BiERL: A Meta Evolutionary Reinforcement Learning Framework via Bilevel Optimization

Junyi Wang[*1]**, Yuanyang Zhu**[*1]**, Zhi Wang**[†1]**, Yan Zheng**[2]**, Jianye Hao**[2] **and Chunlin Chen**[1]

[1]Department of Control Science and Intelligent Engineering, Nanjing University
[2]College of Intelligence and Computing, Tianjin University

Abstract. Evolutionary reinforcement learning (ERL) algorithms recently raise attention in tackling complex reinforcement learning (RL) problems due to high parallelism, while they are prone to insufficient exploration or model collapse without carefully tuning hyperparameters (aka meta-parameters). In the paper, we propose a general meta ERL framework via bilevel optimization (BiERL) to jointly update hyperparameters in parallel to training the ERL model within a single agent, which relieves the need for prior domain knowledge or costly optimization procedure before model deployment. We design an elegant meta-level architecture that embeds the inner-level's evolving experience into an informative population representation and introduce a simple and feasible evaluation of the meta-level fitness function to facilitate learning efficiency. We perform extensive experiments in MuJoCo and Box2D tasks to verify that as a general framework, BiERL outperforms various baselines and consistently improves the learning performance for a diversity of ERL algorithms.

1 Introduction

Reinforcement learning (RL) is achieving tremendous breakthroughs in various fields, such as games [37], robotics [40], and even matrix multiplication in fundamental mathematics [10]. An alternative approach to solving high-dimensional RL problems is using black-box optimization, yielding an explosive growth of evolutionary reinforcement learning (ERL) algorithms such as evolution-guided policy gradients [21], CERL [20], MERL [26], ERL-Re^2 [16], etc. In this paper, we focus on a particular set of optimization algorithms in this class, evolution strategies (ES) [30], which has been reported to be competitive to popular backpropagation-based algorithms such as policy gradient and deep Q-learning due to better parallelization and strong exploration ability [38].

Typically, ES computes the zero-order search gradient by aggregating a population of individuals that come from perturbing the network parameters with Gaussian noise. Unlike policy gradient methods that directly differentiate the learning objective, perturbing the parameter space has no explicit correlations to the policy outputs. Hence, current ES algorithms are extremely sensitive to hyperparameters (aka meta-parameters), e.g., the noise covariance and learning rate, especially the noise level that controls the magnitude of exploration. It could easily lead to insufficient exploration or model collapse without carefully tuning these hyperparameters. Well-known procedures, such as random search and sequential optimization [19], require fully training enormous models to identify appropriate hyperparameters. While effective for simple tasks, they lack adaptation

property in weight updates, which is validated to be effective from commonly used adaptive optimizers, such as Adam [22]. Research also shows that the performance could still be brittle using static hyperparameters in complex and dynamic domains [7]. To leverage adaptation of hyperparameters, a straightforward idea is to train multiple agents with a large array of hyperparameters in parallel as in population-based training (PBT) [19]. However, it could be too costly in practice due to maintaining enormous agents simultaneously within a single learning process [35]. Generally, existing works either require prior domain knowledge or expensive computation for reliable hyperparameter optimization.

To address this issue, we propose a meta ERL framework via bilevel optimization (BiERL) to adaptively update meta-parameters parallelly to training the ERL model within a single agent, relieving the need for explicit domain knowledge in advance or costly optimization procedure before model deployment. The meta-level adaptively adjusts hyperparameters of the ES model according to the inner-level's population representation over the agent's experience, while the inner-level trains the ES model via the hyperparameters given by the meta-level. We design an informative population representation encoder that considers the history of how the inner level evolves over a certain time horizon. The ES algorithm also trains the meta-level network to maximize the final returns achieved after inner-level learning. To facilitate learning efficiency, we introduce a simple and feasible evaluation of the meta-level fitness function using a truncated estimate of complete inner-level optimization. Also, we develop a nonparametric implementation based on Bayesian optimization (BO) to learn adaptive hyperparameters, promoting BiERL as a more general framework. Usually, the nonparametric meta-level design with BO is easier to be implemented with a simpler procedure, while the parametric one with a neural network can facilitate more effective joint adaptation on multiple hyperparameters [15].

In summary, our main contributions can be enumerated as follows:

1. We propose a novel and general meta ERL framework that produces efficient meta-parameter optimization without the requirement for prior domain knowledge or costly optimization procedure before model deployment.
2. We introduce an elegant meta-level architecture that embeds the inner-level's evolving experience into an informative population representation, and we design a simple and feasible evaluation of the meta-level fitness function to facilitate learning efficiency.
3. We perform extensive experiments to verify that as a general framework, BiERL outperforms various baselines and consistently improves learning performance across a diverse range of ERL algorithms.

---

* Equal contributions.
† Correspondence to: Zhi Wang (zhiwang@nju.edu.cn).

Please check ArXiv or contact the authors for any appendices or supplementary material mentioned in the paper.

2 Related Work

Evolution strategies. ES algorithms Evolution Strategies (ES) algorithms have emerged as a scalable alternative for RL tasks, owing to their high parallelism and strong exploration capabilities [30, 21]. However, previous works have observed that ES algorithms are exclusively on-policy, extracting only a limited amount of information from samples. Consequently, in comparison to backpropagation-based algorithms, ES algorithms often necessitate more rollouts and are susceptible to local optima [7]. Previous efforts to enhance the performance of ES algorithms have included adding regularization terms to encourage exploration [8] or employing various noise perturbations [1]. While ES algorithms are generally less sensitive to hyperparameter settings than traditional DRL [30], prior research has shown that hyperparameter fine-tuning can still be effective for them. Although some feasible ideas on hyperparameters have been proposed in these works [28, 34], they are predominantly empirical and primarily focus on a single hyperparameter. In contrast to these previous efforts, our method can embed the agent's evolving experience to dynamically change multiple hyperparameters.

Hyperparameter optimization. Hyperparameter optimization methods for RL are mainly sequential optimization [19], such as Bayesian optimization. However, these methods need multiple training runs to find optimal hyperparameters. Some other hyperparameter optimization methods, e.g., evolutionary algorithms, are categorized as parallel search [24]. These methods have high parallelism but need more computational resources. Both of these methods find static hyperparameters before training. Nevertheless, in complex tasks, static hyperparameters may lead to a decline in performance.

To overcome this limitation, PBT [19], which combines sequential optimization with evolutionary algorithms, has shown promising potential in online hyperparameter optimization due to its generality and robustness. Inspired by this concept, several successful works have been proposed, such as ALFA [4], OHT-ES [35], and Hyper-Distill [23]. In these methods, multiple agents with a large array of hyperparameters are updated in parallel to the model training. However, it could be too costly in practice due to maintaining enormous agents simultaneously. Additionally, while the online hyperparameter optimization of ES algorithms has already shown its effectiveness in previous works [29], most works on ES algorithms still rely on fine-tuning to find static hyperparameters before model training, which is exactly the main novel contribution of our methods.

Meta-ES. To improve performance, some works attempt to combine ES algorithms with meta-learning. These hybrid methods are named meta-ES. Existing meta-ES methods can be divided into two categories, multi-task meta-ES, and single-task meta-ES.

Most meta-ES methods belong to the multi-task meta-ES, which aims to train agents that can quickly adapt to similar but unseen tasks with ES algorithms. Representative works include ES-MAML [32], Evolvability ES [13], and Baldwinian Meta-Learning [11], which are all based on the MAML framework [12]. Other works, such as EPG [18] and EvoGrad [5], take advantage of the gradient-free features of ES algorithms to avoid calculating second-order derivatives in meta-learning. Nevertheless, these methods merely treat ES algorithms as an auxiliary technique and ignore the performance of the ES algorithm itself.

On the contrary, single-task meta-ES methods focus on the meta-parameters (aka hyperparameters) of ES algorithms. These works point out that the meta-parameters need to vary as learning progresses for a better performance [29]. To realize an adaptive meta-parameters, existing meta-ES methods include $[1, 2(1, \lambda)\gamma]$-meta-ES [17], OMPAC [9], and Meta-evolution [6]. However, these methods only apply simple techniques such as random search or heuristic function, which are only for solving low-dimensional tasks. Different from these works, we make it feasible for our method to dynamically change meta-parameters in more complex RL tasks.

Bilevel optimization. Bilevel optimization includes two sub-objectives called the meta/upper-level and inner/lower-level objectives, respectively. The meta-level objective must be solved subject to the optimality of the inner-level one and vice versa.

The mixture of RL algorithms and bilevel optimization framework has achieved success in many domains [31], such as neural architecture search [27], imitation learning [2], intrinsic rewards [33], and hyperparameter optimization [25]. Our method also utilizes a bilevel optimization framework for better generalizability in different ERL methods. Additionally, to compensate for additional computation costs from the conventional bilevel optimization framework, we propose a simple and feasible approximation of the meta-level evaluation and introduce a warm starting mechanism to replace random initialization in our method.

3 Method

3.1 Bilevel Framework

Our proposed BiERL is a meta ERL framework via bilevel optimization that jointly updates hyperparameters $\mathcal{H}_t$ at a certain timestep t in parallel to training the parameters of the ES model within a single agent. We aim to solve a bilevel optimization problem where our goal is to find hyperparameters $\mathcal{H}_t$ that maximize the meta-level fitness function F of the model parameterized by θ_t and trained with the inner-level fitness function f and $\mathcal{H}_t$:

$$
\begin{aligned}
\mathcal{H}_{t+1}^* &= \arg\max_{\mathcal{H}_t} F(\mathcal{H}_t, \theta_t^*(\mathcal{H}_t)), \\
\text{where } \theta_t^*(\mathcal{H}_t) &= \arg\max_{\theta_t} f(\mathcal{H}_t, \theta_t).
\end{aligned}
\tag{1}
$$

Figure 1 presents the overall structure of BiERL and the comparison to conventional ERL methods. Generally, conventional ERL methods train the model with static hyperparameters, e.g., the noise covariance and learning rate of ES. However, static hyperparameters could easily lead to inappropriate exploration or even model collapse without explicit domain knowledge in advance when dealing with complex tasks. On the contrary, our BiERL framework attempts to tackle such deficiencies by introducing adaptive hyperparameter optimization. The meta-level will adaptively adjust hyperparameters of ERL according to how the inner-level evolves over a certain time horizon. Then, the inner level trains the ERL model for several timesteps using the hyperparameters given by the meta-level. Taking the noise level of ES as an example, BiERL is more likely to bypass bad local optima by enlarging the noise covariance to encourage more exploration in the parameter space and also avoid model collapse by annealing the parameter noise.

3.2 Inner-level

In the inner level of BiERL, the agent trains the parameters θ_t via ES to maximize the fitness function $f(\theta_t(\mathcal{H}_t))$ based on the hyperparameters $\mathcal{H}_t$ from the meta-level.
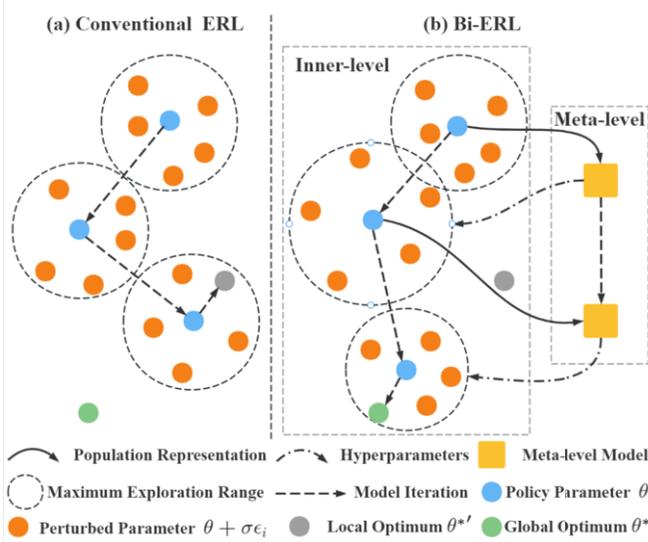
Figure 1: The overall structure of BiERL.

Evolution strategies. Inspired by natural evolution, ES algorithms are representative algorithms of "black-box optimization". ES algorithms seek to optimize the fitness function $f(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\sum_{i=0}^{\infty} \gamma^i r_i \right]$, which evaluates the parameters θ. Instead of directly calculating the gradient of the return like backpropagation-based algorithms, ES algorithms use an estimator of the "search gradient" based on a population of perturbed parameters.

In each iteration, perturbed parameters $\theta + \sigma \epsilon_i$, $i = 1, 2, ..., n$ are sampled and evaluated, where σ is the noise covariance, n is the size of the population, and $\epsilon_i \sim \mathcal{N}(0, \boldsymbol{I})$ is the Gaussian noise. Thus, the Monte Carlo estimate of the search gradient $\nabla_\theta f(\theta)$ becomes

$$
\begin{aligned}
\nabla_\theta f(\theta) &= \nabla_\theta \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \boldsymbol{I})} [f(\theta + \sigma \epsilon)] \\
&= \frac{1}{\sigma} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \boldsymbol{I})} [f(\theta + \sigma \epsilon) \epsilon] \\
&\approx \frac{1}{n\sigma} \sum_{i=1}^{n} f(\theta + \sigma \epsilon_i) \epsilon_i.
\end{aligned}
\tag{2}
$$

Then, the model parameters θ_t are updated as

$$
\theta_{t+1} \leftarrow \theta_t + \frac{\alpha}{n\sigma} \sum_{i=1}^{n} f(\theta_t + \sigma \epsilon_i) \epsilon_i,
\tag{3}
$$

where α is the learning rate.

As a general framework, BiERL is able to update hyperparameters for various ERL methods, e.g., Vanilla ES [30], NSR-ES [8], and ESAC [34]. Experimental results have verified that our BiERL framework can improve the learning performance for multiple ES algorithms. During the training of the inner-level network, the population representation at each timestep $\boldsymbol{f}_t$ will be stored in a small-scale replay buffer for the meta-level. The agent will update the meta-level model when the inner-level is trained with k iterations.

3.3 Meta-level

First, we introduce the network architecture of the meta-level design with the informative population representation encoder. Then, we present the training process of the meta-level, where a simple and

feasible evaluation of the fitness function is crucial for efficient learning. In addition, a warm starting mechanism is introduced to facilitate the training process. Finally, we give the nonparametric implementation of the meta-level to learn adaptive hyperparameters.

Network architecture. The meta-level aims to adjust the meta-parameters according to the inner-level's evolving situation. An elegant design of the meta-level architecture should at least satisfy three conditions: (1) maintaining a small scale to alleviate the costly optimization procedure; (2) capturing the history information of the evolution to gain better representation; (3) exhibiting good generalization across different tasks and diverse architectures of the inner-level.

Inspired by this, Figure 2 presents the proposed network structure of the meta-level model. We consider the information of all the individuals in the population over a certain time horizon. In each iteration of the inner-level, a population of n perturbed parameters $\theta_t + \sigma_t \epsilon_i$, $i = 1, ..., n$ is generated with the fitness function $\boldsymbol{f}_t = [f(\theta_t + \sigma_t \epsilon_1), ..., f(\theta_t + \sigma_t \epsilon_n)]$ calculated. Here, we take this kind of population representation over the past k timesteps $\mathbb{S}_t = [\boldsymbol{f}_{t-k+1}, ..., \boldsymbol{f}_t]$ as the input to the meta-level network. We use a replay buffer to store the inner-level's fitness function over the past k timesteps as $\mathbb{S}_t$, which is fed into an LSTM network ψ to encode the evolving population representation as $\mathbb{X}_t = \psi_t(\mathbb{S}_t)$. Then, an MLP ϕ takes as input the population representation that indicates the history evolving situation and outputs the adaptive hyperparameters for the inner-level ES model, e.g., the noise covariance σ or learning rate α, as $\mathcal{H}_{t+1} = \phi_t(\mathbb{X}_t)$.

Using the above design, the dimension of the meta-level's input can remain at a small scale, as opposed to using the entire network weights that may reach millions of dimensions [4]. This is significant for efficiently deploying adaptive hyperparameter optimization on real-world applications. Then, the evolving situation of the inner-level can be precisely embedded by the population representation encoder that leverages the advances of LSTM. Finally, using the fitness function as the population representation is agnostic to the inner-level's network design and the task at hand. Existing works usually use the entire weights of the inner-level as the input [4], leading to different meta-network designs for different tasks due to different dimensions of the state-action spaces. In contrast, our meta-network can keep the same architecture for different tasks by using the task-agnostic fitness function as the population representation. Hence, the meta-level model is agnostic to the design of the inner-level's network design (other than the algorithm generalization across tasks), which could enjoy better method universality.

Model training. The meta-level network is also trained using the ES algorithm to maximize the final returns achieved after inner-level learning. Let $\vartheta = [\psi, \phi]$ denote the parameters of the meta-level model. Evaluating the meta-level fitness function $F(\mathcal{H}_{\vartheta_t}, \theta_t^*(\mathcal{H}_{\vartheta_t}))$ is essential for the model training. The evaluation is based on the final return after the inner-level ES model has been completely optimized with the hyperparameters $\mathcal{H}_{\vartheta_t}$ given by the meta-level. It is obvious that such a process could largely decrease learning efficiency.

To design a simple and feasible evaluation of $F(\mathcal{H}_{\vartheta_t}, \theta_t^*(\mathcal{H}_{\vartheta_t}))$, we use the return after one-step optimization as a truncated estimate of the return after complete inner-level optimization. Specifically, the meta-level will first sample a group of m perturbed parametric models $\vartheta_t^j = \vartheta_t + \omega \varepsilon_j$, $\varepsilon_j \sim \mathcal{N}(0, \boldsymbol{I})$, $j = 1, 2, ..., m$, where ω is the meta-level noise covariance. Each perturbed model ϑ_t^j will output a set of hyperparameter $\mathcal{H}_{\vartheta_t^j}$ based on the inner-level's population rep-
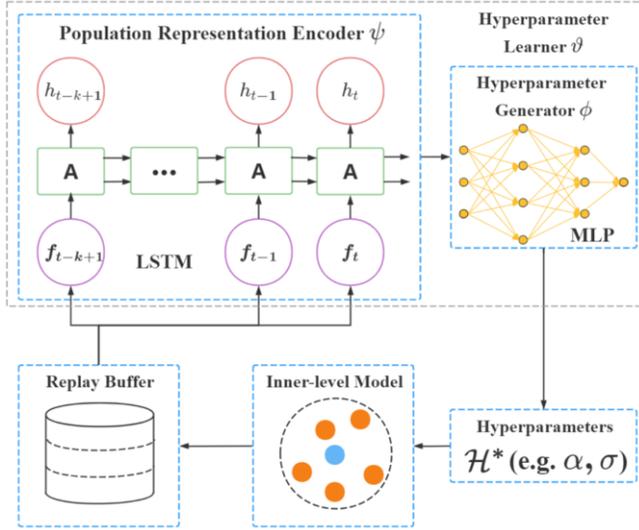
Figure 2: The structure of the Meta-level model.

Algorithm 1: Training process of the meta-level

Input: Meta-level population size m, meta-level learning rate β, meta-level noise covariance ω, θ_t, $\mathbb{S}_t$, $\vartheta_t = [\phi_t, \psi_t]$

Output: New meta-level model ϑ_{t+1}

1 Sample $\varepsilon_1, ..., \varepsilon_m \sim \mathcal{N}(0, \boldsymbol{I})$
2 **for** $j = 1, 2, ..., m$ **do**
3 　　$\vartheta_t^j = \vartheta_t + \omega\varepsilon_j$　$\mathbb{X}_t^j = \psi_t^j(\mathbb{S}_t)$
4 　　$\mathcal{H}_{\vartheta_t^j} = \phi_t^j(\mathbb{X}_t^j)$
5 　　Estimate $F(\mathcal{H}_{\vartheta_t^j}, \theta_t^*(\mathcal{H}_{\vartheta_t^j}))$ by Eq. (5)
6 **end**
7 Update ϑ_t by Eq. (6)

---

resentation. Then, the inner-level model $\theta_t(\mathcal{H}_{\vartheta_t^j})$ will be updated to $\hat{\theta}_t(\mathcal{H}_{\vartheta_t^j})$ with only one-step optimization as

$$\hat{\theta}_t^j \leftarrow \theta_t^j + \frac{\alpha_t^j}{n\sigma_t^j} \sum_{i=1}^{n} f(\theta_t^j + \sigma_t^j \epsilon_i)\epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \boldsymbol{I}), \quad (4)$$

where $\hat{\theta}_t^j$ and θ_t^j denote $\hat{\theta}_t(\mathcal{H}_{\vartheta_t^j})$ and $\theta_t(\mathcal{H}_{\vartheta_t^j})$, respectively, and α_t^j and σ_t^j are adaptive hyperparameters generated by $\mathcal{H}_{\vartheta_t^j}$. We repeat this process for l times to obtain a Monte Carlo estimate of the final meta-level fitness function as

$$F\left(\mathcal{H}_{\vartheta_t^j}, \theta_t^*(\mathcal{H}_{\vartheta_t^j})\right) \approx F\left(\mathcal{H}_{\vartheta_t^j}, \hat{\theta}_t(\mathcal{H}_{\vartheta_t^j})\right)$$
$$= \mathbb{E}\left[f\left(\hat{\theta}_t(\mathcal{H}_{\vartheta_t^j})\right)\right] \quad (5)$$
$$\approx \frac{1}{l} \sum_{i=1}^{l} [f\left(\hat{\theta}_t^{j,i}\right)].$$

The search gradient estimation of ϑ is computed as

$$\vartheta_{t+1} \leftarrow \vartheta_t + \frac{\beta}{m\omega} \sum_{j=1}^{m} F\left(\mathcal{H}_{\vartheta_t^j}, \theta_t^*(\mathcal{H}_{\vartheta_t^j})\right) \varepsilon_j, \quad (6)$$

where β is the learning rate of the meta-level model. Algorithm 1 summarizes the training process.
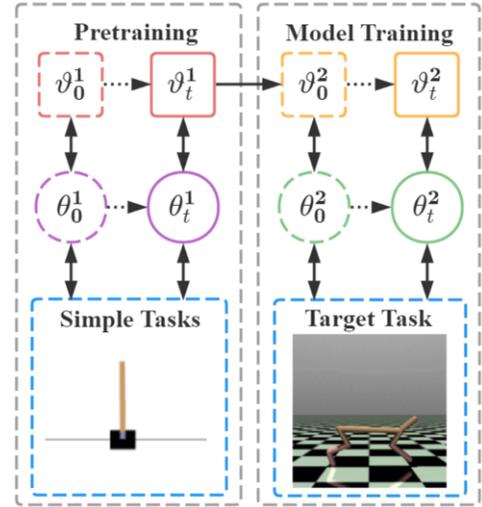

Figure 3: Warm starting mechanism.

Warm starting. The training of the meta-level model is complicated since it involves an ES update of the inner-level for each individual in the meta-level population. Inspired by [3], we introduce a warm staring mechanism to improve the training process. Instead of random initialization, we pretrain the meta-level model in simple tasks and use it as the initialization of the model in the target task, as shown in Figure 3. Conventional warm-starting mechanisms may require tasks to be relatively similar. However, as mentioned above, the design of the population representation encoder is agnostic to the task at hand, so it is more feasible for the warm starting mechanism to transfer some knowledge from simple tasks to help train the difficult target task. In practice, we only pretrain the meta-level once in a simple task and save the meta-level for any future initialization.

Nonparametric design. To promote BiERL as a more general framework, we also develop a nonparametric design of the meta-level based on BO to learn adaptive hyperparameters. Different from the parametric design that requires maintaining a network, the nonparametric model only constructs a dynamic meta-level fitness function $F(\mathcal{H}_t, \theta_t^*(\mathcal{H}_t))$ and find the optimal solution with BO periodically. This is equivalent to dividing the entire training process into iteratively alternating between training the inner-level ES model and optimizing the inner-level's hyperparameters through BO. Generally, the nonparametric design with BO is easier to be implemented with a simpler architecture, while the parametric one with the neural network architecture can facilitate more effective joint adaptation on multiple hyperparameters. Analogous to the estimation approach introduced in the parametric model, the meta-level fitness function $F(\mathcal{H}_t, \theta_t^*(\mathcal{H}_t))$ is also constructed after one-step optimization as

$$F(\mathcal{H}_t, \theta_t^*(\mathcal{H}_t)) \approx \frac{1}{l} \sum_{i=1}^{l} \left[f(\hat{\theta}_t^i)\right]. \quad (7)$$

3.4 Integrated Algorithm

Based on the above implementations, Algorithm 2 presents the integrated BiERL framework that jointly updates hyperparameters in parallel to training the ES model within a single agent. In the meta-level, we can apply the parametric network with ES update or the nonparametric BO procedure to produce adaptive hyperparameters.

Algorithm 2: BiERL

Input: Population size n, interval k
Output: Optimal inner-level policy parameters θ^*

1 Initialize θ_0, $\mathcal{H}_0$, Replay Buffer $\mathcal{R}$
2 Warm starting $\vartheta_0 = [\phi_0, \psi_0]$
3 **for** $t = 0, 1, 2, ...$ **do**
4   **if** *Use parametric model* **then**   // Meta-level
5   Update ϑ_t by Algorithm 1
6   **else if** *Use nonparametric model* **then**
7   Construct $F(\mathcal{H}_t, \theta_t^*(\mathcal{H}_t))$ by Eq. (7)
8   Use BO to find $\mathcal{H}_{t+1}^*$
9   **end**
10   **for** $i = 0, 1, 2, ..., k - 1$ **do**   // Inner-level
11   Use ERL to update $\theta_{t+1,i}$ with $\mathcal{H}_{t+1,i}$
12   **if** *Use parametric model* **then**
13   Store $\boldsymbol{f}_{t+1,i}$ to $\mathcal{R}$
14              $\mathbb{S}_{t+1,i} = [\boldsymbol{f}_{t,i+1}, ..., \boldsymbol{f}_{t+1,i}]$
15              $\mathbb{X}_{t+1,i} = \psi_{t+1}(\mathbb{S}_{t+1,i})$
16              $\mathcal{H}_{t+1,i+1} = \phi_{t+1}(\mathbb{X}_{t+1,i})$
17   **end**
18   **end**
19 **end**

These two designs are suitable for different situations with respective advantages, which will be furtherly demonstrated in the experiments.

4 Experiments

We conduct extensive experiments to verify the effectiveness of BiERL on a benchmark of continuous control problems in OpenAI Gym [14, 39]. Specifically, we seek to answer the following research questions (RQs). The source code is available at https://github.com/chriswang98sz/BiERL.

- **RQ1 (Performance):** Can BiERL improve the performance of basic ERL methods across different RL tasks?
- **RQ2 (Scalability):** Can BiERL framework effectively adapt to other hyperparameter optimization?
- **RQ3 (Ablation):** What are the respective contributions of different modules to overall performance?

4.1 Performance (RQ1)

To demonstrate and illustrate the outperformance of our method, we compare BiERL with three representative ERL methods including Vanilla ES [30], NSR-ES [8], and ESAC [34] on a total of 6 MuJoCo [36] and 2 Box2D tasks in the OpenAI Gym [14]. We apply our BiERL architecture to the three baselines as the meta-level optimizer with the parametric model (PM) and nonparametric model (NPM) for optimizing the noise covariance σ, respectively. Each algorithm runs with five random seeds and is reported in mean $\pm$ standard deviation. Further implementation details are given in Appendix.

The comparisons across BiERL, Vanilla ES, NSR-ES, and ESAC baselines are shown in Figure 4. We make a conclusion from the results: BiERL > NSR-ES > Vanilla ES > ESAC[1]. Specifically, (1) applying the PM and NPM to three baselines can significantly improve the performance, and lead to high learning efficiency, which

---

[1] Here, we use the binary comparison operators to indicate the performance order of these algorithms.

validates the effectiveness of BiERL. It implies that adaptive hyperparameters for changing the noise covariance σ can help the agent learn more efficiently; (2) NSR-ES obtains better performance than Vanilla ES and ESAC by plugging the PM or NPM of BiERL. The reason may be that NSR-ES is able to avoid local optima encountered by combining exploration with reward maximization; (3) Vanilla ES achieves comparable performance with the ESAC across all the tasks, which implies that there are potentials for rapid improvements since Vanilla ES reproduces one policy in each iteration and may update the same policy for several iterations; (4) plugging PM into baselines achieves more gains over NPM most of the time, which validates that PM could capture long-term dependencies from the population representation using the advance of LSTM; (5) BiERL can achieve more statistically significant advantages in complex tasks (e.g., Ant-v2 and HalfCheetah-v2) than the simple ones (e.g., Walker2d-v2 and LunarLanderContinuoous-v2) across all baselines, which shows the superiority of BiERL in challenging tasks. It should benefit from the meta-level generating adaptive hyperparameters, leading to adequate exploration to avoid local optima and discover a better policy.

Since our focus is a general framework that realizes efficient meta-parameter optimization for general ERL algorithms, we mainly investigate the performance improvement of deploying BiERL on a diversity of typical ERL algorithms. In addition, we also present experimental results with typical state-of-the-art DRL methods, including TRPO, PPO, and SAC, as is shown in Figure 5. It can be observed that our method can outperform the state-of-the-art DRL methods in these tasks, especially at the beginning of the training process, which further validates the effectiveness of adapting the hyperparameters during learning and the efficiency of our method.

Next, Figure 6 shows the detailed running time regarding the learning iterations of BiERL, Vanilla ES, and DRL methods evaluated on the example Ant-v2 and LunarLanderContinuous-v2 tasks. It can be observed that the required running time of BiERL is roughly at the same scale as both baselines of ES and DRL methods. Owing to the simple and feasible approximation of the meta-level evaluation and the parallelizability of ERL methods, the general framework of BiERL does not significantly increase the time complexity of ERL algorithms, while it is capable of consistently improving the learning performance for various ERL algorithms. In addition to performance comparison, high parallelization is another promising advantage of ERL methods, including BiERL. We can use distributed computing environments to further reduce the computation time, and obtain greater superiority compared to DRL methods.

In summary, BiERL can consistently improve the learning performance of various ERL methods with little increase in the computation time. More experiments conducted on other MuJoCo and Box2D tasks in OpenAI Gym are included in the Appendix.

4.2 Scalability (RQ2)

To answer RQ2, we assess the scalability of BiERL. We extend the learning rate α as joint optimization hyperparameters by the meta-level to study whether BiERL can consistently produce high performance with a better mechanism of adaptive learning rate. To ensure the positivity of the learning rate, we clip the parameterization $\alpha \in [0.016, 0.024]$ and denote it as online $\sigma \& \alpha$.

Figure 7 presents the scalable nature of BiERL. From Figure 7-(a) and -(b), by comparison with fixed learning rate α, we find that the online $\sigma \& \alpha$ method obtains improved performance for both tasks. It implies that our BiERL can provide prominent performance boosts and scalability. Further, we study the variation of the learning rate
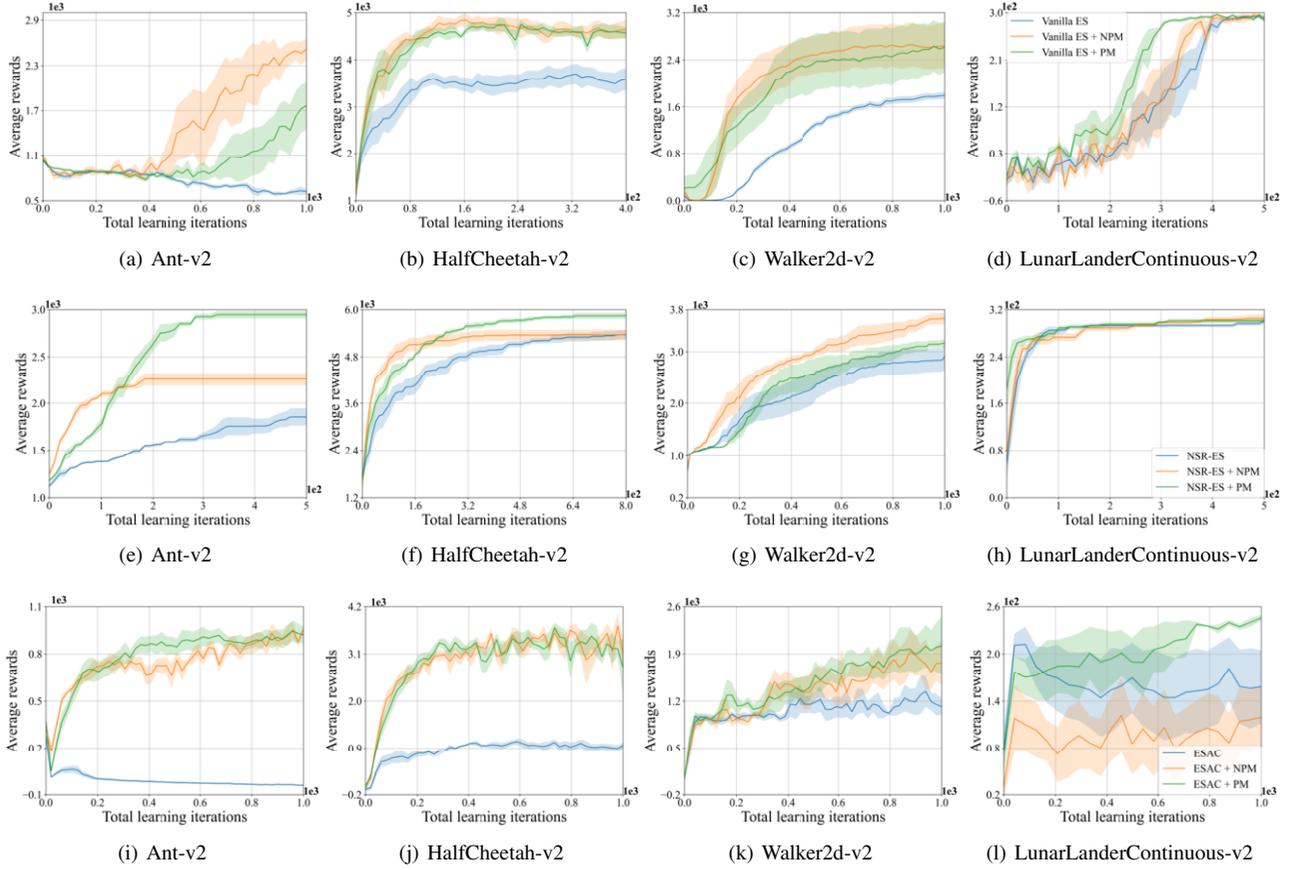
Figure 4: Performance of different algorithms in three standard MuJoCo and one Box2D environments. Figures (a)-(d), (e)-(h), and (i)-(l) show the learning curves of Vanilla ES, NSR-ES, and ESAC by plugging our PM and NPM methods, respectively. All the curves are averaged over 5 different random seeds, and the standard deviation is shown as a shaded region.
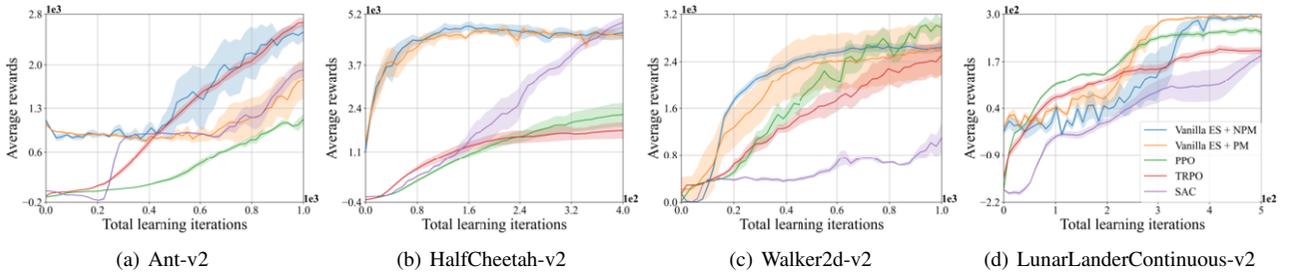
Figure 5: Performance of our method and state-of-the-art DRL methods in three standard MuJoCo and one Box2D environments. All the curves are averaged over 5 different random seeds, and the standard deviation is shown as a shaded region.

and noise covariance during the learning process. As shown in Figure 7-(c) and 7-(d), noise covariance σ becomes small as the policy tends to converge to the robust policy during the learning process. It is required that a smaller noise covariance σ can produce more stable behaviors, which can reduce catastrophic outcomes with less randomized behaviors. It can also be observed that the learning rate α is generated in a dynamic range for each inner-loop step. An interesting behavior to note is that the ranges of generated hyperparameter values are small. We believe that such a phenomenon could be owed that the learning rate α allows the model to focus on learning stability.

Moreover, we study the influence of another two hyperparameters, the population size of the inner-level n and the meta-level m. These

experimental results are provided in Figure 8. We find that a larger population size tends to achieve high performance. However, when the population size is too large, some policies with similar behavior will be selected and updated, harming the efficiency of BiERL. Thus, an appropriate population size (e.g., 150 or 200) can better trade off between performance and efficiency.

4.3 Ablation study (RQ3)

To investigate the role of each part of the parametric meta-level, we perform a series of ablation studies including (1) **PM without population representation** represents that the input of the meta-level solely considers the unperturbed parameters $f(\theta_t)$ instead of popu-
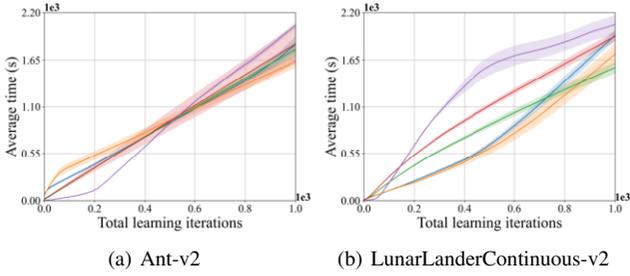
Figure 6: Running time regarding the learning iterations of BiERL, Vanilla ES, and DRL methods evaluated on the two example tasks.

(a) HalfCheetah (Performance) (b) Walker2d (Performance)

(c) HalfCheetah ($\sigma\&\alpha$) (d) Walker2d ($\sigma\&\alpha$)
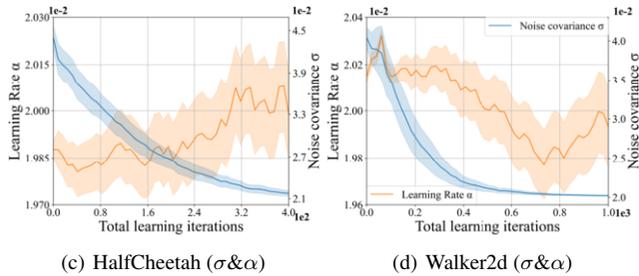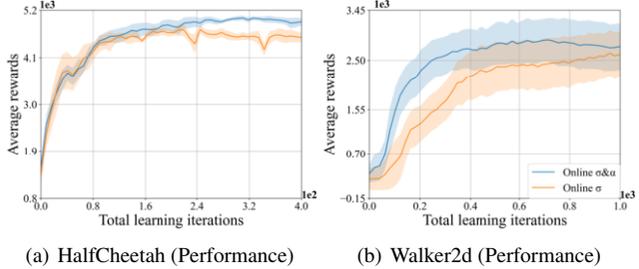
Figure 7: Scalability analysis on the learning rate α for BiERL. (a)-(b) show learning curves, and (c)-(d) show the numerical values of hyperparameters during learning.

(a) HalfCheetah (n) (b) Walker2d (n)

(c) HalfCheetah (m) (d) Walker2d (m)

Figure 8: Performance of BiERL with different population sizes n or m, given the same number of iterations.

(a) HalfCheetah (b) Walker2d

Figure 9: Ablation experiments where the population representation, LSTM encoder, and warm starting are removed separately.

lation information; (2) **PM without LSTM encoder** represents that the meta-level solely removes the LSTM encoder; (3) **PM without warm starting** represents that randomly initializing the meta-level instead of using the warm starting mechanism.

Figure 9 presents the ablation results of the three parts. The parametric model conditioned on each part of BiERL still exhibits performance improvement over both tasks, suggesting that all parts play important roles. While PM without the population representation learns faster during the beginning learning stage, it finally obtains lower returns, which could be fallen into the deceptive trap without sharing information with the exploration in the parameter space. When removing the LSTM encoder module leads to a degradation in learning performance across both tasks, it implies that the LSTM encoder module can help the agent efficiently capture the history information and obtain better performance. We also observe that PM without warm starting learns inefficiently at the early stage, yet its overall performance is slightly worse than PM. This result demonstrates the importance of a warm starting in complex domains, where pretraining in a simple task only yields successful results for finetuning in the complex target task. In summary, our proposed PM model conditioned on all parts gives the best performance, indicating that population representation, the LSTM encoder, and the warm starting parts are complementary parts of BiERL.
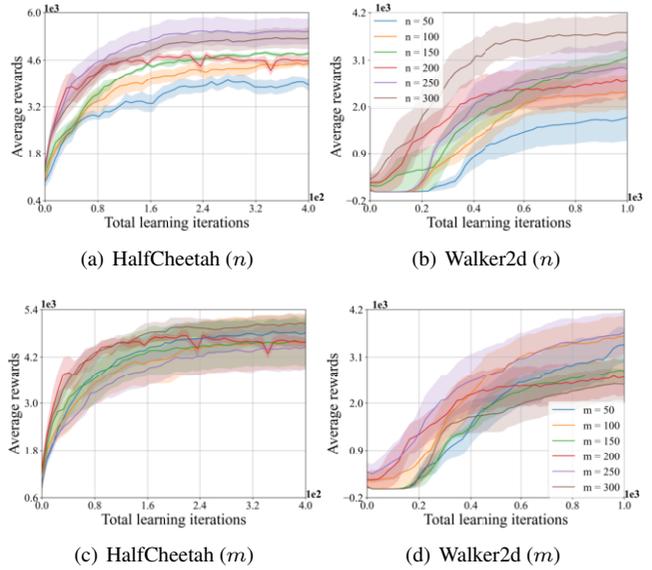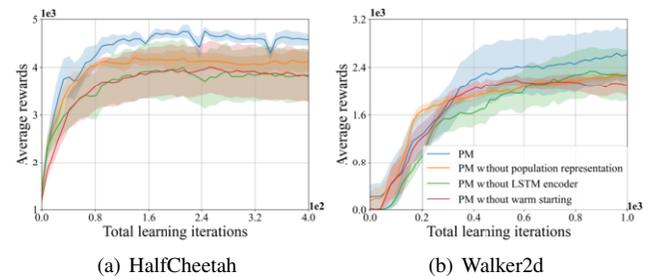
5 Conclusion

We propose a general BiERL framework to realize efficient meta-parameter optimization without prior domain knowledge or costly optimization procedure. We introduce an elegant meta-level architecture that efficiently adjusts hyperparameters according to the informative representation of the inner-level's evolving experience, and we design a simple and feasible evaluation of the meta-level fitness function to facilitate learning efficiency. Empirically, our method provides significant performance gains over representative baselines on a range of RL tasks. We believe that our work can initiate several interesting future research directions. For instance, one can explore different types of meta-level architecture, other than the LSTM and MLP networks used here. Another interesting future work could be combining BiERL with representation learning in behavior space and analyzing the performance of BiERL theoretically.

References

[1] Oladayo S Ajani and Rammohan Mallipeddi, 'Adaptive evolution strategy with ensemble of mutations for Reinforcement Learning', *Knowledge-Based Systems*, **245**, 108624, (2022).

[2] Sanjeev Arora, Simon Du, Sham Kakade, et al., 'Provable representation learning for imitation learning via bi-level optimization', in *Proceedings of International Conference on Machine Learning*, pp. 367–376. PMLR, (2020).

[3] Jordan Ash and Ryan P Adams, 'On warm-starting neural network training', in *Proceedings of Advances in Neural Information Processing Systems*, volume 33, pp. 3884–3894, (2020).

[4] Sungyong Baik, Myungsub Choi, Janghoon Choi, et al., 'Meta-learning with adaptive hyperparameters', in *Proceedings of Advances in Neural Information Processing Systems*, volume 33, pp. 20755–20765, (2020).

[5] Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales, 'EvoGrad: Efficient gradient-based meta-learning and hyperparameter optimization', in *Proceedings of Advances in Neural Information Processing Systems*, volume 34, pp. 22234–22246, (2021).

[6] David M Bossens, Jean-Baptiste Mouret, and Danesh Tarapore, 'Learning behaviour-performance maps with meta-evolution', in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 49–57, (2020).

[7] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker Holder, et al., 'Provably robust blackbox optimization for reinforcement learning', in *Proceedings of Conference on Robot Learning*, pp. 683–696. PMLR, (2020).

[8] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, et al., 'Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents', in *Proceedings of Advances in Neural Information Processing Systems*, volume 31, (2018).

[9] Stefan Elfwing, Eiji Uchibe, and Kenji Doya, 'Online meta-learning by parallel algorithm competition', in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 426–433, (2018).

[10] Alhussein Fawzi, Matej Balog, Aja Huang, et al., 'Discovering faster matrix multiplication algorithms with reinforcement learning', *Nature*, **610**(7930), 47–53, (2022).

[11] Chrisantha Fernando, Jakub Sygnowski, Simon Osindero, et al., 'Meta-learning by the Baldwin effect', in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1313–1320, (2018).

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine, 'Model-agnostic meta-learning for fast adaptation of deep networks', in *Proceedings of International Conference on Machine Learning*, pp. 1126–1135. PMLR, (2017).

[13] Alexander Gajewski, Jeff Clune, Kenneth O Stanley, et al., 'Evolvability ES: scalable and direct optimization of evolvability', in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 107–115, (2019).

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, et al., 'OpenAI Gym', *arXiv preprint arXiv:1606.01540*, (2016).

[15] Bin Gu, Guodong Liu, Yanfu Zhang, et al., 'Optimizing large-scale hyperparameters via automated learning algorithm', *arXiv preprint arXiv:2102.09026*, (2021).

[16] Jianye Hao, Pengyi Li, Hongyao Tang, Yan Zheng, Xian Fu, and Zhaopeng Meng, 'ERL-Re^2: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation', in *Proceedings of International Conference on Learning Representations*, (2023).

[17] Michael Hellwig and Hans-Georg Beyer, 'Analysis of a meta-ES on a conically constrained problem', in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 673–681, (2019).

[18] Rein Houthooft, Yuhua Chen, Phillip Isola, et al., 'Evolved policy gradients', in *Proceedings of Advances in Neural Information Processing Systems*, volume 31, (2018).

[19] Max Jaderberg, Valentin Dalibard, Simon Osindero, et al., 'Population based training of neural networks', *arXiv preprint arXiv:1711.09846*, (2017).

[20] Shauharda Khadka, Somdeb Majumdar, Tarek Nassar, et al., 'Collaborative evolutionary reinforcement learning', in *Proceedings of International Conference on Machine Learning*, pp. 3341–3350. PMLR, (2019).

[21] Shauharda Khadka and Kagan Tumer, 'Evolution-guided policy gradient in reinforcement learning', in *Proceedings of Advances in Neural Information Processing Systems*, volume 31, (2018).

[22] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', in *Proceedings of International Conference on Learning Representations*, (2015).

[23] Hae Beom Lee, Hayeon Lee, JaeWoong Shin, et al., 'Online hyperparameter meta-learning with hypergradient distillation', in *Proceedings of International Conference on Learning Representations*, (2021).

[24] Petro Liashchynskyi and Pavlo Liashchynskyi, 'Grid search, random search, genetic algorithm: a big comparison for NAS', *arXiv preprint arXiv:1912.06059*, (2019).

[25] Matthew Mackay, Paul Vicol, Jonathan Lorraine, et al., 'Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions', in *Proceedings of International Conference on Learning Representations*, (2018).

[26] Somdeb Majumdar, Shauharda Khadka, Santiago Miret, et al., 'Evolutionary reinforcement learning for sample-efficient multiagent coordination', in *Proceedings of International Conference on Machine Learning*, pp. 6651–6660. PMLR, (2020).

[27] Dong Pang, Xinyi Le, and Xinping Guan, 'RL-DARTS: Differentiable neural architecture search via reinforcement-learning-based meta-optimizer', *Knowledge-Based Systems*, **234**, 107585, (2021).

[28] Jack Parker Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts, 'Effective diversity in population based reinforcement learning', in *Advances in Neural Information Processing Systems*, volume 33, pp. 18050–18062, (2020).

[29] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, et al., 'Parameter space noise for exploration', in *Proceedings of International Conference on Learning Representations*, (2018).

[30] Tim Salimans, Jonathan Ho, Xi Chen, et al., 'Evolution strategies as a scalable alternative to reinforcement learning', *arXiv preprint arXiv:1703.03864*, (2017).

[31] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb, 'A review on bilevel optimization: from classical to evolutionary approaches and applications', *IEEE Transactions on Evolutionary Computation*, **22**(2), 276–295, (2017).

[32] Xingyou Song, Wenbo Gao, Yuxiang Yang, et al., 'ES-MAML: Simple Hessian-Free Meta Learning', in *Proceedings of International Conference on Learning Representations*, (2019).

[33] Bradly Stadie, Lunjun Zhang, and Jimmy Ba, 'Learning intrinsic rewards as a bi-level optimization problem', in *Proceedings of Conference on Uncertainty in Artificial Intelligence*, pp. 111–120. PMLR, (2020).

[34] Karush Suri, 'Off-policy evolutionary reinforcement learning with maximum mutations', in *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, pp. 1237–1245, (2022).

[35] Yunhao Tang and Krzysztof Choromanski, 'Online hyper-parameter tuning in off-policy learning via evolutionary strategies', *arXiv preprint arXiv:2006.07554*, (2020).

[36] Emanuel Todorov, Tom Erez, and Yuval Tassa, 'MuJoCo: A physics engine for model-based control', in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, (2012).

[37] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, et al., 'Grandmaster level in StarCraft II using multi-agent reinforcement learning', *Nature*, **575**(7782), 350–354, (2019).

[38] Zhi Wang, Chunlin Chen, and Daoyi Dong, 'Instance weighted incremental evolution strategies for reinforcement learning in dynamic environments', *IEEE Transactions on Neural Networks and Learning Systems, DOI: 10.1109/TNNLS.2022.3160173*, (2022).

[39] Zhi Wang, Chunlin Chen, and Daoyi Dong, 'Lifelong incremental reinforcement learning with online Bayesian inference', *IEEE Transactions on Neural Networks and Learning Systems*, **33**(8), 4003–4016, (2022).

[40] Yuanyang Zhu, Zhi Wang, Chunlin Chen, and Daoyi Dong, 'Rule-based reinforcement learning for efficient robot navigation with space reduction', *IEEE/ASME Transactions on Mechatronics*, **27**(2), 846–857, (2021).