which attempts to mimic its best parent policy while generating similar state distributions. The mutation operator is more effective than random parameter perturbation, while it also maintains genetic diversity (Parker-Holder et al. 2020). Evolutionary Reinforcement Learning with Two-scale State Representation and Policy Representation (ERL-Re^2) (Li et al. 2022) points out that the common knowledge between EA and RL policy networks should not be neglected. Therefore, all EA and RL policies in ERL-Re^2 share the same nonlinear state representation while maintaining their respective linear policy representations. Furthermore, to ensure alignment between the current Q-values and the experiences generated by the policies, the mutation of ERL-Re^2 occurs at the behavioral level rather than across all parameters. Inspired by the above-mentioned method, we propose distillation mutation to alleviate the destruction of crossover and mutation. Distillation mutation initially guides the student policy towards minimizing the distance from the teacher policy. After distilling, it mutates by using the magnitude of gradients obtained from the last distillation as the variance for Gaussian mutation, and the direction of the gradients as the direction for Gaussian mutation. In addition, RACE (Li et al. 2023)

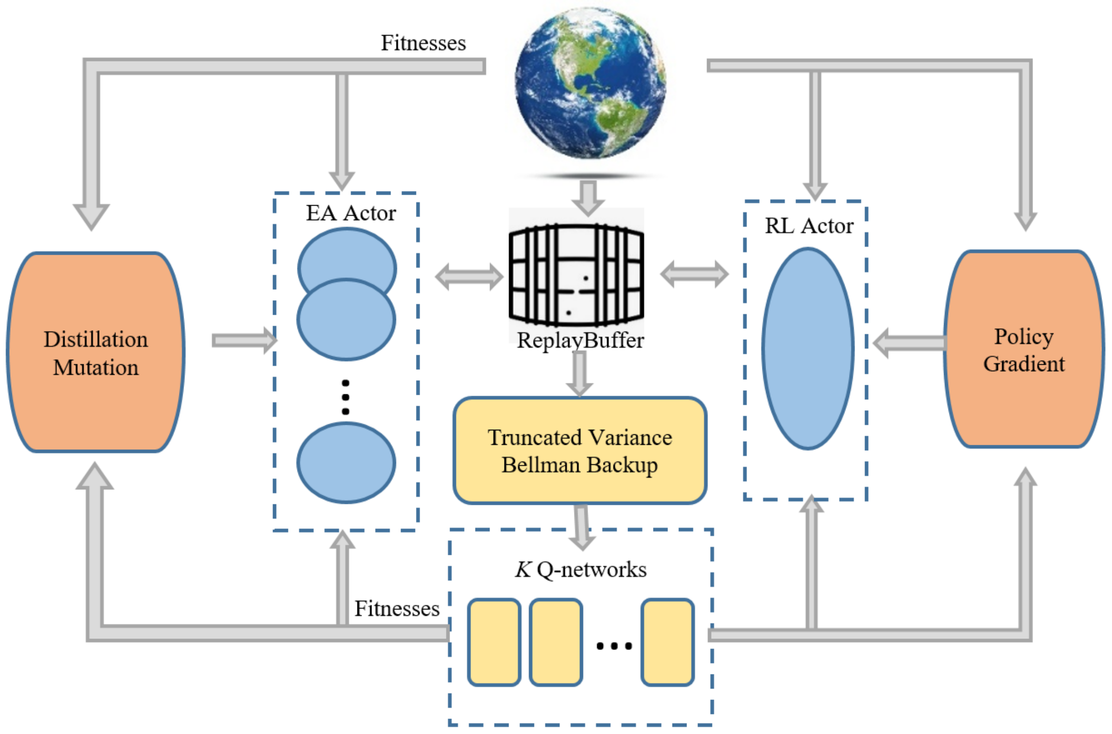


Figure 3: Illustration of ERL-TD. Multiple Q-networks are used to evaluate the value of the state-action pairs. The RL actor network is updated according to the direction of Q-values improvement. The EA actor networks learn from the RL actor network or the elite actor network in the population. Moreover, the experiences generated by all actor networks are shared.

believes that there should be the same hidden feature representation in all policies space to improve the efficiency of search in multi-agent reinforcement learning (MARL).

At the same time, off-policy RL faces a critical issue of overestimation. To address the issue of overestimation, TD3 utilizes the minimum evaluation value of two target networks as the update target. MeanQ (Liang et al. 2022) focuses on addressing the issue of the overestimation for discrete actions by using the mean evaluation values of multiple networks as the update target. Randomized Ensembled Double Q-Learning (REDQ) (Chen et al. 2020) randomly selects two out of ten target networks for target evaluation and chooses the lowest evaluation value as the target value. Additionally, REDQ increases its update frequency (UTD ratio) to 20 times, resulting in running slowly. In contrast, our approach uses the mean of multiple networks minus their variance as the target value to reduce overestimation, and it switches to using the mean evaluation directly as the target value once the evaluation stabilizes.

Proposed Algorithm

In this section, we provide an framework diagram of ERL-TD to understand the overview intuitively in Figure 3. Additionally, we introduce the details of each component in this work.

Overview

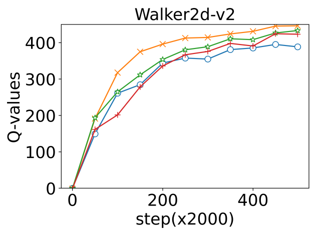
Based on the ERL framework, ERL-TD is designed by integrating EA into SAC algorithm. Our framework consists of an interactive environment, an experience buffer, an RL actor network, multiple Q-networks, and multiple EA actor networks. As shown in Algorithm 1, the first step is to initialize the components mentioned above. We utilize the actor networks to select action a based on the state s, and feed the actions to the environment. The environment returns the reward r and the next state s'. The data at time t consisting of (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer D. The RL side including Q-networks and actor network keeps updating at every step, while the EA side including the actor networks executes distillation mutation after all the EA actor networks have completed multiple episodes.

During the update process on the RL side, we dynamically adjust the batch size of the samples based on the number of the data in replay buffer. Because the experience pool only contains a few data in the early stage, updating it frequently will lead to the primacy bias (Nikishin et al. 2022). After sampling the data, the Q-networks are updated by the truncated variance Bellman backup, while the actor network is improved along the directions of Q-values and policy entropy. Generally, the evaluation function of an EA actor network is based on either the cumulative rewards in a episode

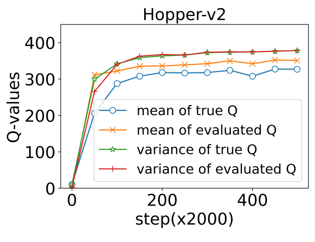
Algorithm 1: ERL-TD

```
1: Initialize $K$ Q-networks $Q_{\theta_k}$ for $k = 1, \dots, K$
2: Initialize $K$ target Q-networks $Q_{\hat{\theta}_k}$, $\hat{\theta}_k \leftarrow \theta_k$ for $k = 1, \dots, K$
3: Initialize RL policy network $\pi_{rl}$ with weights $\psi_{rl}$
4: Initialize a population of $M$ policy networks $\pi_{ea}$ with weights $\psi_{ea}^m$ for $m = 1, \dots, M$
5: Initialize replay buffer $D$ to capacity $N$
6: for $t = 1$ to $T$ do
7:     Sample action $a_t$ according to policy $\pi_\psi$
8:     Input action $a_t$ into the environment
9:     Get next state $s_{t+1}$ from the environment
10:    Initialize two lists of length $L$ with all values set to zero for storing historical $Q\_var^\alpha$
11:    Store transition $(s_t, a_t, r_t, s_{t+1}, list(Q\_var^\alpha)$ in $D$
12:    for $n = 1$ to $N$ do
13:        Sample $K$ batch transitions $B(s_t, a_t, r_t, s_{t+1})$
14:        Calculate the target Q-values by Formula (2) for $K \times B$ transition
15:        Calculate and update the current $Q\_var^\alpha$
16:        Calculate the standard deviation of the $Q\_var^\alpha$ list in a transition
17:        for $k = 1$ to $K$ do
18:            Update $Q_{\theta_k}$ with $B^k$ transition by minimizing $J_Q(\theta) = \frac{1}{2}(\mathcal{T}^\pi Q - Q_{eval}^k)^2$, where the $\mathcal{T}^\pi Q$ is the truncated variance bellman backup in Formula (3)
19:        end for
20:        Calculate policy update value with $K \times B$ transitions
21:        Update policy $\pi_\psi$ by maximizing $J_\pi(\psi) = \mathbb{E}_{a_t \sim \pi_\psi}[Q_{mean}(s_t, \pi_\psi(s_t)) - \alpha log \pi_\psi(a_t|s_t)]$, where $Q_{mean}(s_t, \pi_\psi(s_t))$ is the mean of multiple Q-values
22:    end for
23:    fitness = Evaluate($\pi_{ea}$, $D$, $\mathcal{P}$)
24:    Rank the population based on fitness scores
25:    Select the first $e$ policy network $\pi^e \in \pi_{ea}$ as elites
26:    $\pi_{ea}$ = DistillationMutation($\pi_{rl}$, $\pi^e$)
27: end for
```

or the estimated values from the Q-network. In our method, both evaluation methods are used simultaneously to balance bias and variance (Schulman et al. 2015). If only reward values are used, acquiring an accurate policy evaluation requires a significant amount of data. As only a limited number of sample data are available for evaluating the policy, there will be high evaluation variance for the return. On the other hand, if only the evaluation values from the Q-networks are used, there is a bias introduced by inaccurate fitting. After evaluation, the policy with the highest value is selected as the elite policy. The elite policy or RL policy serves as the teacher, while the policies in the population except the elite policy are treated as students for distillation mutation.



(a)



(b)

Figure 4: The overestimation of mean Q-values and variance of Q-values on the (a) Walker2d-v2 and (b) Hopper-v2 environments.

Truncated Variance Bellman Backup

Formally, we consider K networks Q_{θ_k} for $k = 1, \dots, K$, where θ_k denotes the parameters of the k-th Q-network. Each Q-network is randomly initialized to ensure fairness for the state-action pairs. The traditional off-policy RL updates the Q-networks by minimizing the TD error. However, some experiments in (Thrun and Schwartz 1993; Fox, Pakman, and Tishby 2015) show that the update way of off-policy RL suffers from the overestimation bias, which is caused by the stochasticity of Q-values estimation and the improvement of policy. Therefore, we propose the truncated variance Bellman backup to address the above issue. As shown in Figure 1, the variance of multiple Q-values does not converge to zero, but the change of variance tends to stabilize over time. Therefore, to avoid introducing new bias in the later stage of training, we use the mean of different networks as the Bellman update target when Q_var is stable, otherwise we use the mean of different networks minus the α power of the variance as the Bellman update target. To demonstrate the accuracy of the Q-values estimation in ERL-TD, we plot the estimated value of mean Q-values and truncated variance Q-values in Figure 4(a) and 4(b). The estimated Q-values calculated by the current Q-networks with 500 states are compared with the true Q-values, which are represented by the mean of discounted return generated by the current policy. The curves are learnt from the Hopper-v2 and Walker2d-v2 environments. It is evident that ERL-TD with truncated variance alleviates the overestimation bias. Formally, we use a formula to describe the truncated variance Bellman backup, as follows:

$$\mathcal{T}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma E_{S_{t+1} \sim p}[V(s_{t+1})], \tag{2}$$

where

$$V(s_{t+1}) = E_\pi[Q_{mean}(s_{t+1}, a_{t+1}) - Q\_over(s_{t+1}, a_{t+1})], \tag{3}$$

where

$$Q\_over = \begin{cases} 0, & STD_{history}[Q\_var^\alpha] < \vartheta \\ Q\_var^\alpha, & STD_{history}[Q\_var^\alpha] > \vartheta \end{cases} \tag{4}$$

where Q_{mean} and Q_var respectively represent the mean and variance of multiple Q-networks, and $STD_{history}[Q\_var^\alpha]$ is the standard deviation of the historical Q_var^α list. α is an exponent used to control temperature. ϑ is the threshold used to determine the stability of Q_var.

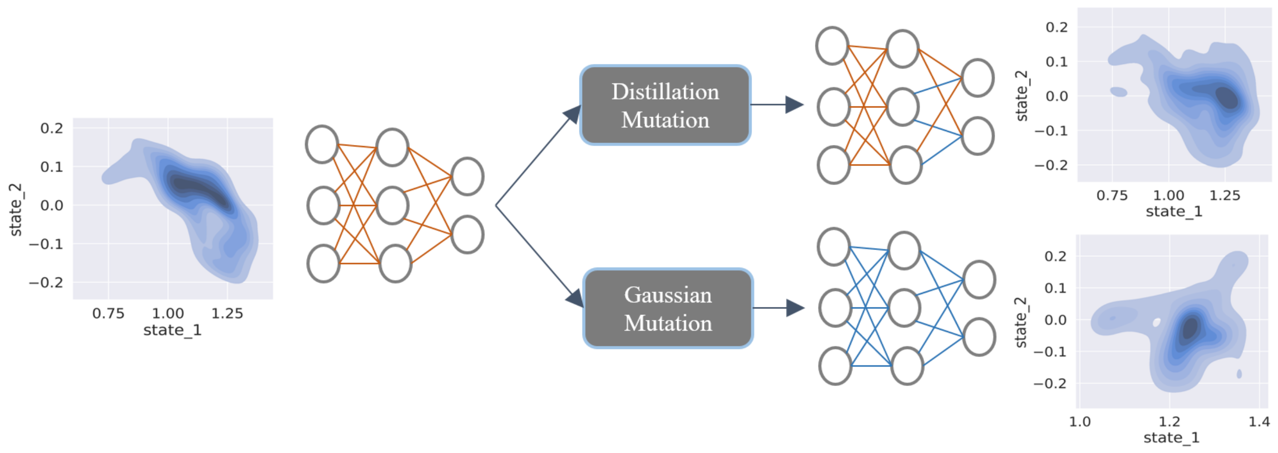


Figure 5: The figure illustrates that the state distributions of the policies with Gaussian mutation and distillation mutation compare with the state distribution of the policy network considered highly valuable. The density distribution plot is computed from the first and second dimensions of the states, which are generated by running multiple episodes on the Hopper-v2 environment using the policy networks. The policy network that the current Q-network considers high value is plotted in the left side. Specifically, it can be the RL policy network or the elite policy network. The policy networks after distillation mutation and Gaussian mutation are plotted in the right side, respectively.

Distillation Mutation

To address the issue of generating a large number of random solutions by traditional mutations, we propose the distillation mutation. In our approach, distillation brings the policy in the population close to the highly valuable policy and provides guidance for mutation. We select either the elite policy or the RL policy as the teacher. All policies in the population except the elite policy act as students learning from the teacher. Specifically, the process of distillation involves sampling data from the replay buffer D, using the output of the teacher network as the target and the output of the student networks as the estimate, calculating the gradient g of the loss function to update the student networks. Note that the gradient g of the last distillation is not used to update the weights, but to guide mutation. We construct a Gaussian model $\mathcal{N}(0, |g|)$ and sample from this model to obtain the magnitude of mutation. The direction of the gradient g determines the direction of mutation. The distillation mutation produces a policy network that is close to the highly valuable policy networks while still maintaining exploration. The distributions of the state generated by the policy networks using distillation mutation and Gaussian mutation are plotted in Figure 5. It is evident that the policy network with distillation mutation is closer to the teacher network compared to the policy network with Gaussian mutation.

Experimental Results

We evaluate ERL-TD on several continuous control benchmarks, which include OpenAI Gym (Brockman et al. 2016) and DeepMind Control Suite (DMC) (Tassa et al. 2018).

Setups

OpenAI Gym For the OpenAI Gym experiments with proprioceptive inputs (e.g., positions and velocities), we compare ERL-TD with several competitive RL algorithms, including ERL-Re^2 (Li et al. 2022), PDERL (Bodnar, Day, and Lió 2020), CERL (Khadka et al. 2019), CEM-RL (Pourchot and Sigaud 2019), ERL (Khadka and Tumer 2018), SAC (Haarnoja et al. 2018), and TD3 (Fujimoto, Hoof, and Meger 2018). Specifically, ERL-Re^2 is a state-of-the-art ERL variant. To compare their performance, we report their learning curves on six complex environments (HalfCheetah-v2, Walker2d-v2, Hopper-v2, Ant-v2, Humanoid-v2, and Swimmer-v2) in OpenAI Gym, each of which is run with five different seeds for 1000k steps.

DeepMind Control Suite (DMC) The DeepMind Control Suite presents a great challenge due to its large dimension of pixel-input. To demonstrate the robustness of our algorithm, we integrate ERL-TD and DrQ (Kostrikov, Yarats, and Fergus 2020), which are compared with some competitive RL algorithms, including Deep Planning Network (PlaNet) (Hafner et al. 2019b), Dreamer (Hafner et al. 2019a), Contrastive Unsupervised Representations for Reinforcement Learning (CURL) (Laskin, Srinivas, and Abbeel 2020), Reinforcement Learning with Augmented Data (RAD) (Laskin et al. 2020), Data-regularized Q (DrQ) (Kostrikov, Yarats, and Fergus 2020), and SUNRISE (Lee et al. 2021). PlaNet is a model-based RL algorithm, which improves efficiency by training a dynamics model. Dreamer is an RL agent that solves long-horizon tasks from images purely by latent imagination. CURL extracts high-dimensional pixel features by combining the idea of comparative learning. By employing data augmentation techniques, RAD and DrQ enhance the learning ability of RL for pixel input. SUNRISE

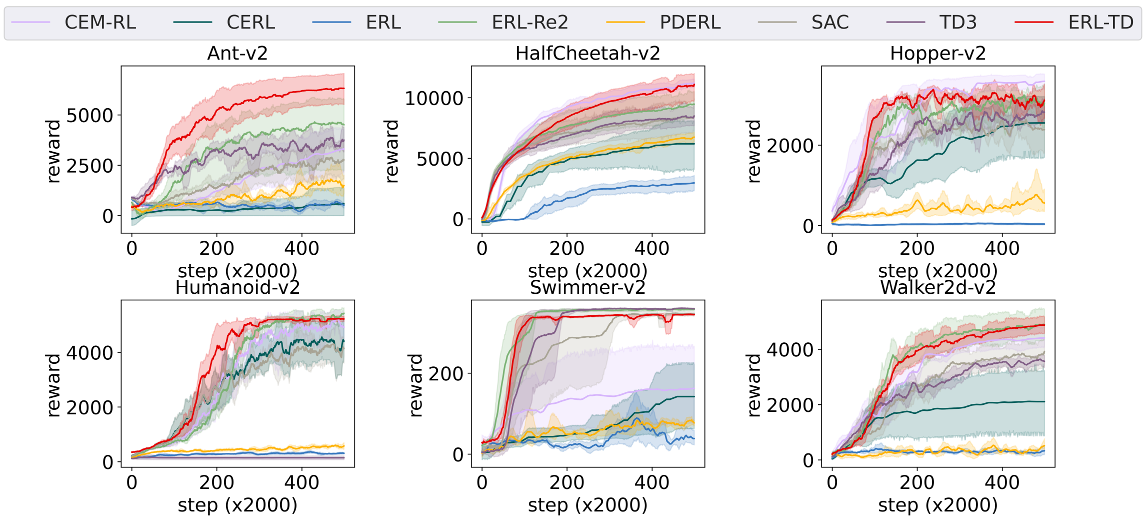


Figure 6: Performance on the OpenAI Gym. We plot the curves obtained from running five different seeds, where the solid lines represent the mean and the shaded regions represent the standard deviation.

500K step	PlaNet	Dreamer	CURL	SUNRISE	RAD	DrQ	ERL-TD (DrQ)
finger-spin	561±284	796±183	926±45	**983**±1	975±16	938±103	967±7
cartpole-swingup	475±71	762±27	845±45	**876**±4	873±3	868±10	875±5
reacher-easy	210±44	793±164	929±44	982±3	916±49	942±71	**990**±5
cheetah-run	305±131	570±253	518±28	**678**±46	624±10	660±96	637 ±15
walker-walk	351±58	897±49	902±43	**953** ±13	938±9	921±45	889 ±21
ball_in_cup-catch	460±380	879±87	959±27	969±5	966±9	963±9	**973**±19
100K step							
finger-spin	136±216	341±70	767±56	905±57	811±146	901±104	**907**±53
cartpole-swingup	297±39	326±27	582±146	591±55	373±90	759±92	**789** ±13
reacher-easy	20±50	314±155	538±233	722±50	567±54	601±213	**983**±14
cheetah-run	138±88	235±137	299±48	413±35	381±79	344±67	**421** ±53
walker-walk	224±48	277±12	403±24	**667** ±147	641±89	612±164	475 ±54
ball_in_cup-catch	0±0	246±174	769±43	633±241	666±181	913±53	**972**±27

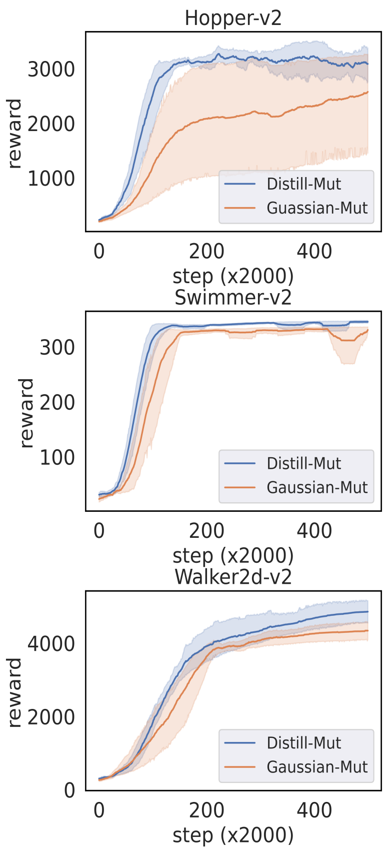
Table 1: Performance on DeepMind Control Suite at 100k and 500k. The normal-sized font and smaller font respectively indicate the mean and the standard deviation over 10 runs. Additionally, the bolded font represents the best results.

reweights the target Q-values based on the uncertainty of Q-ensemble. In this experiment, we report scores for ERL-TD and baseline methods by running 100k and 500k steps on six environments (finger-spin, cartpole-swingup, reacher-easy, cheetah-run, walker-walk, and ball_in_cup-catch).

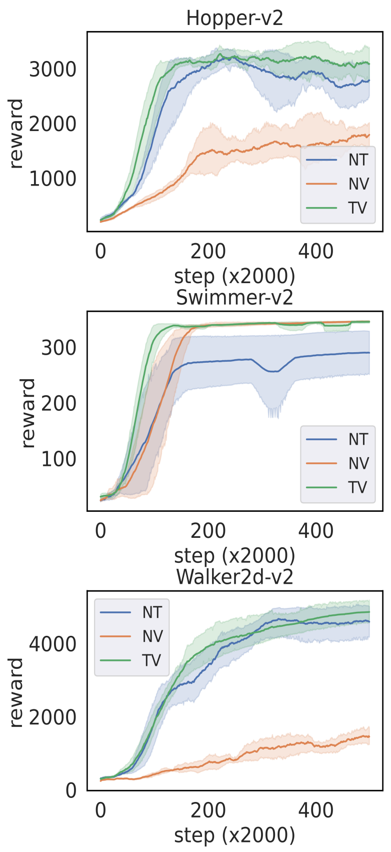
Comparative Evaluation

Performance on OpenAI Gym For the experiments in OpenAI Gym, we evaluate the performance of ERL-TD on six popular environments. Figure 6 displays the mean and standard deviation of cumulative rewards from different seeds. It is evident that ERL-TD achieves outstanding results in all environments. For the maximum cumulative reward, ERL-TD shows significant improvements on Ant-v2 and HalfCheetah-v2. In particular, compared to all other baselines, ERL-TD nearly doubled the maximum cumulative reward on Ant-v2. In contrast, the scores within one million steps on ERL and PDERL is almost none. Additionally, ERL-TD achieves excellent efficiency on Ant-v2, Hopper-v2, and Humanoid-v2. At 20k steps, the cumulative reward of ERL-TD is twice as high as other algorithms on Ant-v2 and Hopper-v2. At 40k steps, the cumulative reward of ERL-TD on Humanoid-v2 and HalfCheetah-v2 are significantly greater than baseline algorithms. Meanwhile, we utilize the same setting of all hyperparameters on all environments, except for the discount factor on swimmer-v2. Our parameters setting is easier than ERL-Re^2. ERL-Re^2 is the state-of-the-art ERL algorithm, while tuning its hyperparameters is challenging.

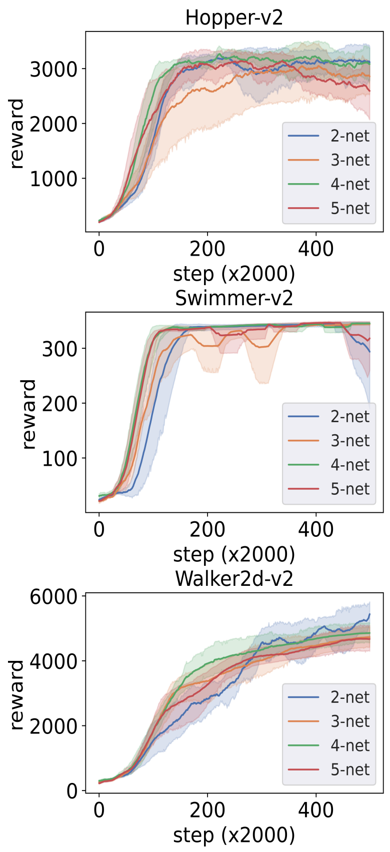
Performance on DMC For the DMC experiments, our algorithm ERL-TD with DrQ is compared with six baseline RL algorithms on six environments in Table 1. At 100k steps, COE-RAD achieves the highest scores on all environments except for the walker-walk. In the reacher-easy



(a) Mutation



(b) Truncated Variance Bellman Backup



(c) Number of Q-networks

Figure 7: (a) The curves respectively represent two types of mutations: distillation mutation (Distill-Mut) and Gaussian mutation (Gaussian-Mut). (b) The experiment decomposes the truncated variance Bellman backup to separately demonstrate the effect of each component. (c) The experiment is conducted to study the performance impact of the number of Q-networks.

and ball_in_cup-catch experiments, it performs better at 100k steps than other baseline algorithms at 500k steps. This obviously illustrates the excellent sampling efficiency of our algorithm. For 500k steps, our algorithm continues to achieve state-of-the-art results in two environments: reacher-easy and ball_in_cup-catch. For the scores on the remaining environments, our algorithm is close to the other algorithms. In conclusion, our algorithm demonstrates strong robustness.

Ablation Study

To demonstrate the effectiveness of each component, we conduct ablation experiments with one million steps on the OpenAI Gym environments. Specifically, the ablation experiments involve distillation mutation, truncated variance Bellman backup, the number of Q-networks, the methods to reduce overestimation, and setting of critic update α.

Distillation Mutation and Gaussian Mutation One of our main contributions is distillation mutation. The distillation mutation treats the policy considered with high Q-values as the teacher. All policies in the population, except for the elite policy, act as students learning from the teacher. The gradient calculated by the final distillation process serves as the guidance for mutation. The Gaussian mutation is a traditional mutation that adds Gaussian noise to all policies except for the elite policy. It attempts to randomly explore a better policy through this mutation, the advantage of which is to maintain the diversity of policy. In Figure 7(a), we compare the performance of these two methods on three environments: Hopper-v2, Swimmer-v2, and Walker2d-v2. We observe that distillation mutation significantly improves the speed of convergence and obtains higher cumulative discounted rewards than Gaussian mutation.

Truncated Variance Bellman Backup To show the performance of Truncated Variance bellman backup (TV), we compare the results of TV, NV, and NT. NV directly takes the mean of Q-values as the target for updating the current Q-networks, without considering the variance as the target. NT adds the variance of the Q-values to the Bellman operator based on NV but without truncation. As shown in Figure 7(b), we observe that NV obtains the worst performance, which converges very slowly on the Hopper-v2 and Walker2d-v2 environments. It indicates that controlling overestimation is extremely important. NT shows poor performance in all three environments, especially in Swimmer-

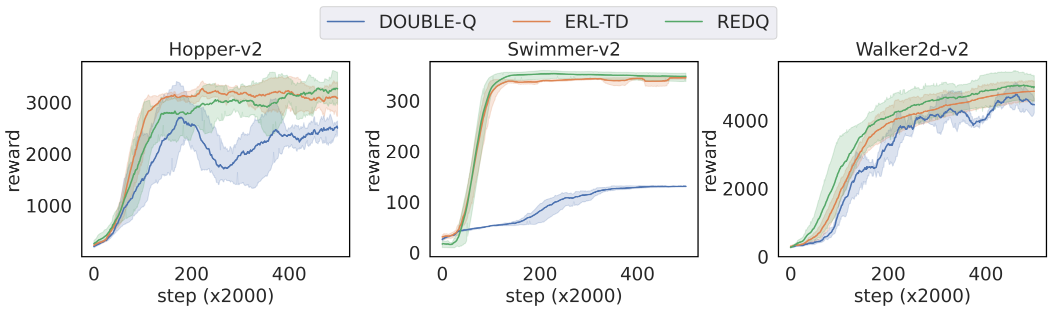


Figure 8: Address the issue of overestimation.

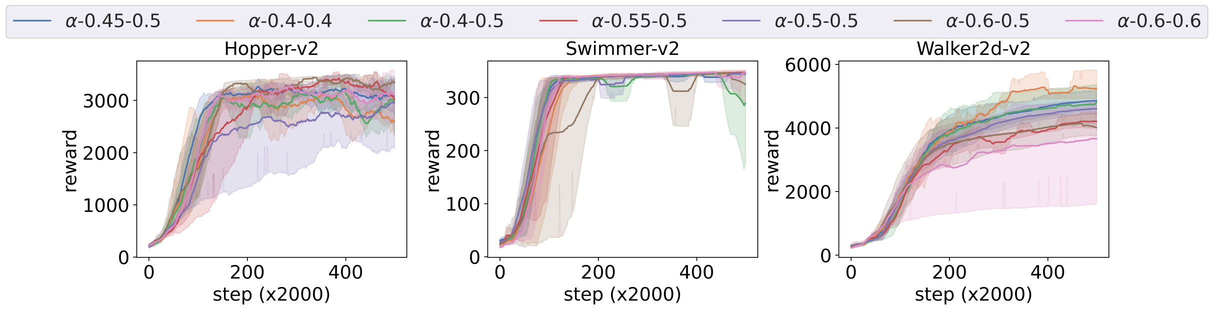


Figure 9: The performance of ERL-TD with different values of α.

v2. This shows that the truncation of variance is crucial, as it can prevent the network from prematurely converging to local optima.

Env	SAC	ERL	ERL-TD	ERL-TD (REDQ)
Hopper	534.52	473.42	853.74	4520.63

Table 2: Time measurements (in seconds) of training 10000 frames, executed on the NVIDIA Geforce RTX 3090.

Number of Q-networks To observe the impact of the number of Q-networks, we conduct an ablation experiment for the number of Q-networks. In Figure 7(c), we vary the number of Q-networks $K \in \{2, 3, 4, 5\}$. ERL-TD shows instability on Swimmer-v2 and Walk2d-v2 when $K = 2$. At the same time, if $K = 3$ or 5, the performance of ERL-TD is poor and unstable on Hopper-v2 and Swimmer-v2. Therefore, considering the performance on multiple environments, we recommend using four Q-networks to evaluate the value of state-action pairs.

Methods to Reduce Overestimation In Figure 8, we compared two methods for selecting the smaller value of two target networks (DOUBLE-Q) and the method used by REDQ, which randomly selects two networks out of 10 as target values. The results show that REDQ performs similarly to ours, but it requires 5 times the amount of time. As shown in Table 2, ERL-TD runs for 10,000 frames on our machine with 853.74 seconds, while our newly implemented REDQ in ERL-TD method takes 4,520.63 seconds to run.

Setting of Critic Update α The mean and standard deviation are on the same scale, so our intuition suggests that the value of α should be set around 0.5. The properties of the exponential function change when the base is equal to one. Therefore, we set different values for α when the base is greater than 1 and less than 1. The impact of α is demonstrated in Figure 9. We use "α-x-y" to describe all legends, where x and y represent the values of α when the base is greater and less than 1, respectively.

Conclusion

In this paper, we propose the ERL-TD algorithm, which leverages the variance of the ensemble networks to mitigate the overestimation of Q-values while preventing underestimation. To enhance the quality of exploration in the policy network, we propose the distillation mutation. On the Gym and DMC environments, we demonstrate that ERL-TD has extremely powerful performance.

As shown in Table 2, the limitation of our algorithm is long running time. The main reason is that using the variance of multiple networks to calculate the gradient consumes a lot of resources. How to use a Q-network to express the Q-values accurately is an exciting research direction in the future. For example, we can control the overestimation of Q-values by utilizing the evaluation of populations in ERL, which is on-policy and does not suffer from overestimation. So it can be combined with off-policy to balance the bias and variance in the estimation of Q-values.

Acknowledgments

This work was supported by the National Natural Science Funds for Distinguished Young Scholar under Grant 62325307, in part by the National Natural Science Foundation of China under Grants 62173236 and 62073225, in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076, and in part by the Shenzhen Science and Technology Program under Grants JCYJ20220531101411027 and 20220809141216003, and in part by the Scientific Instrument Developing Project of Shenzhen University under Grant 2023YQ019.

References

Bäck, T.; and Schwefel, H.-P. 1993. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1): 1–23.

Bodnar, C.; Day, B.; and Lió, P. 2020. Proximal distilled evolutionary reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3283–3290.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Chen, X.; Wang, C.; Zhou, Z.; and Ross, K. W. 2020. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *International Conference on Learning Representations*.

Floreano, D.; Dürr, P.; and Mattiussi, C. 2008. Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1: 47–62.

Fogel, D. B. 2006. *Evolutionary computation: toward a new philosophy of machine intelligence*. John Wiley & Sons.

Fox, R.; Pakman, A.; and Tishby, N. 2015. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Gangwani, T.; and Peng, J. 2018. Policy Optimization by Genetic Distillation. In *International Conference on Learning Representations*.

Grinsztajn, N.; Furelos-Blanco, D.; and Barrett, T. D. 2022. Population-Based Reinforcement Learning for Combinatorial Optimization. *arXiv preprint arXiv:2210.03475*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Hafner, D.; Lillicrap, T.; Ba, J.; and Norouzi, M. 2019a. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019b. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, 2555–2565. PMLR.

Khadka, S.; Majumdar, S.; Nassar, T.; Dwiel, Z.; Tumer, E.; Miret, S.; Liu, Y.; and Tumer, K. 2019. Collaborative evolutionary reinforcement learning. In *International conference on machine learning*, 3341–3350. PMLR.

Khadka, S.; and Tumer, K. 2018. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems*, 31.

Kostrikov, I.; Yarats, D.; and Fergus, R. 2020. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*.

Laskin, M.; Lee, K.; Stooke, A.; Pinto, L.; Abbeel, P.; and Srinivas, A. 2020. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895.

Laskin, M.; Srinivas, A.; and Abbeel, P. 2020. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 5639–5650. PMLR.

Lee, K.; Laskin, M.; Srinivas, A.; and Abbeel, P. 2021. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, 6131–6141. PMLR.

Li, P.; Hao, J.; Tang, H.; Zheng, Y.; and Fu, X. 2023. RACE: Improve Multi-Agent Reinforcement Learning with Representation Asymmetry and Collaborative Evolution. In *International Conference on Machine Learning*, 19490–19503. PMLR.

Li, P.; Tang, H.; Jianye, H.; ZHENG, Y.; Fu, X.; and Meng, Z. 2022. ERL-Re^2: Efficient Evolutionary Reinforcement Learning with Shared State Representation and Individual Policy Representation. In *Deep Reinforcement Learning Workshop NeurIPS 2022*.

Liang, L.; Xu, Y.; McAleer, S.; Hu, D.; Ihler, A.; Abbeel, P.; and Fox, R. 2022. Reducing variance in temporal-difference value estimation via ensemble of deep networks. In *International Conference on Machine Learning*, 13285–13301. PMLR.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lüders, B.; Schläger, M.; Korach, A.; and Risi, S. 2017. Continual and one-shot learning through neural networks with dynamic external memory. In *Applications of Evolutionary Computation: 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part I 20*, 886–901. Springer.

Luo, Y.; Wang, Y.; Dong, K.; Zhang, Q.; Cheng, E.; Sun, Z.; and Song, B. 2023. Relay Hindsight Experience Replay: Self-guided continual reinforcement learning for sequential object manipulation tasks with sparse rewards. *Neurocomputing*, 126620.

Ma, Y.; Liu, T.; Wei, B.; Liu, Y.; Xu, K.; and Li, W. 2022. Evolutionary Action Selection for Gradient-Based Policy Learning. In *International Conference on Neural Information Processing*, 579–590. Springer.

Mitchell, M. 1998. *An introduction to genetic algorithms*. MIT press.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.; Veness, J.; Bellemare, M.; Graves, A.; Riedmiller, M.; Fidjeland, A.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Ni, F.; Hao, J.; Lu, J.; Tong, X.; Yuan, M.; Duan, J.; Ma, Y.; and He, K. 2021. A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3441–3451.

Nikishin, E.; Schwarzer, M.; D'Oro, P.; Bacon, P.-L.; and Courville, A. 2022. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, 16828–16847. PMLR.

Parker-Holder, J.; Pacchiano, A.; Choromanski, K. M.; and Roberts, S. J. 2020. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 18050–18062.

Pourchot, A.; and Sigaud, O. 2019. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *7th International Conference on Learning Representations, ICLR 2019*.

Rey, D.; Hammad, A. W.; and Saberi, M. 2023. Vaccine allocation policy optimization and budget sharing mechanism using reinforcement learning. *Omega*, 115: 102783.

Risi, S.; and Togelius, J. 2015. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1): 25–41.

Sallab, A. E.; Abdou, M.; Perot, E.; and Yogamani, S. 2017. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Sigaud, O. 2022. Combining Evolution and Deep Reinforcement Learning for Policy Search: a Survey. *ACM Transactions on Evolutionary Learning*.

Spears, W. M.; De Jong, K. A.; Bäck, T.; Fogel, D. B.; and De Garis, H. 1993. An overview of evolutionary computation. In *European conference on machine learning*, 442–459. Springer.

Stanley, K. O.; and Miikkulainen, R. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2): 99–127.

Such, F. P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K. O.; and Clune, J. 2018. Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning.

Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; Casas, D. d. L.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690*.

Thrun, S.; and Schwartz, A. 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, volume 6, 1–9.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Zhao, F.; Hu, X.; Wang, L.; Zhao, J.; Tang, J.; et al. 2022. A reinforcement learning brain storm optimization algorithm (BSO) with learning mechanism. *Knowledge-Based Systems*, 235: 107645.

Zhou, M.; Wan, Z.; Wang, H.; Wen, M.; Wu, R.; Wen, Y.; Yang, Y.; Yu, Y.; Wang, J.; and Zhang, W. 2023. MALib: A Parallel Framework for Population-based Multi-agent Reinforcement Learning. *J. Mach. Learn. Res.*, 24: 150–1.