

# ULTHO: Ultra-Lightweight yet Efficient Hyperparameter Optimization in Deep Reinforcement Learning

Mingqi Yuan<sup>1</sup>, Bo Li<sup>1</sup>, Xin Jin<sup>2,3\*</sup>, Wenjun Zeng<sup>2,3</sup>

<sup>1</sup>Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China

<sup>2</sup>Ningbo Institute of Digital Twin, Eastern Institute of Technology, Ningbo, China

<sup>3</sup>Ningbo Key Laboratory of Spatial Intelligence and Digital Derivative, Ningbo, China

## Abstract

Hyperparameter optimization (HPO) is a billion-dollar problem in machine learning, which significantly impacts the training efficiency and model performance. However, achieving efficient and robust HPO in deep reinforcement learning (RL) is consistently challenging due to its high non-stationarity and computational cost. To tackle this problem, existing approaches attempt to adapt common HPO techniques (e.g., population-based training or Bayesian optimization) to the RL scenario. However, they remain sample-inefficient and computationally expensive, which cannot facilitate a wide range of applications. In this paper, we propose ULTHO, an ultra-lightweight yet powerful framework for fast HPO in deep RL within single runs. Specifically, we formulate the HPO process as a multi-armed bandit with clustered arms (MABC) and link it directly to long-term return optimization. ULTHO also provides a quantified and statistical perspective to filter the HPs efficiently. We test ULTHO on benchmarks including ALE, Procgen, MiniGrid, and PyBullet. Extensive experiments demonstrate that the ULTHO can achieve superior performance with a simple architecture, contributing to the development of advanced and automated RL systems. Our code is available at the [GitHub repository](https://github.com/yuanmingqi/ULTHO)<sup>1</sup>.

## 1. Introduction

Deep reinforcement learning (RL) has propelled significant advancements across various fields, such as game playing [46, 55, 60], chip design [25], algorithm innovation [22, 43], and large language model (LLM) [6, 12, 41, 47]. However, training deep RL agents involves a number of design decisions and hyperparameter configurations, in which minor variations can result in substantial effects on learning efficiency and final performance [19]. While many efforts

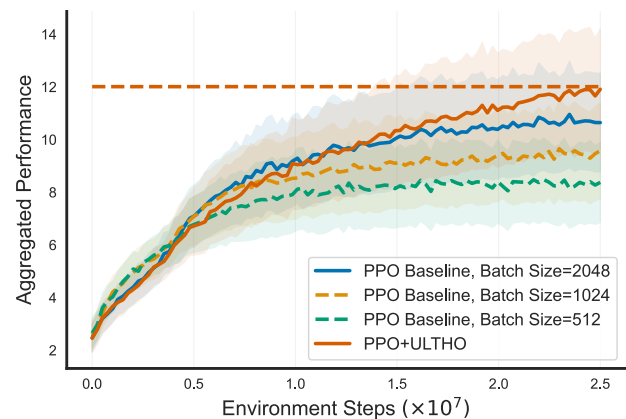


Figure 1. Aggregated performance comparison on the Procgen benchmark with procedurally-generated environments. The choice of batch size significantly impacts the agent’s performance. ULTHO can effectively perform HPO across different tasks and learning stages, thereby improving the overall performance.

have been devoted to improving the design choices [20, 28–30] and implementation details [3, 31, 51, 67], RL-specific hyperparameter optimization (HPO) does not receive sufficient attention in the community. This gap is particularly notable given the high sensitivity of deep RL algorithms to HPs, which restricts their broader applications and the development of advanced and automated RL systems [24].

HPO is a fundamental problem in machine learning, with traditional methods like grid search [64] and random search [11] being widely used to automate the process. More advanced techniques, such as Bayesian optimization [56] and multi-fidelity approaches like Hyperband [39], have been developed to improve efficiency by leveraging surrogate models and early stopping mechanisms. However, these methods are primarily designed for static optimization problems, where the optimal HPs remain fixed throughout the training process. In contrast, deep RL operates in non-stationary environments where the optimal HPs may vary across different training stages, especially in dynamic environments such as Procgen [16] and Craftax [44]. This non-

\*Corresponding author

<sup>1</sup><https://github.com/yuanmingqi/ULTHO>

stationarity, along with the high computational cost of deep RL training, makes traditional HPO methods less effective for these applications.

To tackle the HPO challenge in deep RL, several approaches have been proposed to dynamically adjust HPs during training. For instance, [33] introduced population-based training (PBT) that evolves a population of agents to explore and exploit different HP schedules, enabling adaptation to non-stationary settings. Notably, PBT is utilized to solve the complex 3D multiplayer game, Quake III Arena, achieving human-level performance by optimizing the agent policies along with internal reward signals in a hierarchical manner [34]. [24] further extended PBT and proposed SEARL, which optimizes the HPs and also the neural network architecture while simultaneously training the agent, significantly improving sample efficiency by sharing experience across the population. However, PBT-like methods remain sample-inefficient and computationally expensive, limiting their practicality to a wider range of tasks. To address these issues, single-run HPO methods such as HOOF [50] have been developed. HOOF adapts HPs using off-policy importance sampling, optimizing a one-step improvement objective with sampled trajectories. As a gradient-free algorithm, HOOF can significantly reduce overhead and promote sample efficiency. However, HOOF greedily focuses on HP configurations that maximize the value of the updated policy without introducing an exploration mechanism. Moreover, its reliance on importance sampling also makes it prone to high variance when policy distributions shift, hurting its robustness in highly dynamic environments.

Inspired by the discussions above, we propose **ULTHO**: **Ultra-Lightweight Hyperparameter Optimization**, a general, simple, yet powerful framework for achieving efficient and robust HPO in deep RL within single runs. Unlike previous methods that focus on short-term improvements or require complex learning processes, ULTHO performs HPO from the perspective of the hierarchical bandits, ensuring long-term performance gains with minimal computational overhead. Our main contributions are summarized as follows:

- We formulate the HPO process as a multi-armed bandit with clustered arms (MABC), which optimizes HPs adaptively across different tasks and learning stages in a hierarchical fashion and effectively reduces the sample complexity. ULTHO selects the appropriate HPs based on the estimated task return, ensuring the maximization of long-term returns while balancing exploration;
- Our framework currently provides two algorithms, *i.e.*, a normal version and an extended version for continual optimization, solving HPO within single runs

and providing a quantified and statistical perspective to analyze the potential of distinct HPs. In particular, ULTHO has a simple architecture and requires no additional learning processes, which can be compatible with a broad range of RL algorithms;

- Finally, we evaluate ULTHO on ALE (arcade game environments), Procgen (sixteen procedurally-generated environments), MiniGrid (environments with sparse rewards), and PyBullet (robotics environments with continuous action space). Extensive experiments demonstrate that ULTHO can achieve superior performance with remarkable computational efficiency.

## 2. Related Work

### 2.1. HPO in Machine Learning

HPO is essential in machine learning as it significantly impacts model performance, convergence speed, and generalization ability. Prominent techniques, including random and grid search [11], Bayesian optimization (BO) [56, 59, 62], multi-fidelity search strategies [21, 35, 39], gradient-based methods [13, 42, 45, 63], population-based training [33, 48, 61], and RL-based methods [18, 32, 36]. For instance, [21] proposes BOHB that combines BO and Hyperband to balance the exploration-exploitation trade-off while dynamically allocating resources, significantly improving efficiency and robustness over a single method. [36] further extends PBT and proposed population-based bandits (PB2) by incorporating a MAB strategy to adaptively explore HP schedules, improving sample efficiency over standard PBT. [36] formulates HPO as a sequential decision problem and solves it with RL, which enables adaptive HP tuning and reduces reliance on hand-crafted acquisition functions. However, the methods above are prone to be computationally expensive and less effective for RL scenarios.

In this paper, we solve the HPO problem from the perspective of hierarchical bandits, achieving efficient scheduling within single runs and significantly reducing the computational overhead.

### 2.2. HPO in Reinforcement Learning

Extensive research [19, 20, 28, 30, 58, 69] has shown the importance of HPO in deep RL. However, directly applying HPO methods from general machine learning to deep RL is consistently inefficient due to its non-stationarity and high computational cost. One promising approach to address this is meta-gradients [63], which dynamically adjust HPs during training. For instance, [68] uses meta-gradient descent to self-tune all the differentiable HPs of an actor-critic loss function and discover auxiliary tasks, while improving off-policy learning using a novel leaky V-trace operator. [23] further enhances this by bootstrapping the meta-

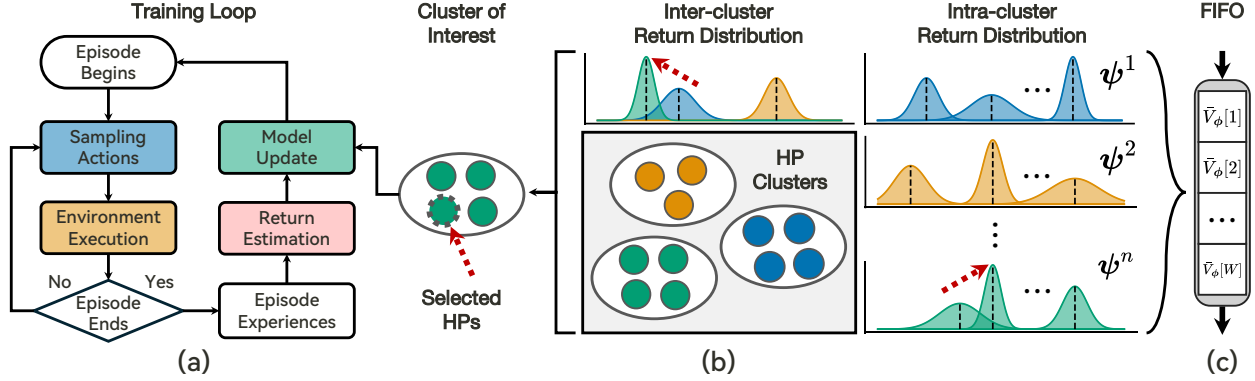


Figure 2. Overview of ULTHO framework. (a) The key phases of RL algorithms. ULTHO serves as a plug-and-play module that feeds the optimized HPs to the RL algorithm. (b) ULTHO maintains inter-cluster and intra-cluster return distributions to perform HPO in a hierarchical manner. (c) A sliding window is used to store the estimated task returns for updating the return distributions.

learning process, allowing agents to meta-learn more efficiently across tasks and providing robust performance in dynamic environments. However, these methods require access to the algorithms’ gradients and result in more computational overhead, limiting their use in common RL settings.

In this paper, we optimize HPs based solely on the estimated task returns, avoiding complex learning processes and not requiring access to internal data such as gradients, thereby facilitating a wider range of RL algorithms.

### 2.3. MAB Algorithms for Deep RL

MAB problems are closely related to RL, as both involve decision-making under uncertainty [5]. While RL focuses on sequential decisions to maximize cumulative rewards, MAB methods optimize immediate actions, making them effective for addressing subproblems within RL frameworks. For example, [52] proposed UCB-DrAC, which employs a bandit algorithm to select optimal data augmentations, significantly improving generalization in procedurally-generated environments. Similarly, AIRS [65] formulates intrinsic reward selection as a bandit problem, dynamically adapting rewards to enhance exploration at different learning stages. Finally, [66] applies bandit algorithms to adaptively control the data exploitation, enhancing the data efficiency and generalization while reducing the overall computational overhead.

In this paper, our framework unifies the methods above by extending the MAB problem with clustered arms, enabling more efficient HPO via hierarchical exploration.

## 3. Background

### 3.1. Reinforcement Learning

We study the RL problem considering a Markov decision process (MDP) [10, 37] defined by a tuple  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, d_0, \gamma)$ , where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  is the action space, and  $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  is the extrinsic reward function,  $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$  is the transition function that

defines a probability distribution over  $\mathcal{S}$ ,  $d_0 \in \Delta(\mathcal{S})$  is the distribution of the initial observation  $s_0$ , and  $\gamma \in [0, 1]$  is a discount factor. The goal of RL is to learn a policy  $\pi_\theta(\mathbf{a}|s)$  to maximize the expected discounted return:

$$J_\pi(\theta) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, \mathbf{a}_t) \right]. \quad (1)$$

### 3.2. HPO in RL

HPO in RL aims to find the optimal set of HPs that maximizes the expected return of the agent’s policy [19, 24]. In this context, HPs refer to parameters that are not directly learned by the RL algorithm but influence the learning process, such as learning rates, exploration parameters, and network architecture choices. The objective of HPO in RL can be formulated as solving the following black-box optimization problem:

$$\psi^* = \operatorname{argmax}_{\psi \in \Psi} J_\pi(\theta, \psi, \mathbf{A}), \quad (2)$$

where  $\psi^*$  is the optimal configuration,  $\Psi$  is the HP space, and  $\mathbf{A}$  is an RL algorithm.

## 4. The ULTHO Framework

In this section, we introduce ULTHO, an ultra-lightweight, unified, yet powerful framework designed for efficient and robust HPO in RL, whose overview is presented in Figure 2. Our key insights are as follows: (i) Similar to [50] and [68], ULTHO also dynamically tunes HPs within a single training run, enabling adaptation to different learning stages and significantly reducing the trial-and-error cost; (ii) ULTHO adopts a hierarchical approach [7, 34] to organize and select the HP candidates. Specifically, we perform a two-tier HPO to traverse the search space more efficiently; (iii) Unlike prior methods [33, 50, 68] often prioritizes short-term improvements, ULTHO focuses on optimizing long-term returns, providing a systematic approach

---

**Algorithm 1** The ULTHO with UCB

---

- 1: Initialize the policy network  $\pi_\theta$  and value network  $V_\phi$ ;
- 2: Initialize a set  $\Psi$  of HPs, an exploration coefficient  $c$ , a window length  $W$  for estimating the utility functions;
- 3:  $\forall \psi \in \Psi, \forall \psi \in \psi$ , let

$$N(\psi) = 1, U(\psi) = 0, R(\psi) = \text{FIFO}(W)$$
$$N(\psi) = 1, U(\psi) = 0, R(\psi) = \text{FIFO}(W)$$

- 4: **for** each episode  $e$  **do**
  - 5:   Sample rollouts using the policy network  $\pi_\theta$ ;
  - 6:   Perform the generalized advantage estimation (GAE) to get the estimated task returns;
  - 7:   Select a cluster  $\psi_e$  using Eq. (5);
  - 8:   Select a HP  $\psi_e$  from the  $\psi_e$  using Eq. (6);
  - 9:   Update policy network and value network;
  - 10:   Compute  $\bar{V}_\phi$  obtained by the new policy;
  - 11:   Add  $\bar{V}_\phi$  to the queue  $R(\psi_e)$  and  $R(\psi_e)$ ;
  - 12:   Update  $U(\psi_e)$  and  $U(\psi_e)$  using Eq. (7);
  - 13:    $N(\psi_e) \leftarrow N(\psi_e) + 1, N(\psi_e) \leftarrow N(\psi_e) + 1$ .
  - 14: **end for**
- 

that ensures more sustainable performance gains; (iv) To facilitate a wide range of RL algorithms, ULTHO requires no additional learning processes or access to internal data, such as gradients, making it universally applicable without requiring complex modifications to the underlying algorithm.

#### 4.1. MABC for HPO

To simplify the notations, we reuse  $\Psi$  to denote the set of HP clusters:

$$\Psi = \{\psi^1, \psi^2, \dots, \psi^n\}, \quad (3)$$

where

$$\psi^i = \{\psi_1^i, \psi_2^i, \dots, \psi_m^i\} \quad (4)$$

is an individual cluster of HPs.

Then, the HPO at different learning stages can be formulated as a multi-armed bandit with clustered arms (MABC) [14], and the optimization objective is to maximize the long-term return evaluated by the task reward function.

We solve the defined MABC problem by hierarchically applying the upper confidence bound (UCB) [4] algorithm, which is an effective and widely-used method to balance exploration-exploitation in bandit problems. Specifically, at the  $i$ -th episode, we first select a cluster from  $\Psi$  by the following policy:

$$\underbrace{\psi_i = \operatorname{argmax}_{\psi \in \Psi} \left[ U_i(\psi) + c \sqrt{\frac{\log i}{N_i(\psi)}} \right]}_{\text{Inter-cluster Scheduling Policy}}, \quad (5)$$

---

**Algorithm 2** The Relay-ULTHO

---

- 1: Execute the Algorithm 1;
  - 2: Get the  $\psi^{\text{COI}}$  and  $\psi^{\text{NOI}}$  using Eq. (8);
  - 3: Execute Algorithm 1 solely with  $\psi^{\text{COI}}$  or  $\psi^{\text{NOI}}$ ;
  - 4: Output the best-performing cluster.
- 

where  $N_i(\psi)$  is the number of times that  $\psi$  has been chosen before the  $i$ -th episode, and  $c$  is the exploration coefficient.

Equipped with the selected cluster, we then select a specific HP value following a similar policy:

$$\underbrace{\psi_i = \operatorname{argmax}_{\psi \in \psi_i} \left[ U_i(\psi) + c \sqrt{\frac{\log i}{N_i(\psi)}} \right]}_{\text{Intra-cluster Scheduling Policy}}. \quad (6)$$

The selected HP value will be used for policy updates. Then the utility functions of the cluster and the corresponding value are updated using a sliding window average of the past mean episode returns obtained by the agent:

$$\underbrace{U_i(\psi) = \frac{1}{W} \sum_{j=1}^W \bar{V}_\phi[j], \bar{V}_\phi = \frac{1}{T} \sum_{t=1}^T V_\phi(s_t)}_{\text{Utility Evaluation}}, \quad (7)$$

where  $V_\phi(s_t)$  is the estimated return predicted by the value network,  $T$  is the episode length, and  $W$  is the window length.

#### 4.2. Relay-ULTHO

Algorithm 1 summarizes the workflow of ULTHO with UCB. It is evident that the ULTHO changes only one HP at a time, which helps stabilize the learning process due to the high sensitivity of RL to HPs. However, optimizing multiple HPs simultaneously can lead to unstable learning behavior, slower convergence, and potentially sub-optimal policies in certain environments. Moreover, it is unclear whether all HPs consistently affect performance across different environments and learning stages. With this in mind, we propose an extended algorithm entitled Relay-ULTHO, as depicted in Algorithm 2.

The core idea behind Relay-ULTHO is to identify two HP clusters: the cluster of interest (COI) and the neglected cluster of interest (NOI). COI is the cluster that is selected most frequently, indicating it has the greatest impact on the agent's performance. In contrast, NOI is the cluster with the least number of selections. After executing Algorithm 1 with an initial set of clusters  $\Psi$ , we compute the total count  $N_{\text{end}}(\psi)$  for each cluster. Then we define the COI and NOI as

$$\psi^{\text{COI}} = \operatorname{argmax}_{\psi \in \Psi} N_{\text{end}}(\psi), \quad (8)$$
$$\psi^{\text{NOI}} = \operatorname{argmin}_{\psi \in \Psi} N_{\text{end}}(\psi).$$

Once the COI and NOI are identified, the optimization process is focused on these clusters, allowing the algorithm to refine the most influential cluster and further explore the neglected cluster. This approach can improve the completeness of exploration, thereby squeezing out additional performance gains. However, we also need to highlight that this algorithm may result in sub-optimal performance due to its restricted landscape.

## 5. Experiments

In this section, we design the experiments to investigate the following questions:

- **Q1:** Can ULTHO improve performance as compared to using fixed HP values? (See Figure 4, 5, 9, 10, 20, 21, and Table 1)
- **Q2:** Can the relay optimization further enhance the ULTHO algorithm? (See Figure 4, 5, 9, 10, 20, 21, and Table 1)
- **Q3:** What are the detailed decision processes of the ULTHO algorithm? (See Figure 6, 7, 24, and 25)
- **Q4:** How does ULTHO behave in sparse-rewards environments and continuous control tasks? (See Figure 9 and 10)
- **Q5:** How robust is the ULTHO algorithm? (See Figure 8)

### 5.1. Setup

#### 5.1.1. Benchmark Selection

We first evaluate the ULTHO using the arcade learning environment (ALE) benchmark [9], a collection of arcade game environments that requires the agent to learn motor control directly from images. Specifically, we focus on a subset of ALE known as ALE-5, which typically produces median score estimates for all games that are within 10% of their true values [2]. For all environments, we stack 4 consecutive frames to form an input state with the data shape of (84, 84, 4). Additionally, we introduce the Procgen benchmark with 16 procedurally-generated environments. Procgen is similar to the ALE benchmark yet involves much higher dynamicity and presents a more difficult challenge for HPO. All the environments use a discrete fifteen-dimensional action space and generate (64, 64, 3) RGB observations, and we use the *easy* mode and train the agents on 200 levels before testing them on the full distribution of levels. Furthermore, we introduce the MiniGrid [15] and PyBullet [17] to test ULTHO in sparse-rewards environments and continuous control tasks.

#### 5.1.2. Algorithmic Baselines

For the RL algorithm, we select the proximal policy optimization (PPO) [54] as the baseline, which is a representative algorithm that produces considerable performance on

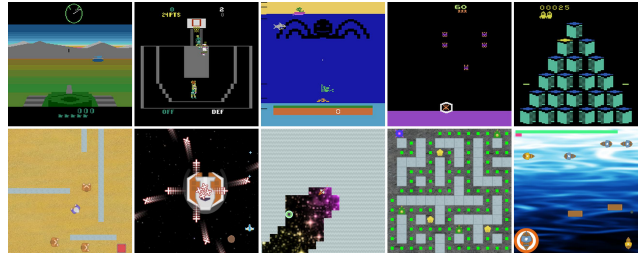


Figure 3. Screenshots of the ALE (top) and Procgen (below) benchmarks.

most existing RL benchmarks. For the HPO algorithms, we select random search (RS) [11], PBT [33], PB2 [48], the Bayesian optimization tool SMAC [40], the combination of SMAC and the Hyperband scheduler (SAMC+HB) [39], and TMIHF [49]. The details of the selected algorithmic baselines can be found in Appendix A.

#### 5.1.3. HP Clusters

We select HP clusters based on the prior practice reported in [24, 30, 33, 50, 66]. Specifically, our HP clusters include the learning rate (LR), batch size (BS), value loss coefficient (VLC), entropy loss coefficient (ELC), and the number of update epochs (NUE). To ensure a balanced exploration, we let each cluster have the same number of values. For the detailed configuration of each cluster, please refer to Appendix B.

#### 5.1.4. Evaluation Metrics

For the baselines that require multiple training runs (*e.g.*, PBT), we use the maximum observed past scores as their final performance. For our ULTHO algorithm, we simply use the observed episode return at the end of training as its performance. Note that the score of each method on each environment is computed as the average episode returns over 100 episodes and 5 random seeds. Additionally, we compare computational efficiency by evaluating the required training budgets for all the methods.

## 5.2. Results Analysis

The following results analysis is performed based on the pre-defined research questions. We provide the detailed training curves of all the methods and configurations in Appendix C.

#### 5.2.1. Comparison with HPO Baselines

We first compare the performance of ULTHO and four HPO baselines on the ALE benchmark, and their normalized scores and training budgets are illustrated in Figure 4. Here, the baseline PPO is trained using the common HPs reported in its original paper [54]. ULTHO successfully outperforms the PPO baseline in all five environments and achieves the highest performance in two environments, especially in *Q\*Bert* environment. In contrast, the SMAC and

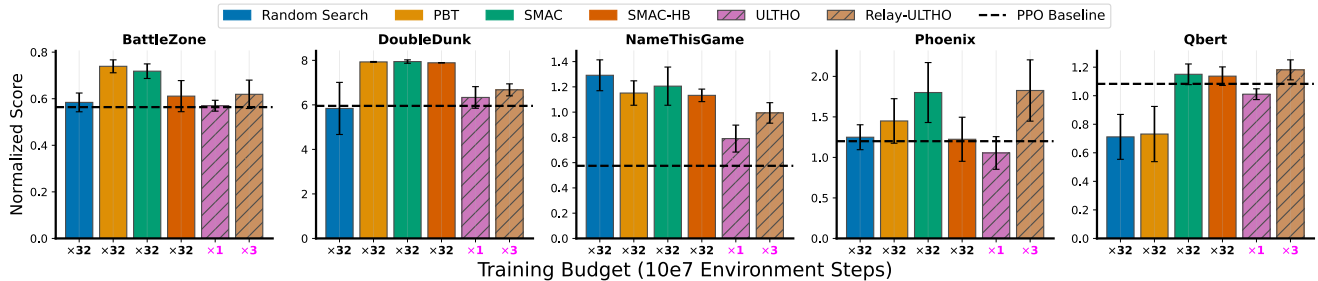


Figure 4. Performance comparison of the two ULTHO algorithms and HPO baselines on the ALE-5 benchmark, in which the mean and standard error are computed using five random seeds. All the scores are normalized using the min-max normalization with the human expert performance. The two ULTHO algorithms achieve similar or even higher performance with a few training budgets as compared to other HPO baselines. An ablation study that uses the same training budgets for all the methods can be found in Appendix D.

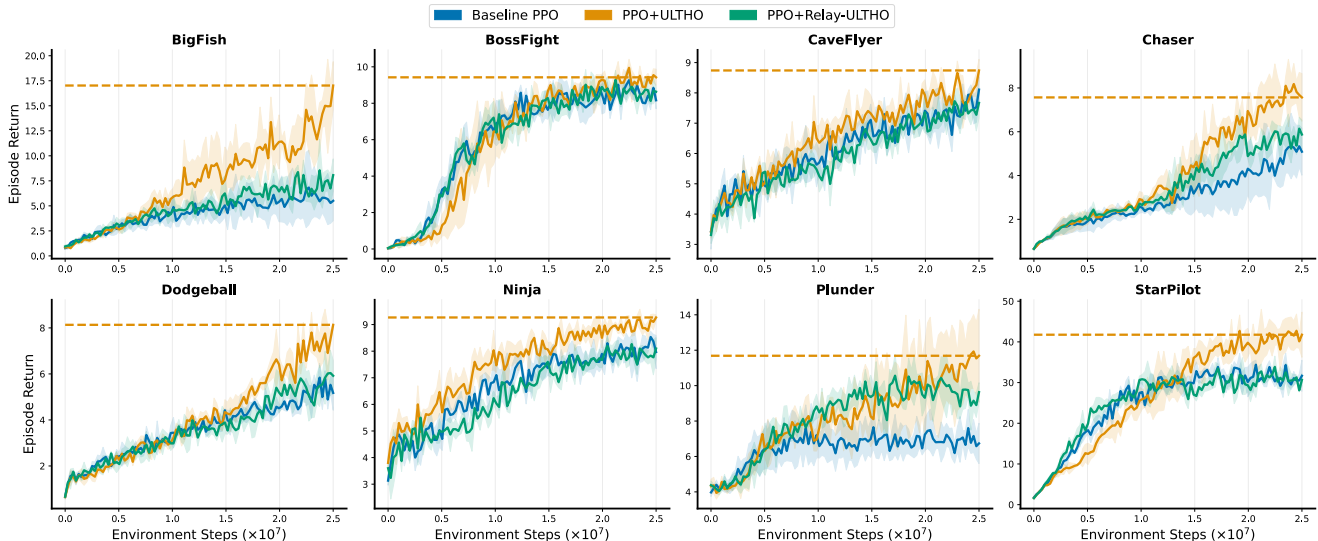


Figure 5. Training performance of the PPO and its combinations with two ULTHO algorithms in eight Procgen environments, in which the mean and standard deviation are computed using five random seeds. ULTHO consistently outperforms the vanilla PPO agent in all the environments, significantly improving the sample efficiency. However, the Relay-ULTHO does not achieve any further performance gains, indicating that a more comprehensive policy, which continuously adapts multiple HPs, is crucial for highly dynamic environments.

SMAC+HB rank second and third in terms of average performance across all the environments. The PBT method excels in the *BattleZone* environment, but it fails to outperform the baseline PPO in the *Q\*bert* environment. By adaptively tuning HPs within the training run, ULTHO can achieve remarkable efficiency with much lower computational cost as compared to the baselines.

Next, we report the performance of the ULTHO on the Procgen benchmark. Similarly, the baseline PPO is trained using the reported HPs in [16]. Figure 5 illustrates the training performance comparison in eight environments, and the full curves are provided in Figure 20. ULTHO outperforms the baseline PPO in 15 environments, achieving significant performance gains in environments like *BigFish*, *Chaser*, and *StarPilot*. Additionally, we evaluate ULTHO on the full distribution of levels, and Table 1 illustrates the performance comparison with three HPO baselines. ULTHO outperforms these baselines regarding both the aggregated

Method	Agg. Mean	Agg. IQM
PPO	40.42±25.18	38.41±6.59
PBT <sup>‡</sup>	33.55±23.49	29.27±13.68
PB2 <sup>‡</sup>	44.63±23.56	40.5±11.41
TMIHF <sup>‡</sup>	51.16±20.16	48.7±4.67
ULTHO	<b>56.0±21.87</b>	<b>56.41±6.95</b>
Relay-ULTHO	48.04±26.88	49.01±5.0

Table 1. Normalized test performance (%) comparison on the Procgen benchmark. Here, **IQM** denotes the interquartile mean suggested by [1], and <sup>‡</sup> indicates the training is conducted by a population of agents. Similar to the training performance comparison, the normal ULTHO method achieves the highest score by providing a more comprehensive HPO policy.

mean and IQM. These results demonstrate that ULTHO can effectively select the appropriate HPs at different learning stages and significantly improve learning efficiency even in highly dynamic environments.

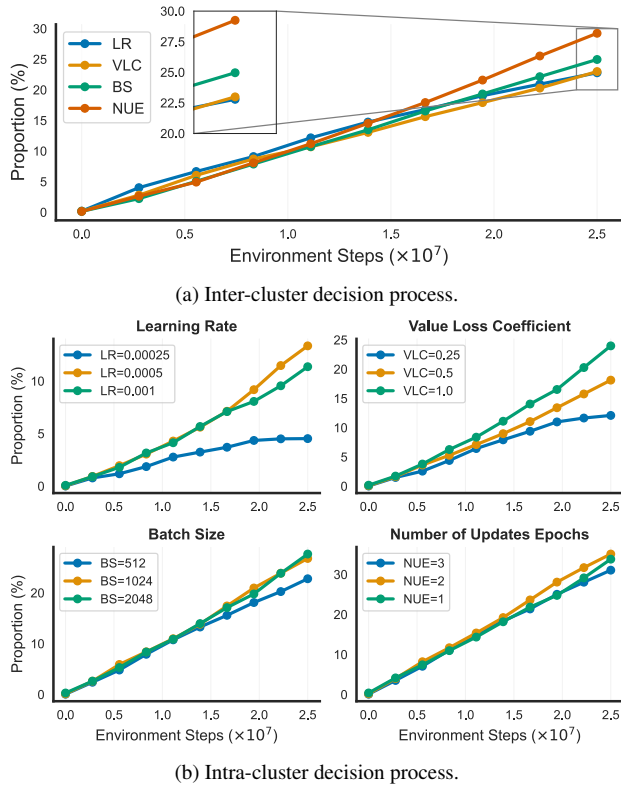


Figure 6. Aggregated inter-cluster and intra-cluster decision processes of ULTHO on the whole Procgen benchmark. Among the four HP clusters, the NUE cluster is selected the most, suggesting its high impact on overall performance.

### 5.2.2. Capability of Relay Optimization

The high sensitivity to HPs of deep RL makes it risky to optimize multiple HPs simultaneously, leading to the development of the Relay-ULTHO algorithm to enhance exploration and stability. To evaluate its effectiveness, we test it on both the ALE and Procgen benchmarks. Relay-ULTHO produces a much higher performance than the normal version, showing the incremental potential by identifying the COI and NOI from the HP space. However, Relay-ULTHO fails to outperform the normal version on the Procgen benchmark. While Relay-ULTHO is effective in environments with relatively stable dynamics, it struggles in highly non-stationary settings like Procgen, where optimal HP configurations may shift continually across training stages. Therefore, it is feasible to combine the two algorithms to realize a more robust HPO process.

### 5.2.3. The Detailed Decision Process

Furthermore, we analyze the detailed decision processes of ULTHO. Figure 6a illustrates the cumulative proportion of each cluster selected during the whole training. It is clear that ULTHO primarily selects the NUE cluster, while the other three clusters account for approximately 20%. For intra-cluster decisions, we find that the LR=5e-4, VLC=1.0, BS=2048, and NUE=2 are the most popular choices. Ad-

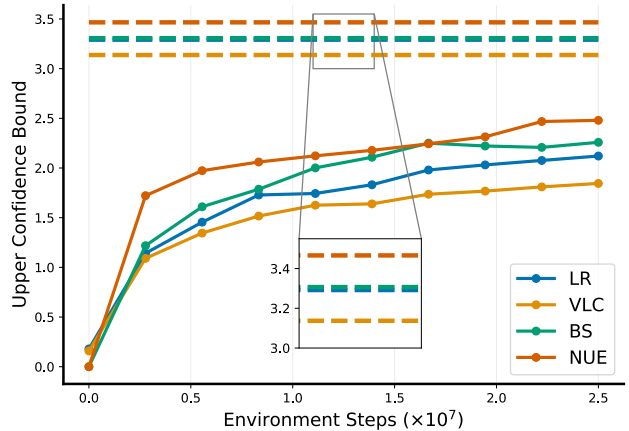


Figure 7. The variation of aggregated confidence intervals on the whole Procgen environments. Here, the solid line represents the mean value, and the dashed line represents the final upper confidence bound. It is evident that the NUE cluster obtains the highest upper confidence bound at the end of training, which aligns with the selection proportion illustrated in Figure 6.

ditionally, Figure 7 illustrates the variation of aggregated confidence intervals of these clusters during the training. ULTHO shows the explicitly distinct preferences of HP clusters, which aligns with the cumulative proportion illustrated in Figure 6a. These confidence intervals provide a statistical and quantified perspective to identify the importance of each HP cluster, contributing to a more efficient filtering approach. Finally, we provide the detailed decision processes of each method and environment in Appendix E.

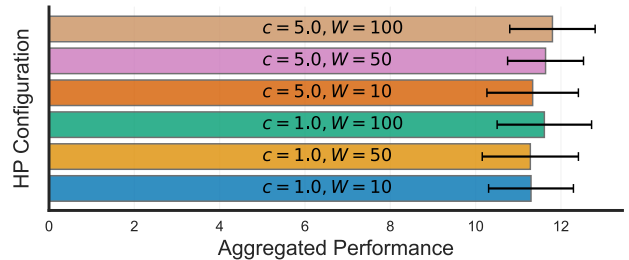


Figure 8. Aggregated performance of ULTHO on the Procgen benchmark with different configurations of  $c$  and  $W$ , and the mean and standard error are computed across all the environments. These results demonstrate that ULTHO is robust to the variation of the two internal HPs.

### 5.2.4. Ablation Studies

While ULTHO achieves efficient HPO, the utilized UCB method relies on two HPs, the exploration coefficient  $c$  and the length of the sliding window  $W$ . To evaluate the robustness of ULTHO, we perform experiments using different configurations of the two HPs on the Procgen benchmark, and the aggregated training performance is illustrated in Figure 8. It is evident that the ULTHO is relatively insensitive to the choice of  $c$  and  $W$ , yet a bigger  $W$  helps achieve a more reliable return estimation. Therefore, ULTHO main-

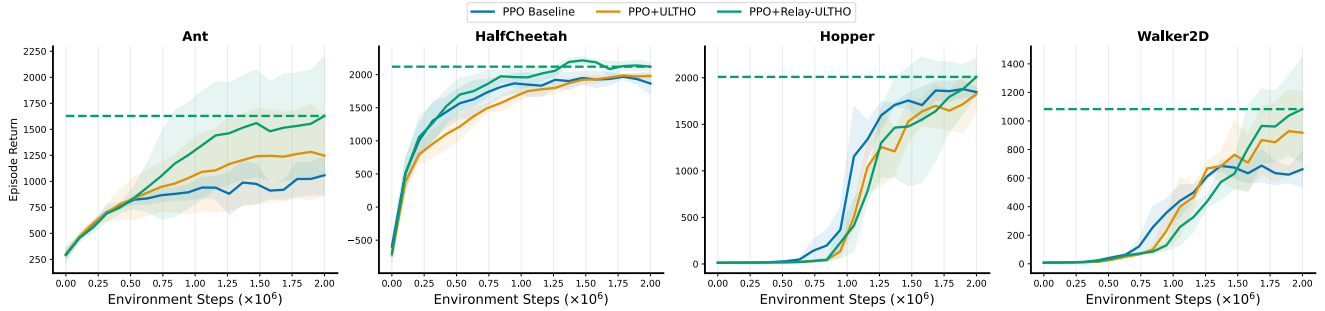


Figure 9. Performance of the PPO baseline and two ULTHO algorithms on the PyBullet benchmark, and the mean and standard deviation are computed using five random seeds. Our method can effectively improve the sample efficiency through high-quality HPO in continuous control tasks.

tains stable performance across different HPs and can be easily adapted to various training scenarios. More ablation studies regarding the number of HP clusters and training budgets can be found in Appendix D.

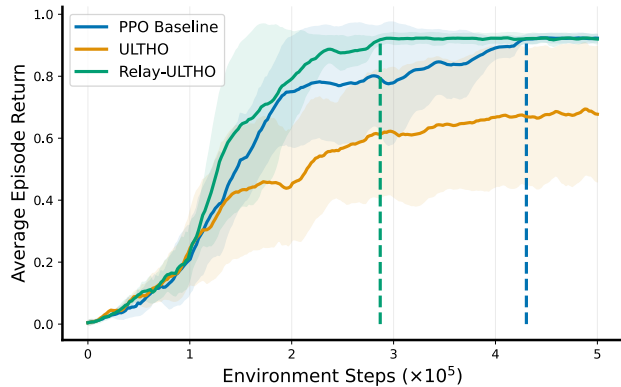


Figure 10. Aggregated performance of ULTHO on the MiniGrid benchmark, and the mean and standard deviation are computed across all the environments. For the relatively stationary environments, Relay-ULTHO can further obtain performance gains through incremental optimization.

### 5.2.5. Performance in Sparse-rewards Environments

Additionally, we evaluate ULTHO on the MiniGrid benchmark with sparse-rewards and goal-oriented environments. Specifically, we conduct experiments using *DoorKey-6x6*, *LavaGapS7*, and *Empty-16x16*. Figure 10 illustrates the aggregated learning curves of the PPO baseline and two ULTHO algorithms. While ULTHO fails to outperform the PPO baseline, the relay optimization takes fewer environment steps to solve the tasks, highlighting its capability to achieve efficient HPO in both dense and sparse-reward settings. More experimental details are provided in Appendix B.

### 5.2.6. Performance on Continuous Control Tasks

Finally, we evaluate ULTHO on the PyBullet benchmark with continuous control tasks. Four environments are utilized, namely *Ant*, *HalfCheetah*, *Hopper*, and *Walker2D*.

Figure 9 illustrates the aggregated learning curves of the vanilla PPO agent and two ULTHO methods, which produce significant performance gain on the whole benchmark. Similarly, as analyzed before, relay optimization brings advantages in relatively stationary environments. These results underscore the effectiveness of ULTHO in enhancing RL algorithms across both discrete and continuous control tasks. Additional experimental details can be found in Appendix B.

## 6. Discussion

In this paper, we investigated the problem of HPO in deep RL and proposed an ultra-lightweight yet powerful framework entitled ULTHO, which performs HPO on the fly and requires no complex learning process. ULTHO formulates the HPO process as a multi-armed bandit with clustered arms, enabling efficient and adaptive HP selection across different learning stages. We evaluate ULTHO on ALE, Procgen, MiniGrid, and PyBullet benchmarks. Extensive experiments demonstrate that ULTHO can effectively enhance RL algorithms with a simple architecture, contributing to the development of advanced and automated RL systems.

Still, there are currently remaining limitations to this work. As a lightweight framework, ULTHO currently operates on categorical HP values, which discretizes HP search into predefined clusters. While this ensures computational efficiency and stable optimization, it may limit the granularity of HP tuning in some cases. Extending ULTHO to handle continuous HP spaces for more precise tuning is a promising direction for future work. Additionally, while ULTHO uses a UCB-based strategy for exploration-exploitation balance, we have yet to explore alternative bandit algorithms, such as Exp3 or Thompson sampling. Evaluating these strategies within ULTHO could provide valuable insights into their impact on RL-specific HPO and potentially further improve performance in dynamic environments. Future work will focus on mitigating these issues and further enhancing ULTHO, making it more robust, adaptive, and efficient for HPO in deep RL.

## Acknowledgements

This work is supported, in part, by the HKSAR RGC under Grant No. PolyU 15224823, the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2024A1515011524, the NSFC under Grant No. 62302246, the ZJNSFC under Grant No. LQ23F010008, and the Ningbo under Grants No. 2023Z237 & 2024Z284 & 2024Z289 & 2023CX050011 & 2025Z038. We thank the high-performance computing center at Eastern Institute of Technology and Ningbo Institute of Digital Twin for providing the computing resources.

## References

- [1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021. 6
- [2] Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *International Conference on Machine Learning*, pages 421–438. PMLR, 2023. 5
- [3] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. 1
- [4] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002. 4
- [5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002. 3
- [6] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022. 1
- [7] Juan Cruz Barsece, Jorge Palombarini, and Ernesto Martinez. A hierarchical two-tier approach to hyper-parameter optimization in reinforcement learning. *SADIO Electronic Journal of Informatics and Operations Research*, 19(2):2–27, 2020. 3
- [8] Jannis Becktepe, Julian Dierkes, Carolin Benjamins, Aditya Mohan, David Salinas, Raghu Rajan, Frank Hutter, Holger Hoos, Marius Lindauer, and Theresa Eimer. Arlbench: Flexible and efficient benchmarking for hyperparameter optimization in reinforcement learning. In *Seventeenth European Workshop on Reinforcement Learning*. 13
- [9] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. 5
- [10] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957. 3
- [11] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012. 1, 2, 5, 12
- [12] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024. 1
- [13] Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Evograd: Efficient gradient-based meta-learning and hyperparameter optimization. *Advances in neural information processing systems*, 34:22234–22246, 2021. 2
- [14] Emil Carlsson, Devdatt Dubhashi, and Fredrik D. Johansson. Thompson sampling for bandits with clustered arms. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2212–2218. International Joint Conferences on Artificial Intelligence Organization, 2021. 4
- [15] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrad & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA, 2023*. 5, 14
- [16] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020. 1, 6
- [17] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. URL <http://pybullet.org>, 2016–2018. 5
- [18] Xingping Dong, Jianbing Shen, Wenguan Wang, Ling Shao, Haibin Ling, and Fatih Porikli. Dynamical hyperparameter optimization via deep reinforcement learning in tracking. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1515–1529, 2019. 2
- [19] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *International Conference on Machine Learning*, pages 9104–9149. PMLR, 2023. 1, 2, 3
- [20] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020. 1, 2
- [21] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pages 1437–1446. PMLR, 2018. 2
- [22] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatin, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. 1

- [23] Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning. In *International Conference on Learning Representations*, 2022. 2
- [24] Jörg K.H. Franke, Gregor Koehler, André Biedenkapp, and Frank Hutter. Sample-efficient automated deep reinforcement learning. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 5
- [25] Anna Goldie, Azalia Mirhoseini, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, et al. Addendum: A graph placement methodology for fast chip design. *Nature*, pages 1–2, 2024. 1
- [26] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 22
- [27] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *Proceedings of the International Conference on Machine Learning*, pages 2681–2691, 2019. 12
- [28] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 1, 2
- [29] Chloe Ching-Yun Hsu, Celestine Mendler-Dünnner, and Moritz Hardt. Revisiting design choices in proximal policy optimization. *arXiv preprint arXiv:2009.10897*, 2020.
- [30] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*, 2022. 1, 2, 5
- [31] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. 1
- [32] Arman Iranfar, Marina Zapater, and David Atienza. Multiagent reinforcement learning for hyperparameter optimization of convolutional neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(4):1034–1047, 2021. 2
- [33] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. 2, 3, 5, 12
- [34] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019. 2, 3
- [35] Jiantong Jiang, Zeyi Wen, Atif Mansoor, and Ajmal Mian. Efficient hyperparameter optimization with adaptive fidelity identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26181–26190, 2024. 2
- [36] Hadi S Jomaa, Josif Grabocka, and Lars Schmidt-Thieme. Hyp-rl: Hyperparameter optimization by reinforcement learning. *arXiv preprint arXiv:1906.11527*, 2019. 2
- [37] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 3
- [38] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. <https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail>, 2018. 13, 14
- [39] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Roshtamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. 1, 2, 5, 12
- [40] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022. 5, 12
- [41] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024. 1
- [42] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015. 2
- [43] Daniel J Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, 2023. 1
- [44] Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2024. 1
- [45] Paul Micaelli and Amos J Storkey. Gradient-based hyperparameter optimization over long horizons. *Advances in Neural Information Processing Systems*, 34:10798–10809, 2021. 2
- [46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fiedjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 1
- [47] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. 1

- [48] Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. Provably efficient online hyperparameter optimization with population-based bandits. *Advances in neural information processing systems*, 33:17200–17211, 2020. 2, 5, 12
- [49] Jack Parker-Holder, Vu Nguyen, Shaan Desai, and Stephen J Roberts. Tuning mixed input hyperparameters on the fly for efficient population based autorl. *Advances in Neural Information Processing Systems*, 34:15513–15528, 2021. 5, 12, 14
- [50] Supratik Paul, Vitaly Kurin, and Shimon Whiteson. Fast efficient hyperparameter tuning for policy gradient methods. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 3, 5
- [51] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. 1
- [52] Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5402–5415, 2021. 3
- [53] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *Proceedings of the International Conference on Learning Representations*, 2015. 12
- [54] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5, 12
- [55] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 1
- [56] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012. 1, 2
- [57] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. 22
- [58] Mateusz Tejer and Rafał Szczepański. On the importance of hyperparameters tuning for model-free reinforcement learning algorithms. In *2024 12th International Conference on Control, Mechatronics and Automation (ICCA)*, pages 78–82. IEEE, 2024. 2
- [59] A Helen Victoria and Ganesh Maragatham. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 12(1):217–223, 2021. 2
- [60] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 1
- [61] Xingchen Wan, Cong Lu, Jack Parker-Holder, Philip J Ball, Vu Nguyen, Binxin Ru, and Michael Osborne. Bayesian generational population-based training. In *International conference on automated machine learning*, pages 14–1. PMLR, 2022. 2
- [62] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019. 2
- [63] Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2402–2413, 2018. 2
- [64] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020. 1
- [65] Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Automatic intrinsic reward shaping for exploration in deep reinforcement learning. In *International Conference on Machine Learning*, pages 40531–40554. PMLR, 2023. 3
- [66] Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Adaptive data exploitation in deep reinforcement learning. *arXiv preprint arXiv:2501.12620*, 2025. 3, 5
- [67] Mingqi Yuan, Zequn Zhang, Yang Xu, Shihao Luo, Bo Li, Xin Jin, and Wenjun Zeng. Rllte: Long-term evolution project of reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. 1
- [68] Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. *Advances in neural information processing systems*, 33:20913–20924, 2020. 2, 3
- [69] Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 4015–4023. PMLR, 2021. 2