Value-Evolutionary-Based Reinforcement Learning

Pengyi Li[1] **Jianye Hao**[1] **Hongyao Tang**[1] **Yan Zheng**[1] **Fazl Barez**[2 3 4]

Abstract

Combining Evolutionary Algorithms (EAs) and Reinforcement Learning (RL) for policy search has been proven to improve RL performance. However, previous works largely overlook value-based RL in favor of merging EAs with policy-based RL. This paper introduces **V**alue-**E**volutionary-**B**ased **R**einforcement **L**earning (**VEB-RL**) that focuses on the integration of EAs with value-based RL. The framework maintains a population of value functions instead of policies and leverages negative Temporal Difference error as the fitness metric for evolution. The metric is more sample-efficient for population evaluation than cumulative rewards and is closely associated with the accuracy of the value function approximation. Additionally, VEB-RL enables elites of the population to interact with the environment to offer high-quality samples for RL optimization, whereas the RL value function participates in the population's evolution in each generation. Experiments on MinAtar and Atari demonstrate the superiority of VEB-RL in significantly improving DQN, Rainbow, and SPR. Our code is available on https://github.com/yeshenpy/VEB-RL.

1. Introduction

Reinforcement Learning (RL) (Sutton & Barto, 1998) has witnessed growing success in various practical tasks such as game AI (Vinyals et al., 2019), robot control (Johannink et al., 2019), and automatic driving (Zhou et al., 2010; Li et al., 2015). The combination of RL and deep neural networks, known as Deep Reinforcement Learning (DRL), has been instrumental in achieving impressive results. Thanks to the strong approximation capability of neural networks, DRL can efficiently learn using gradient information. However, DRL has limitations, including poor exploration capability (Hao et al., 2023b; Liu et al., 2024b), convergence, and susceptibility to suboptimal policies (Khadka & Tumer, 2018; Li et al., 2024). Moreover, it is highly reliant on the accuracy of the reward signal, low-quality rewards (e.g. sparse, noisy, or deceptive) can drastically reduce performance (Li et al., 2022). Evolutionary Algorithms (EAs) (Bäck & Schwefel, 1993) can be used as an alternative to RL (Such et al., 2017; Salimans et al., 2017) for gradient-free optimization. In contrast to RL, EAs maintain a population instead of a single individual and introduce perturbations to the individuals to produce offspring for better solutions. EAs have shown to be more efficient for exploration, convergence, and robustness compared to RL (Li et al., 2024). Despite these advantages, EAs suffer from low sample efficiency, often requiring several orders of magnitude more samples than RL (Sigaud, 2022; Li et al., 2024).

Many methods are proposed to combine EAs and RL to leverage their complementary strengths for more efficient policy search (Sigaud, 2022; Li et al., 2024). ERL (Khadka & Tumer, 2018) combines Genetic Algorithm (GA) (Katoch et al., 2021) with DDPG (Lillicrap et al., 2016). In ERL, EA and RL are concurrently optimized. EA provides samples generated during the population evaluation process to RL for optimization, aiming to enhance sample efficiency. In turn, RL injects the optimized RL policy into the population to participate in the evolutionary process. ERL defines the fitness of policies in the population as the averaged cumulative reward for several episodes of interaction with the environment, which subsequently became a commonly adopted fitness metric for ERL-related works (Pourchot & Sigaud, 2019; Bodnar et al., 2020; Marchesini et al., 2020; Hao et al., 2023a). Despite the success of ERL and its extensions, previous methods for combining EAs with RL mainly focus on policy-based RL, which employs policies to build the population and ranks individuals based on cumulative rewards obtained from interactions with the environment (Li et al., 2024). However, the value-based RL and its properties are often overlooked in the ERL field.

Applying existing frameworks to value-based RL faces two problems: (i) Value-based algorithms optimize the value function. It is feasible to directly view value functions as policies for evolution through the cumulative rewards, but this ignores the principle of value iteration (Sutton & Barto,

[1]College of Intelligence and Computing, Tianjin University, China [2]Edinburgh Centre for Robotics [3]University of Oxford [4]Centre for the Study of Existential Risk, University of Cambridge . Correspondence to: Jianye Hao <jianye.hao@tju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1998; Mnih et al., 2013), and (ii) low-quality samples generated by the population can potentially lead to suboptimal RL optimization (Zhang et al., 2022; Liu et al., 2021) and reduce sample efficiency. To address the problems, we propose a new framework called **V**alue-**E**volutionary-**B**ased **R**einforcement **L**earning (**VEB-RL**) that combines EAs with value-based RL. VEB-RL maintains a population of Q networks (action value networks) and their corresponding target networks. Instead of using cumulative rewards as the fitness metric for population evaluation, we tailor a new fitness metric, the negative Temporal Difference (TD) error, which quantifies the difference between the Q-network's estimations and the target values. The tailored fitness metric aligns with the goal of value iteration and does not require interaction with the environment, making it more sample-efficient. With the tailored fitness metric, we can evaluate individual fitness and rank individuals without interaction. To offer high-quality samples and avoid the potential negative impacts of low-quality samples on RL optimization, we propose a new mechanism *Elite Interaction*, which restricts interaction with the environment to only the elite individuals of the population. Our main contributions are three-fold:

- First, we propose a simple yet efficient framework VEB-RL to enhance Value-Based RL. Considering the properties of value-based RL, we propose to use the negative TD error as the fitness. To the best of our knowledge, we are the first to use the negative TD error as the fitness for population evolution.
- Second, we propose *Elite Interaction* to guarantee the quality of the data samples and improve sample efficiency.
- Last but not least, we propose two variants of VEB-RL and integrate them with DQN, Rainbow, and SPR. The experiments on MinAtar, Atari, and Atari 100k show that VEB-RL significantly improves the RL algorithms and outperforms other strong ERL baselines. Additionally, VEB-RL can improve non-value-based RL and demonstrate its effectiveness on MUJOCO tasks.

2. Preliminaries

Reinforcement Learning Consider a Markov decision process (MDP) (Puterman, 1990), defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, T \rangle$. At each step t, the agent uses a policy π to select an action $a_t \sim \pi(s_t) \in \mathcal{A}$ according to the state $s_t \in \mathcal{S}$ and the environment transits to the next state s_{t+1} according to transition function $\mathcal{P}(s_t, a_t)$ and the agent receives a reward $r_t = \mathcal{R}(s_t, a_t)$. The return is defined as the discounted cumulative reward, denoted by $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ where $\gamma \in [0, 1)$ is the discount factor and T is the maximum episode horizon. Value-based RL first learns optimal value function $Q^*_\theta(s, a)$ and then gets optimal policy according to $Q^*_\theta(s, a)$, i.e., $\pi^* = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$.

DQN (Mnih et al., 2013) is a representative off-policy value-based algorithm consisting of a Q network Q_θ and a target network $Q_{\theta'}$ with the parameters θ and θ' respectively. The Q network is optimized with the Temporal Difference (TD) (Sutton & Barto, 1998) loss, which is defined as: $\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}}[(r + \max_{a' \sim \mathcal{A}} \gamma Q_{\theta'}(s', a') - Q_\theta(s, a))^2]$, where the experiences (s, a, r, s') are sampled from the replay buffer $\mathcal{D}$. Rainbow (Hessel et al., 2018) incorporates different variants of DQN, i.e., double DQN (van Hasselt et al., 2016), Prioritised Experience Replay (Schaul et al., 2016), Dueling Network Architecture (Wang et al., 2016), Multi-step Returns (Sutton & Barto, 1998), Distributional RL (Bellemare et al., 2017) and Noisy Net (Fortunato et al., 2018), which achieves a quantum jump in performance. Besides, some works further improve the sample efficiency through representation or model-based approaches (Kaiser et al., 2019; Kielak, 2019; Laskin et al., 2020).

Evolutionary Algorithm Evolutionary Algorithms (EAs) (Bäck & Schwefel, 1993; Bäck, 1996; Vikhar, 2016; Zhou et al., 2019) are a class of gradient-free optimization methods. EAs maintain a limited population of individuals and generate new individuals randomly centered around the elite individuals of the previous generation. The individuals in the population are typically defined as policies (i.e., actor networks) $\mathbb{P} = \{\pi_1, \pi_2, ..., \pi_n\}$. The individuals are evaluated by the fitness metric, and those with higher fitness are more likely to be selected as elite individuals. The fitness metric to evaluate the individuals is typically defined as the cumulative total reward of an individual's interaction with the environment for e episodes $\{f(\pi_1), f(\pi_2), ..., f(\pi_n)\}$ where $f(\pi_i) = \frac{1}{e}\sum_{i=1}^{e}[\sum_{t=0}^{T} r_t \mid \pi_i]$. EAs encompass various methods. Here, we primarily introduce two widely used methods in the ERL-related works: Genetic Algorithm (GA) (Mitchell, 1998) and Cross-Entropy Method (CEM) (Pourchot & Sigaud, 2019).

GA (Lambora et al., 2019; Katoch et al., 2021) is the classical evolutionary algorithm that consists of three primary steps: *Evaluation*, *Selection*, *Variation*. First, GA evaluates the population and ranks the individuals in the population by fitness. Subsequently, GA selects the parents according to the selection mechanism such as tournament selection (Khadka & Tumer, 2018). Finally, the parents π_i and π_j are selected to produce offspring π'_i and π'_j by performing the crossover operator, i.e., $\pi'_i, \pi'_j = \texttt{Crossover}(\pi_i, \pi_j)$ or the mutation operator $\pi'_i = \texttt{Mutation}(\pi_i)$. The crossover and mutation usually operate at the parameters of the networks. Typically, k-point crossover randomly exchanges segment-wise (network) parameters of parents while Gaussian mutation adds Gaussian noise to the parameters to randomly generate offspring.

CEM (Boer et al., 2005) is an evolutionary strategy

(ES) (Beyer & Schwefel, 2002) based on the Estimation of Distribution Algorithm (EDA) (Larrañaga & Lozano, 2001). In CEM, the population is represented as a distribution using a covariance matrix $\mathbf{\Sigma}$. It retains a single individual from one generation to the next, serving as the mean μ of the distribution from which new individuals are drawn. As the optimization iterates, the distribution $\mathbf{\Sigma}$ progressively shifts towards the local optimum. Specifically, CEM initially samples the population by adding Gaussian noise to the mean of the distribution, denoted as $x_i \sim \mathcal{N}(\mu, \Sigma)$. The individuals within the population are then evaluated for the fitness $\{f(x_1), f(x_2), ..., f(x_n)\}$, and the top half of the best-performing individuals $\{z_1, ..., z_{\frac{n}{2}}\}$ are employed to update the CEM distribution as outlined below:

$$\begin{aligned} \mu_{new} &= \sum_{i=1}^{|\mathbb{P}|/2} \lambda_i z_i; \\ \Sigma_{new} &= \sum_{i=1}^{|\mathbb{P}|/2} \lambda_i \left(z_i - \mu_{old}\right) \left(z_i - \mu_{old}\right)^T + \epsilon \mathcal{I}, \end{aligned} \tag{1}$$

where λ_i represents the weight assigned to individual i.

3. Related Work

The emerging research field that explores the synergy between EA and RL has gained significant attention recently (Sigaud, 2022; Zhu et al., 2023; Li et al., 2024). Several notable works have been proposed in this context. ERL (Khadka & Tumer, 2018) first proposes a novel hybrid framework, where an actor-critic RL agent and a GA population are optimized concurrently. The RL agent and GA population benefit from each other in a manner where the diverse experiences generated by the population evaluation are provided to the RL replay buffer for gradient optimization. Simultaneously, the evolution of the population is accelerated by injecting the optimized RL actor. PDERL (Bodnar et al., 2020) follows the framework of ERL and improves the genetic operators to solve the catastrophic forgetting problems. Supe-RL (Marchesini et al., 2020) periodically searches around the current RL actor in parameter space and evaluates the generated policies. Then, the RL actor performs a soft update toward the best-performing policy. CEM-RL (Pourchot & Sigaud, 2019) combines CEM and TD3 algorithms. It uses the gradient information of RL critic to update half of the individuals in the population, thereby influencing the distribution of CEM. ERL-Re^2 (Hao et al., 2023a) decouples the policies into the shared non-linear state representations and individual linear policy representations, aiming to share knowledge and reduce the exploration space. RACE (Li et al., 2023) employs the same idea as ERL-Re^2, combining EAs with multi-agent RL to facilitate collaboration through the construction of efficient shared observation representations. CCQD (Xue et al., 2023) maintains multiple shared state representations to improve the sample efficiency of quality diversity algorithms. These algorithms leverage the cumulative rewards for interaction with the environment as the fitness metric to evaluate the population, ignoring the value function approximation. Besides, these algorithms require all individuals in the population to interact with the environment and generate experiences. However, experiences of low quality may negatively impact the optimization of RL (Nikishin et al., 2022). These limitations make it difficult to further enhance value-based RL, which constitutes the primary focus of our work. If readers want to learn more about the integration of EAs and RL, please refer to surveys (Sigaud, 2022; Zhu et al., 2023; Li et al., 2024) for more details.

4. Value-Evolutionary-Based RL

This section presents VEB-RL, a hybrid framework leveraging EAs to enhance Value-based RL. VEB-RL comprises two fundamental components: 1) A tailored fitness metric designed for integration with Value-based RL. 2) *Elite Interaction* mechanism utilized to enhance the quality of samples generated by the population. We first detail these components, following which we introduce two distinct variants: GA-based VEB-RL and CEM-based VEB-RL.

4.1. Tailored Fitness Metric

Previous methods that integrate EAs and Policy-Based RL have two main characteristics. Firstly, populations are composed of policy networks (i.e., actors), and secondly, the cumulative reward interaction with the environment is used as the fitness metric to evaluate the population. However, when combining EAs with value-based RL, the evolving individuals become value networks. In this context, utilizing the fitness metric based on cumulative rewards for population evolution might not be an optimal choice, as value-based RL involves the explicit modeling of expected values. Utilizing cumulative rewards as the fitness metric disregards the approximation of the value function, potentially resulting in inaccurate value approximations.

Taking Deep Q Network (DQN) as a representative, the optimization objective is to get the optimal Q network Q_θ^* to accurately estimate the state-action values by value iteration through the *Bellman Equation*. The loss function of Q is defined as:

$$L(\theta) = \mathbb{E}_{s,s',a\sim\mathcal{D}} \left[\left(r + \max_{a'} Q_{\theta'}\left(s', a'\right) - Q_\theta\left(s, a\right) \right)^2 \right], \tag{2}$$

where s, s', a are sampled from the replay buffer $\mathcal{D}$, $Q_{\theta'}$ is the target Q network. The optimal policy is to take a set of actions a that maximizes the expected value of $Q^*(s, a)$. It is evident that the optimization objective of value-based RL is to acquire an accurate value function, whereas the

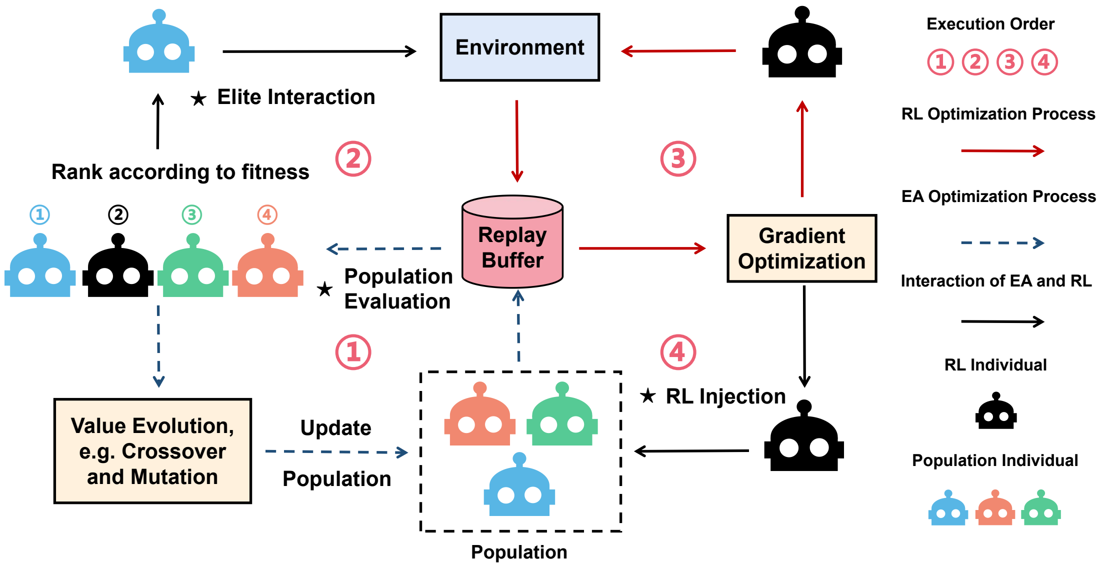


Figure 1. The conceptual illustration of Value-Evolutionary-Based RL. In VEB-RL, a population of value function networks and an RL individual are maintained. Each generation follows the execution sequence indicated by the pink numbers, including: 1) Evaluating and evolving the population. 2) Performing Elite Interaction and storing samples in the replay buffer. 3) Conducting RL interaction and optimizing based on the replay buffer. 4) Injecting the optimized RL individual into the population.

optimization objective of the fitness function based on cumulative rewards is to enhance the policy derived from the value function, which is inconsistent with the optimization objective of value functions. This misalignment could potentially make it difficult to attain a sufficiently accurate value function. We experimentally validate this point in Section 5.3.

To solve the problem, we propose a new population composition and a tailored fitness metric for population evaluation. To be concrete, VEB-RL maintains a population of Q networks and their corresponding target networks, denoted as $\mathbb{P} = \{Q_{\theta_i}, Q_{\theta_i'}\}_{i=1}^{n}$, which provide the foundation for the fitness calculation. To steer population evolution toward more accurate value functions, the fitness metric for evaluating individuals within the population is defined as the negative Temporal Difference error.

$$f\left(\theta_i, \theta_i'\right) = -\mathbb{E}\left[\left(r + \max_{a'} Q_{\theta_i'}\left(s', a'\right) - Q_{\theta_i}\left(s, a\right)\right)^2\right], \quad (3)$$

where s, a, s' are sampled from the replay buffer $\mathcal{D}$, $Q_{\theta_i}, Q_{\theta_i'}$ are sampled from the population $\mathbb{P}$. When the samples cover all possible transitions, the smaller the TD error, the more accurate the value function approximation becomes. Since we can not access all state transitions, we sample a sufficiently large number of instances, sized M, from the replay buffer to approximate the true error. The Q-value approximation becomes more accurate as $f(\theta_i, \theta_i')$ increases. Specifically, the state-action value functions $\{Q_{\theta_i}\}_{i=1}^{n}$ in the population are optimized by EAs, namely GA and CEM.

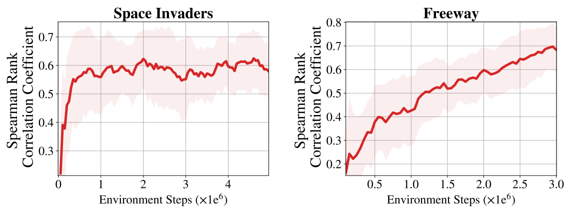


Figure 2. The Spearman rank correlation analysis. The results indicate that the individual rankings based on our fitness metric are highly similar to the rankings obtained through interactions.

The target networks $Q_{\theta_i'}$ is hard updated by its corresponding Q network Q_{θ_i} every H generations. Moreover, we maintain the RL injection mechanism: $Q_{\theta_{rl}}$ optimized by RL, and its target network $Q_{\theta_{rl}'}$ are injected into the population every generation. If the RL individual obtains a higher $f(\theta_{rl}, \theta_{rl}')$ value and is selected as the elite, it will guide the population and facilitate the evolution of the population.; otherwise, $Q_{\theta_{rl}}$ and $Q_{\theta_{rl}'}$ are eliminated.

To demonstrate the effectiveness of this tailored fitness metric, we conduct the Spearman rank correlation analysis (Sedgwick, 2014) on *Space Invader* and *Freeway* tasks to compare the rankings of individuals based on the negative TD error fitness and the rankings derived from cumulative total rewards over 5 episodes of interaction with the environment. The Spearman rank correlation coefficient disregards specific values and focuses solely on the relative order of values. It ranges from -1 to 1, where a value closer to 1

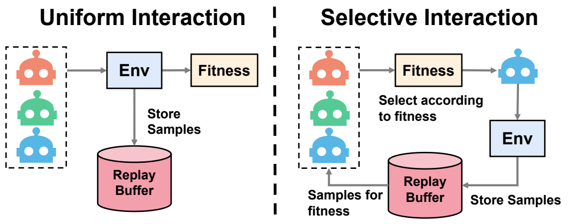


Figure 3. Uniform Interaction vs. Selective Interaction. Uniform Interaction refers to all individuals interacting with the environment to obtain fitness. Selective Interaction refers to actively choosing individuals to interact with the environment based on their fitness.

signifies a higher degree of similarity in rankings, while a value closer to -1 indicates a greater disparity in rankings. The results depicted in Figure 2 illustrate a notable and increasing correlation between the population rankings based on the two fitness metrics throughout the training process. The results highlight that utilizing negative TD error as a fitness metric can relatively accurately reflect the quality of individuals without sample cost, which enhances sample efficiency.

4.2. Elite Interaction

Previous ERL works require all individuals in the population to interact with the environment for fitness (Li et al., 2024) and we refer to this process as *Uniform Interaction*. In these works, a crucial mechanism for enhancing RL is incorporating the samples generated through *Uniform Interaction* into a shared replay buffer for RL optimization. This mechanism mitigates the problem of RL's poor exploration capabilities and avoids the wastage of evaluation samples. However, this introduces new problems: only samples of high-quality individuals contribute to RL policy optimization, while those generated by individuals with low fitness may be invalid and could potentially lead RL into suboptimal solutions (Zhang et al., 2022; Liu et al., 2021).

Thanks to the new fitness metric, individuals in the population no longer need to interact directly with the environment for fitness. Instead, they calculate fitness based on samples from the replay buffer. As a result, we can actively select which individuals interact with the environment to generate samples. We refer to this way as *Selective Interaction*. The illustration of *Uniform Interaction* versus *Selective Interaction* is shown in Figure 3. To resolve the aforementioned problem of *Uniform Interaction*, we propose a new way of interaction according to the concept of *Selective Interaction* called *Elite Interaction* that allows only elite individuals to interact with the environment to generate high-quality samples. To be concrete, we can first obtain the fitness of individuals in the population $\{f(\theta_i, \theta_i')\}_{i=1}^n$ according to Eq. 3. Subsequently, we choose the top N individuals

Algorithm 1 Value-Evolutionary-Based RL

```
Option: Select GA or CEM
Initialize: a shared replay buffer $\mathcal{D}$, The RL Q network $Q_{\theta_{rl}}$ and
its target network $Q_{\theta'_{rl}}$, the population $\mathbb{P} = \{Q_{\theta_i}, Q_{\theta'_i}\}_{i=1}^n$ with
size $n$, if select CEM, define the covariance matrix $\Sigma = \sigma_{\text{init}} \mathcal{I}$
repeat
    # ① EA Optimization: Population Evaluation and Evolution
    if Select CEM then
        Draw the current population $\mathbb{P}$ from $\mathcal{N}(\mu, \Sigma)$ to replace
        the old $\mathbb{P}$ except the injected RL individual.
    Population Evaluation: Sample data with size $M$ from $\mathcal{D}$ to
    compute the fitness $\{f(\theta_i, \theta_i')\}_{i=1}^n$ by Eq. 3.
    Elite Identification: Select the top $N$ individuals with the
    highest fitness as the elites.
    if Select CEM then
        Update $\pi_\mu$ and $\Sigma$ with the top half of $\mathbb{P}$ by Eq. 1.
    else
        Perform Crossover Operator:
        (1): Select the winners by Tournament Selection and others who are not selected as elites or winners are recorded
        as abandoners.
        (2): Select two individuals from elites and winners as
        parents respectively.
        (3): The two parents generate offspring by crossover operator to replace abandoners.
        Repeat the above process until all abandoners are replaced.
        Perform Mutation Operator:
        All non-elite individuals mutate (Add Gaussian noise to
        the parameters) according to a certain probability.
    # ② Elite Interaction:
    The elites interact with the environment and store the experiences generated to $\mathcal{D}$.
    # ③ RL Optimization: Interaction and Gradient Optimization
    RL Interaction: The RL agent interacts with the environment
    and stores the experiences generated to $\mathcal{D}$.
    Optimize the RL Agent: Update $Q_{\theta_{rl}}$ by Eq. 2.
    # ④ RL Injection:
    Inject the optimized $Q_{\theta_{rl}}$ and $Q_{\theta'_{rl}}$ into the population $\mathbb{P}$.
    Update $Q_{\theta'_{rl}}$ periodically.
    Update $Q_{\theta'_i}$ in $\mathbb{P}$ with $Q_{\theta_i}$ every $H$ generations.
until reaches maximum training steps;
```

to interact with the environment. Intuitively, *Elite Interaction* enables well-performing individuals to contribute high-quality samples to the replay buffer $\mathcal{D}$, while poorly performing individuals do not waste resources for interaction and avoid the potential negative impact of low-quality samples on the RL optimization process.

4.3. The Algorithm Framework of VEB-RL

Thanks to the new fitness metric and *Elite Interaction*, EA and RL in VEB-RL are better fused for value function search with high sample efficiency. Figure 1 depicts the VEB-RL architecture. For the evolutionary process, the population obtains fitness based on the samples in the replay buffer $\mathcal{D}$ (Eq. 3). The top N individuals are selected as elite individuals and interact with the environments to provide high-quality samples to the replay buffer $\mathcal{D}$ for RL optimization. Subsequently, we evolve the population with EA to find better Q networks and construct the new population. Regarding the reinforcement process, the RL individual interacts with the environment and stores the experiences in the replay buffer $\mathcal{D}$. Then RL optimizes its Q network based on the shared replay buffer by Eq. 2. The RL target network is hard updated periodically. To promote population evolution, the optimized $Q_{\theta_{rl}}$ and its target network $Q_{\theta'_{rl}}$ are injected into the population every generation. The target networks in the population are updated by their corresponding Q networks every H generations for stable improvement, which is similar to what conventional RL does.

In principle, VEB-RL is a general framework that can combine with arbitrary EAs. We implement the evolutionary process using two distinct EAs: GA and CEM. The pseudocode of VEB-RL is shown in Algorithm 1. In the case of GA-based VEB-RL, evolution involves the select, crossover, and mutation operators. The top N individuals are selected as *elites*. Next, we employ Tournament Selection to select *winners*, which involves picking the best individual from a set of three randomly selected individuals in each iteration and repeating this process a certain number of times. The individuals not chosen as *elites* or *winners* are recorded as *abandoners*. For the crossover operator, we randomly select two individuals from *elites* and *winners* as the parents to produce offspring using k-point crossover. The offspring replaces the *abandoners* until all *abandoners* are replaced. For the mutation operator, all non-elite individuals are added with Gaussian noise according to a certain probability. The population is updated through crossover and mutation. For CEM-based VEB-RL, evolution mainly involves the distribution update. the top half of the population update the π_μ and Σ acorrding to Eq. 1. Besides, the new population draws form $\mathcal{N}(\mu, \Sigma)$ at the beginning of each generation.

Overall, we provide the technical details of GA-based VEB-RL and CEM-based VEB-RL, along with how to combine VEB-RL with value-based RL algorithms. We then empirically evaluate these approaches in the following section.

5. Experiments

This section empirically evaluates VEB-RL on a range of tasks. Then, we provide analysis for a better understanding of VEB-RL. Moreover, a detailed ablation study is conducted to verify the effectiveness of each component.

5.1. Benchmarks & Baselines

We first consider MinAtar benchmark (Young & Tian, 2019) which is a testbed of miniaturized versions of several Atari games. We build GA-based VEB-RL, CEM-based VEB-RL on DQN and evaluate them on all five tasks in MinAtar benchmark, comparing VEB-RL with three advanced ERL baselines in DQN version: ERL (Khadka & Tumer, 2018), CEM-RL (Pourchot & Sigaud, 2019), ERL-Re^2 (Hao et al., 2023a) where ERL-Re^2 is currently the state-of-the-art (SOTA) algorithm in the field of ERL. For a comprehensive study, we also consider Atari benchmark (Mnih et al., 2013) which is a demanding testbed: not only are the inputs high-dimensional, but the game visuals and game mechanics also vary substantially between games. We build GA-based VEB-RL, CEM-based VEB-RL on Rainbow that combines the improvements of various DQN variants. We evaluate them on six popular tasks in Atari to verify whether VEB-RL can further enhance Rainbow. Besides, we also integrate VEB-RL (GA) with SPR (Schwarzer et al., 2021) and evaluate them on nine tasks in Atari 100k. For all baselines, we use the official codes. For a fair comparison, the network architecture (i.e., DQN and Rainbow) used in associated baselines are the same. For all experiments, we give each baseline the same environment steps. Hyperparameters are fine-tuned on all environments.

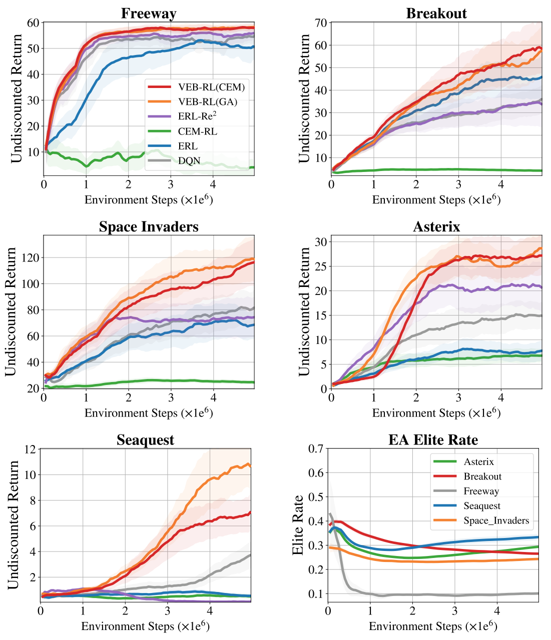


Figure 4. Performance comparison of GA-based VEB-RL, CEM-based VEB-RL and baselines (all in DQN version) in MinAtar and the EA elite rate in all tasks.

All statistics are obtained based on five independent runs with the same seeds. We report the average with 95% confidence regions. For the hyperparameters specific to VEB-RL, N is selected from $\{1, 2, 3, 5\}$. We set the population size to 10 in all tasks for all algorithms, which is consistent with the setting of the previous methods (Khadka & Tumer, 2018; Pourchot & Sigaud, 2019). All implementation details are provided in Appendix A.

	Qbert	Pong	Breakout	Boxing	Alien	Freeway	Seaquest	Asterix	Battle Zone
CURL	1042.4	-16.5	4.9	1.2	558.2	26.7	384.5	734.5	14870.0
OTRainbow	509.3	1.3	9.8	2.5	824.7	25.0	286.9	628.5	4060.6
SimPLe	1288.8	**12.8**	16.4	9.1	616.9	20.3	**683.3**	1128.3	5184.4
SPR	3678.4	-4.2	13.8	25.5	783.3	25.5	541.7	882	9190.0
VEB-SPR	**4099.1**	10.3	**22.2**	**43.1**	**882.7**	**30.6**	677.5	**1201**	**15505.0**

Table 1. Average Performance comparison of GA-based VEB-SPR and other strong methods on Atari 100k.

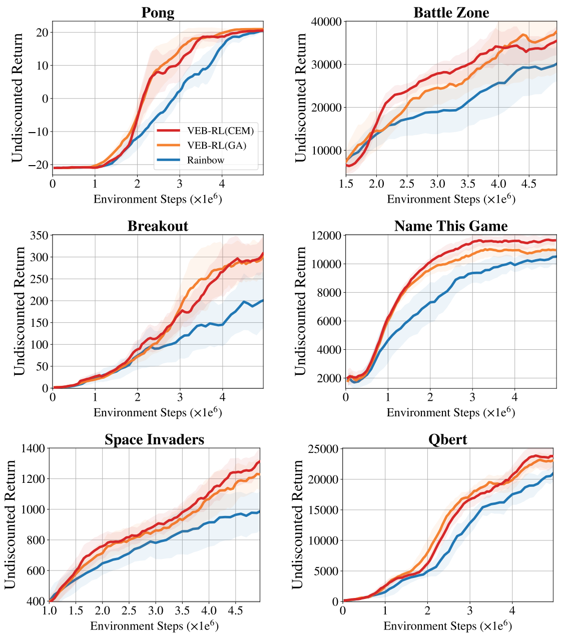
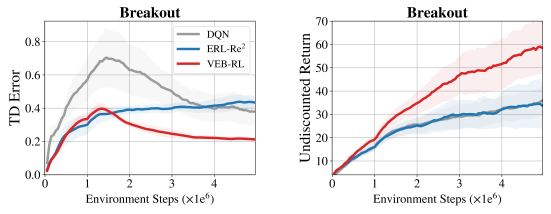
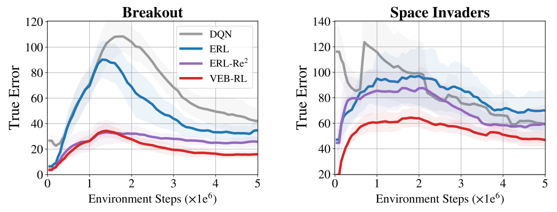


Figure 5. Performance comparison of GA-VEB-RL (Rainbow), CEM-VEB-RL (Rainbow) and Rainbow in Atari.



(a) The curves of the TD error vs. performance



(b) Comparison of true error

Figure 6. Analysis on the new fitness metric.

5.2. Performance Evaluation

We first evaluate the performance on all five tasks of MinAtar: *Breakout*, *Asterix*, *Freeway*, *Space Invaders* and *Seaquest*. All comparison algorithms are built on DQN. Comparisons on the averaged undiscounted return in Figure 4 show that both GA-based VEB-RL and CEM-based VEB-RL significantly improve DQN and outperform other methods across all tasks. Specifically, VEB-RL brings about 100% the performance improvement on *Asterix* and *Seaquest*, about 50% on *Breakout* and *Space Invaders*.

We further verify whether CEM-VEB-RL and GA-VEB-RL can further improve Rainbow on six popular tasks of Atari: *Breakout*, *Space Invaders*, *Qbert*, *Pong*, *Battle Zone* and *Name This Game*, in which agents need to take the high-dimensional pixel images as inputs. The results in Figure 5 show that both GA-VEB-RL and CEM-VEB-RL can significantly improve Rainbow in terms of sample efficiency, which demonstrates the benefits of VEB-RL in challenging high-dimensional control tasks.

To further demonstrate the efficiency of VEB-RL, we conduct additional experiments that combine VEB-RL with SPR (Schwarzer et al., 2021), a SOTA algorithm that achieves superior performance on Atari using data augmentation and representation learning within a 100,000 environment step limit. We implement GA-Based VEB-RL on the official codebase of SPR and evaluate the framework on nine popular tasks with five runs each. Our experimental results presented in Table 1 indicate that GA-based VEB-RL significantly improves SPR by approximately 40% and outperforms other strong baselines (Kaiser et al., 2019; Kielak, 2019; Laskin et al., 2020) on seven out of the nine tasks.

Overall, the experiments show that VEB-RL is an efficient framework that can be integrated with different value-based RL algorithms and provides significant improvement.

5.3. Analysis of VEB-RL

We further analyze VEB-RL from two aspects: 1) Does the new fitness metric adopted in VEB-RL accurately reflect an individual's performance? 2) Can VEB-RL achieve a more accurate value function?

To verify whether the fitness metric is related to the performance, i.e., the cumulative rewards, we plot the curves of the TD error calculated by 5120 samples vs. performance during the training process. The trends in TD error and corresponding performance are shown in Fig. 6(a) with the same color curves. The different color curves represent the training results with different methods. We observe that TD error grows at the beginning of training, and then the TD error and performance show a negative correlation. The growth in the beginning stage is mainly due to incomplete state coverage. When the state cover is relatively comprehensive, two laws can be observed: (1) The lower the TD error, the higher the performance obtained in the same color curves. (2) The lower the TD error curve, the better the performance obtained (Comparison of different methods). These phenomena demonstrate that using the negative TD error as the fitness can be an alternative to the performance obtained by interacting with the environment.

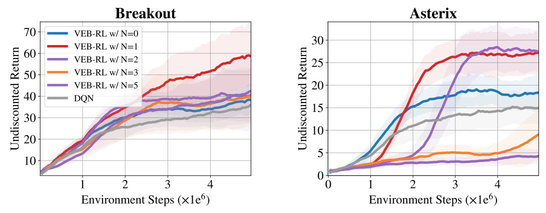
To verify whether VEB-RL can obtain a more accurate value function, we measure the difference between the ground truth values and the estimates of Q networks optimized by different methods. The ground truth values are obtained by running the current learned policies for 1000 episodes and computing the cumulative discounted rewards for all states. The estimates are the maximum Q value estimated by the current learned Q network based on the states collected from 1000 episodes. We take the mean squared difference of the truth values and estimates to reflect the accuracy of the approximation. We select two tasks *Breakout* and *Space Invaders* for verification. The results in Fig. 6(b) show that VEB-RL can obtain smaller errors on both tasks, which demonstrates VEB-RL can help to obtain a more accurate value function than other methods.

Furthermore, we analyze the contributions of EA, i.e., the probability of EA being selected as the elite. The elite rates demonstrate the role of EA in finding more accurate value functions. We show the elite rate of GA-based VEB-RL on the last figure of Fig. 4. We observe that RL has an elite rate of approximately 70% while EA 30% on four of five tasks, where VEB-RL brings significant performance improvements. This demonstrates that EA plays an important role in value function search.

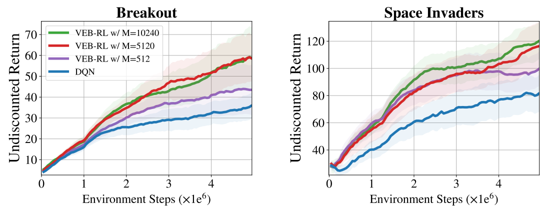
In addition, we compare VEB-RL with a population-based pure RL algorithm in Appendix A.3. The results show that VEB-RL significantly outperforms the population-based pure RL algorithm, indicating that VEB-RL's advantages are not due to simply the introduction of the population. For more details, please refer to Appendix A.3.

5.4. Hyperparameter Analysis

In this section, we analyze the hyperparameters specific to VEB-RL, while keeping the other hyperparameters consistent with those of ERL and CEM-RL. More details can be found in Appendix A.2.



(a) Hyperparameter analysis on N



(b) Hyperparameter analysis on M for the fitness

Figure 7. Analysis of the Hyperparameter in MinAtar.

We analyze the hyperparameter N to investigate how the number of the top individuals interacting with the environment in *Elite Interaction* affects performance. The results in Figure 7(a) demonstrate that performance gradually declines as N increases. This phenomenon is primarily attributed to the fact that, with larger N, poor-performing individuals in the population generate samples for RL optimization. These low-quality samples do not contribute positively to RL optimization; instead, they result in additional sample overhead and may potentially lead RL into suboptimal solutions. Hence, smaller values of N are favored in most tasks. Additionally, it is noticeable that VEB-RL, without utilizing EA to provide elite experiences for RL optimization (i.e., when $N = 0$), slightly outperforms DQN but falls significantly behind VEB-RL with *Elite Interaction*. The performance improvement with $N = 0$ primarily arises from EA's capability to search for better value functions, thus providing better performance. This highlights the efficiency of EA in value function search. However, this hinders the path for EA to facilitate RL improvement, making it inefficient. In contrast, incorporating *Elite Interaction* can provide high-quality experiences to RL, leading to a significant improvement in performance. This further highlights the importance of *Elite Interaction*.

In VEB-RL, it should be noted that RL requires interaction with the environment. When RL is prohibited from interacting with the environment, the distributional shift issue may arise during the training process, resulting in performance collapse. We support this statement with experimental evidence and delve into the underlying reasons through detailed analysis in Appendix A.3.

	TD3	ERL	CEM-RL	PDERL	VEB
Ant	6219	6242	5331	5331	**7356**
Humanoid	6362	6252	213	6733	**6940**
Walker	4806	4774	5015	5015	**5490**

Table 2. Average Performance comparison of GA-based VEB-TD3 and other baselines on MUJOCO tasks.

To ensure accurate fitness evaluation, the sample size M for the fitness should be sufficiently large. However, utilizing samples from the entire replay buffer would lead to excessive computational overhead. Thus we set M to 5120 for all MinAtar tasks. We offer an analysis of the sample size M in the MinAtar tasks. The results depicted in Fig. 7(b) demonstrate that 5120 samples prove to be sufficient, and in certain tasks, a larger sample size could potentially lead to additional performance gains. Conversely, when the sample size is small, there is a risk of performance degradation. This stems from the challenge that the fitness calculations based on small samples might not accurately reflect the ranking of individuals. Due to the more complex visual nature of Atari tasks and the larger network architecture (e.g., Rainbow), we set a smaller M of 1024 for Atari tasks to reduce the computational burden.

5.5. Integration with Actor-Critic methods

The above experiments primarily involve integrating with value-based methods. However, if VEB-RL is restricted to integrating solely with value-based RL methods, it could limit its applications. Thus we combine VEB-RL with actor-critic methods to validate the generalization of VEB-RL. Specifically, we combine VEB-RL with TD3 (Fujimoto et al., 2018). In this setting, VEB maintains a population of Double Q networks and their corresponding target Q networks. Each individual in the population is paired with an actor. Only individuals with small TD errors can optimize their corresponding actors using the Q network. As for elite interaction, only actors corresponding to the top N ranked individuals are allowed to interact with the environment, providing high-quality samples. We evaluate VEB-TD3, TD3, and other ERL-related methods on the Ant, Walker, and Humanoid tasks, which are well-known in ERL-related research. Each experiment is repeated with five runs and 3 million environment steps. The results in Figure 2 show that VEB-RL outperforms TD3 and CEM-RL, ERL, and PDERL, which further demonstrates the efficiency and generalization of VEB-RL.

6. Conclusion

Our work primarily strives to address the research gap in combining EAs with value-based RL within the ERL domain. We present a simple yet efficient framework known as VEB-RL. VEB-RL tailors a new fitness metric, the negative Temporal Difference error, and introduces the Elite Interaction to improve sample efficiency. Experimental results demonstrate the effectiveness of using TD error as an alternative to cumulative rewards and highlight the benefits of Elite Interaction. We integrate VEB-RL with various value-based RL algorithms. Extensive evaluations on challenging benchmarks, including MinAtar, Atari, and Atari 100k, show that VEB-RL significantly outperforms other baseline methods. Additionally, we demonstrate that VEB-RL can be combined with other non-value-based RL algorithms, leading to significant improvements on MUJOCO tasks. This further illustrates the efficiency and generalization of VEB-RL.

7. Limitations & Future Work

For limitations and future works, VEB-RL introduces additional time and resource costs, we provide detailed analysis in Appendix A.3. Moreover, VEB-RL lacks theoretical support, as we only demonstrate its efficiency through experiments without providing optimal theoretical guarantees. This is a challenge that needs to be addressed in future research. Additionally, VEB-RL is an initial solution for value-based RL, but there is room for improvement in the current architecture design (where each individual maintains a Q network and target network) and fitness metric design (sampling a sufficiently large sample for evaluation). For instance, we discover some alternative structural designs that could lead to better performance, as illustrated in Table 8. Furthermore, new mechanisms could be devised to ensure a more comprehensive sample coverage for the current fitness evaluation, rather than random sampling (Liu et al., 2024a), or to design a fitness metric that is more conducive to enhancing value-based RL.

Impact Statement

This paper presents work whose goal is to advance the field of Machine learning. There are many potential social consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant Nos. 92370132, 62106172), the National Key R&D Program of China (Grant No. 2022ZD0116402) and the Xiaomi Young Talents Program of Xiaomi Foundation.

References

Bäck, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. 1996.

Bäck, T. and Schwefel, H. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1993.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *ICML*, 2017.

Beyer, H. and Schwefel, H. Evolution strategies–a comprehensive introduction. *Nat. Comput.*, 2002.

Bodnar, C., Day, B., and Lió, P. Proximal distilled evolutionary reinforcement learning. In *AAAI*, 2020.

Boer, P. D., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. A tutorial on the cross-entropy method. *Ann Oper Res*, 2005.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. In *ICLR*, 2018.

Fujimoto, S., v. Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.

Hao, J., Li, P., Tang, H., Zheng, Y., Fu, X., and Meng, Z. Erl-re^2: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation. In *ICLR*, 2023a.

Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P., and Wang, Z. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *TNNLS*, 2023b.

Hessel, M., Modayil, J., Hasselt, H. V., Schaul, T., Ostrovski, G., W.Dabney, Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., and Levine, S. Residual reinforcement learning for robot control. In *ICRA*, 2019.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. *ICLR*, 2019.

Katoch, S., Chauhan, S. S., and Kumar, V. A review on genetic algorithm: past, present, and future. *Multim. Tools Appl.*, 2021.

Khadka, S. and Tumer, K. Evolution-guided policy gradient in reinforcement learning. In *NeurIPS*, 2018.

Kielak, K. P. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? 2019.

Lambora, A., Gupta, K., and Chopra, K. Genetic algorithm-a literature review. In *COMITCon*. IEEE, 2019.

Larrañaga, P. and Lozano, J. A. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer Science & Business Media, 2001.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020.

Li, P., Tang, H., Yang, T., Hao, X., Sang, T., Zheng, Y., Hao, J., Taylor, M. E., Tao, W., and Wang, Z. PMIC: improving multi-agent reinforcement learning with progressive mutual information collaboration. In *ICML*, 2022.

Li, P., Hao, J., Tang, H., Zheng, Y., and Fu, X. Race: Improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution. In *ICML*, 2023.

Li, P., Hao, J., Tang, H., Fu, X., Zheng, Y., and Tang, K. Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey. *arXiv preprint*, 2024.

Li, X., Xu, X., and Zuo, L. Reinforcement learning based overtaking decision-making for highway autonomous driving. In *ICICIP*, 2015.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Liu, J., Ma, Y., Hao, J., Hu, Y., Zheng, Y., Lv, T., and Fan, C. A trajectory perspective on the role of data sampling techniques in offline reinforcement learning. In *AAMAS*, 2024a.

Liu, J., Wang, Z., Zheng, Y., Hao, J., Bai, C., Ye, Y., Wang, Z., Piao, H., and Sun, Y. Ovd-explorer: Optimism should not be the sole pursuit of exploration in noisy environments. In *AAAI*, 2024b.

Liu, X., Xue, Z., Pang, J., Jiang, S., Xu, F., and Yu, Y. Regret minimization experience replay in off-policy reinforcement learning. In *NeurIPS*, 2021.

Marchesini, E., Corsi, D., and Farinelli, A. Genetic soft updates for policy evolution in deep reinforcement learning. In *ICLR*, 2020.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *arXiv preprint*, 2013.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P., and Courville, A. C. The primacy bias in deep reinforcement learning. In *ICML*, 2022.

Pourchot, A. and Sigaud, O. CEM-RL: combining evolutionary and gradient-based methods for policy search. In *ICLR*, 2019.

Puterman, M. L. Markov decision processes. *HORMS*, 1990.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint*, 2017.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *ICLR*, 2016.

Schwarzer, A., Anand, A., Goel, R., Hjelm, R. D., Courville, A. C., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021.

Sedgwick, P. Spearman's rank correlation coefficient. *Bmj*, 2014.

Sigaud, O. Combining evolution and deep reinforcement learning for policy search: a survey. *arXiv preprint arXiv:2203.14009*, 2022.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint*, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning - an introduction*. MIT Press, 1998.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

Vikhar, P. A. Evolutionary algorithms: A critical review and its future prospects. In *ICGTSPICC*. IEEE, 2016.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 2019.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.

Xue, K., Wang, R., Li, P., Li, D., Jianye, H., and Qian, C. Sample-efficient quality-diversity by cooperative coevolution. In *ICLR*, 2023.

Young, K. and Tian, T. Minatar: An atari-inspired testbed for more efficient reinforcement learning experiments. *CoRR*, 2019.

Zhang, H., Shao, J., Jiang, Y., He, S., Zhang, G., and Ji, X. State deviation correction for offline reinforcement learning. In *AAAI*, 2022.

Zhou, M., Luo, J., Villella, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., et al. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving, 2020. *arXiv preprint*, 2010.

Zhou, Z., Yu, Y., and Qian, C. *Evolutionary learning: Advances in theories and algorithms*. 2019.

Zhu, Q., Wu, X., Lin, Q., Ma, L., Li, J., Ming, Z., and Chen, J. A survey on evolutionary reinforcement learning algorithms. *Neurocomputing*, 2023.

A. Method Implementation Details

All experiments are carried out on NVIDIA GTX 2080 Ti GPU with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz.

A.1. Network Architecture

This section provides a detailed description of the network architectures used in the experiments. For the GA and CEM processes, we utilize the implementations from ERL[1] and CEM-RL[2]. All the processes remain the same, and readers can refer to the source code for specific hyperparameter settings. Regarding the DQN architecture, the Q network consists of one convolutional layer followed by two fully connected layers. The convolutional layer has 16 output channels, a 3×3 filter size, and a stride of 1. The fully connected layers have 1024 and 128 units, respectively. The activation function used between layers is ReLU. Additional details can be found in Table 3. For Rainbow, we follow a popular Rainbow open source project[3] and use the default settings. Readers can refer to the specific source code for more in-depth information. As for SPR, our implementation is based on the official code of SPR[4]. We combine the GA-version VEB-RL with SPR and enhance the population using GA every 1000 steps. In VEB-SPR, only the best individual interacts with the environment.

Table 3. Details of setting.

Parameter	Value
Optimizer	Adam
Learning rate	$3e-4$
Replay buffer size	$1e5$
Number of hidden layers for Q network	2
Number of hidden units per layer	1024, 128
Batch size	32
Number of the convolutional layer	1
Out channels of the convolutional layer	16
Kernel size of the convolutional layer	3×3
The stride of the convolutional layer	1
Discounted factor γ	0.99
Steps to update the target network	1000
Sample size for calculating the fitness N	5120 in MinAtar & 1024 in Atari
Population size	10
Update frequency of target network in the population H	20

A.2. Hyperparameters

This section details the hyperparameter which is different across tasks. Only one hyperparameter N needs to tune across tasks. The specific values of N for each task are listed in Table 4.

A.3. Further Experiments

Analysis of Hyperparameter H We conduct an analysis to examine the effect of the hyperparameter H on performance, which determines the update frequency of target networks across all tasks. The results depicted in Figure 8 demonstrate that different values of H yield similar performance. Notably, when H is set to 0 and the target network is not utilized, a significant degradation in performance is observed. This finding aligns with the original DQN paper, which has consistently shown that omitting target networks in DQN leads to unstable learning.

Time and Resource Consumption In terms of time consumption, VEB-RL optimizes the population once per generation, with each generation consisting of at least 2 episodes (only one elite interaction with the environment). The main time consumption comes from population evaluation. To investigate the time consumption, we conduct experiments comparing

[1]https://github.com/ShawK91/Evolutionary-Reinforcement-Learning
[2]https://github.com/apourchot/CEM-RL
[3]https://github.com/Kaixhin/Rainbow
[4]https://github.com/mila-iqia/spr

Table 4. Details of the hyperparameter N across tasks.

Env name	N (GA version)	N (CEM version)
MinAtar *Freeway*	1	1
MinAtar *Space Invaders*	1	1
MinAtar *Breakout*	2	1
MinAtar *Seaquest*	1	1
MinAtar *Asterix*	2	1
Atari *Breakout*	5	1
Atari *Space Invaders*	2	1
Atari *Pong*	2	3
Atari *Battle Zone*	2	5
Atari *Name This Game*	1	1
Atari *Qbert*	2	1

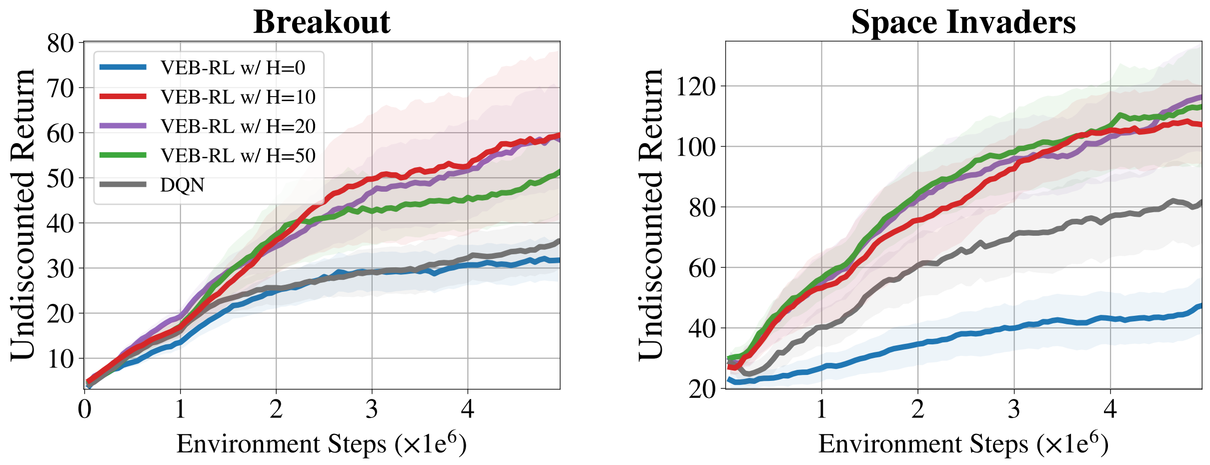


Figure 8. Hyperparameter analysis on H. The experimental results indicate that VEB-RL is not sensitive to the size of the hyperparameter H, but when $H = 0$ (indicating the absence of target networks), it leads to a significant performance decrease. This is consistent with the conclusion in the RL literature that removing target networks results in a decrease in performance.

CEM-based VEB-RL (one elite interaction) and DQN on *Breakout* for 500 million steps (over five seeds). The results show that VEB-RL takes 30.5 hours, while DQN takes 18.2 hours. The majority of the time overhead in VEB-RL arises from population evaluation with the new fitness metric, resulting in an additional 67% time consumption compared to DQN. Increasing the number of elites interacting with the environment can significantly reduce time consumption. For example, VEB-RL (5 elites interaction) only takes 22.5 hours. However, this may lead to performance degradation.

In terms of computational and resource consumption, VEB-RL requires maintaining a population, which incurs overhead in terms of population maintenance and optimization. The number of parameters in VEB-RL is linearly related to the size of the population. We compare the memory overhead of VEB-RL and DQN when loaded onto the GPU for training. VEB-RL incurs a memory overhead of 1221 MiB, while DQN incurs a memory overhead of 1051 MiB. Using VEB-RL does not result in excessive graphics memory overhead. Furthermore, since VEB-RL and DQN share the replay buffer, no additional memory overhead is required. The additional memory and computing overhead of VEB is small due to two factors: 1) the frequency of population evaluation and optimization in VEB is low, occurring once per generation (i.e., on average every 1000 steps), while RL is updated once per time step, and 2) EA in VEB-RL uses random search, eliminating the need for gradient optimization (gradient backpropagation and intermediate variable storage).

Comparison with A Pure RL Algorithm with Multiple Policies Readers might wonder how VEB-RL compares to a population-based pure RL algorithm. This curiosity stems from the idea that the improvements in VEB-RL are due to the introduction of the population rather than the various mechanisms within VEB-RL. To this end, we implement the DQN

	Breakout	Asterix	Space Invaders
CEM-Based VEB-RL	63.1 ± 16.5	27.3 ± 7.8	119.1 ± 25.5
DQN (multiple)	24.4 ± 4.2	4.0 ± 1.2	55.3 ± 12.8

Table 5. Average Performance comparison of CEM-based VEB-RL and DQN (multiple) on MinAtar tasks.

w/ RL Interaction	VEB-RL(N=1)	VEB-RL(N=2)	VEB-RL(N=3)	VEB-RL(N=5)
Breakout	63.4	48.3	43.5	46.9
w/o RL Interaction	VEB-RL(N=1)	VEB-RL(N=2)	VEB-RL(N=3)	VEB-RL(N=5)
Breakout	43.7	36.6	33.0	35.9
w/ RL Interaction	VEB-RL(N=1)	VEB-RL(N=2)	VEB-RL(N=3)	VEB-RL(N=5)
Asterix	29.0	30.4	13.0	5.2
w/o RL Interaction	VEB-RL(N=1)	VEB-RL(N=2)	VEB-RL(N=3)	VEB-RL(N=5)
Asterix	4.9	7.7	4.4	5.5

Table 6. Average Performance comparison on *Breakout* and *Asterix* tasks.

based on the idea of multiple RL policies with a shared replay buffer, maintaining the same number of DQNs as VEB-RL. We compared VEB-RL with DQN (multiple) on three tasks: *Breakout*, *Asterix*, and *Space Invaders*. All structures and hyperparameters are kept consistent, and the experiments are repeated 5 times. The results in Table 5 demonstrate that VEB-RL outperforms DQN (multiple) in terms of the final performance. Moreover, DQN (multiple) requires gradient optimization for each individual in the population, resulting in significant resource overhead. In contrast, VEB-RL is lightweight and more efficient.

The main reason DQN (multiple) performs worse than VEB-RL is that multiple Q networks are improved based on the same experiences through gradient optimization, which tends to get similar solutions and lacks diversity. This leads to the collection of redundant suboptimal experiences. Consequently, DQN (multiple) is prone to getting stuck in suboptimal solutions. In contrast, VEB-RL utilizes EAs to optimize the Value Function with gradient-free optimization, providing stronger global optimization and exploration capabilities. It is more adept at escaping local suboptimal points while ensuring diversity. Additionally, VEB-RL maintains an elite interaction mechanism, preventing low-quality individuals from interacting with the environment.

VEB-RL without RL Interaction In our study, we find that smaller values of N typically result in better performance. An intuitive idea to potentially improve performance is to stop RL interaction and restrict environmental interactions to only elite individuals from the population. However, this approach would result in performance collapse rather than improvement, primarily due to distributional shift. To support the statement, we evaluate VEB w/o RL interaction on *Breakout* and *Asterix* tasks, with $N = 1, 2, 3, \text{and} 5$. The results in Table 6 We observe that VEB-RL (w/ RL interaction) generally outperforms VEB-RL (w/o RL interaction). This is primarily due to the following reasons: Maintaining RL interaction allows for the stable preservation of the data distribution corresponding to the RL policy in the replay buffer, while also ensuring stable exploration capabilities of RL. Excluding RL interaction can lead to compromised exploration capabilities of RL (which can be mitigated by population injection but still exists), Besides, Out Of Distribution (OOD) issues may arise. For example, in the case of Asterix, we can see that VEB-RL w/o RL interaction experiences a direct collapse in performance. This is mainly because of the significant disparity between the true data distribution of RL and the data distribution collected in the replay buffer, resulting in an exponential increase in approximation errors (as shown in Table 7). The excessively large errors hinder RL from participating in the population evolution, leading to a vicious cycle and performance collapse.

Other Selection of Maintaining Q Target Network In our implementation, we initially considered maintaining a single target network for the entire population but ultimately decided to maintain a separate target network for each individual in the population. Through experiments, we verify that similar performance can be obtained for both designs. We conduct experiments using a single target network that is hard updated by the best-performing individual selected by fitness every H

w/o RL Interaction	VEB-RL(N=1)	VEB-RL(N=2)	VEB-RL(N=3)	VEB-RL(N=5)
Asterix	4.49 (10^{12})	6.39 (10^{13})	1.03 (10^{14})	4.86 (10^{13})

Table 7. Value approximation errors of VEB-RL w/o RL Interaction on *Asterix* task.

Table 8. Experiments on different designs of target networks.

Env name	Breakout	Space Invaders	Asterix
Use one target	68.3 ± 33.8	130.6 ± 23.4	28.7 ± 10.0
Use multiple targets	63.1 ± 16.5	119.1 ± 25.5	27.3 ± 7.8

generations. We assess the performance using CEM-based VEB-RL in three tasks: *Breakout*, *Space Invaders*, and *Asterix*. The results in Table 8 demonstrate that a similar level of performance can be achieved using a single target network, with some improvement and less resource consumption.