Simple Policy Optimization

Anonymous Authors[1]

Abstract

PPO (Proximal Policy Optimization) algorithm has demonstrated excellent performance in many fields, and it is considered as a simple version of TRPO (Trust Region Policy Optimization) algorithm. However, the ratio clipping operation in PPO may not always effectively enforce the trust region constraints, this can be a potential factor affecting the stability of the algorithm. In this paper, we propose Simple Policy Optimization (SPO) algorithm, which introduces a novel clipping method for KL divergence between the old and current policies. Extensive experimental results in Atari 2600 environments indicate that, compared to the mainstream variants of PPO, SPO achieves better sample efficiency, extremely low KL divergence, and higher policy entropy, and is robust to the increase in network depth or complexity. More importantly, SPO maintains the simplicity of an unconstrained first-order algorithm. Code is available at https://github.com/MyRepositories-hub/Simple-Policy-Optimization.

1 Introduction

Policy based model-free reinforcement learning algorithms have been widely applied in various fields, and have achieved performance surpassing humans including board games (Silver et al., 2016; 2017; 2018), video games (Berner et al., 2019; Vinyals et al., 2019; Ye et al., 2020), autonomous driving (Kiran et al., 2021; Teng et al., 2023) and intelligent control (Chatzilygeroudis et al., 2019; Yu & Wang, 2022; Wen et al., 2022). Although there are two main learning paradigms in deep reinforcement learning: value-based and policy-based, impressive algorithms such as PPO (Schulman et al., 2017) and A3C (Mnih et al., 2016) rely on the Actor-Critic (AC) framework (Konda & Tsitsiklis, 1999), which is a synthesis of the two paradigms.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

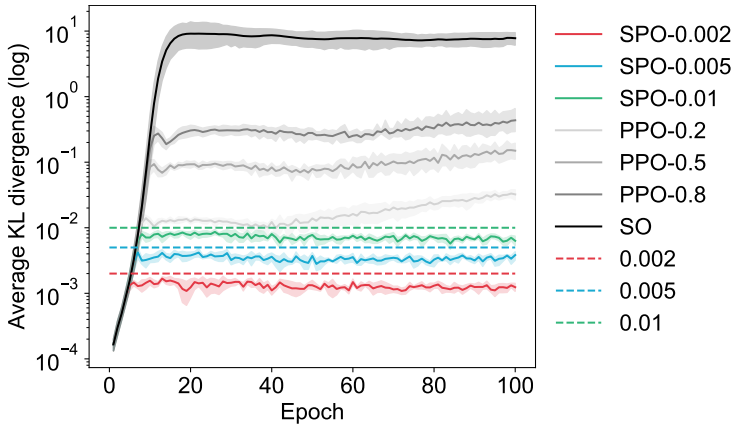


Figure 1. We over-optimize the objective in the Amidar-v5 environment for 100 epochs and select five random seeds for experiments. SPO-x represents the SPO algorithm with the maximum KL divergence set to x, PPO-x represents the PPO algorithm with the ratio clipping parameter set to x, and SO represents directly optimizing the surrogate objective.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm implicitly limits the difference between the old policy and the current policy through the ratio clipping operation, which is reflected in the Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a) algorithm as a hard constraint on their KL divergence expectation. Ratio clipping avoids solving the expensive KL divergence constraint problem, so the PPO algorithm can also be well extended to large-scale reinforcement learning environments (Berner et al., 2019; Ye et al., 2020). Prior works have discussed how ratio clipping can effectively implement trust region constraints. For example, even if the probability ratio is bounded, the corresponding KL divergence is not necessarily bounded (Wang et al., 2020). The performance improvement of PPO algorithm is likely to mainly depend on "code-level optimizations" (Engstrom et al., 2020). There are also some works that design novel probability ratio clipping methods or objective functions (Queeney et al., 2021; Garg et al., 2021; Markowitz & Staley, 2023; Chen et al., 2018), or alternate between a standard RL phase and a distillation phase during the training stage to improve sample efficiency of PPO (Cobbe et al., 2021; Wang et al., 2023).

There are many ways to avoid excessive KL divergence,

which gives birth to many variants of PPO. For example, KL divergence can be added as a penalty term to the surrogate objective and introduce adaptive penalty coefficient (Schulman et al., 2017). During the training process of PPO, a KL threshold is set, and an early stopping strategy is adopted when the KL divergence reaches the threshold (Achiam, 2018). Alternatively, an early stopping strategy can be implemented based on the deviation degree of probability ratio instead of KL divergence (Sun et al., 2022).

However, the above works may not have fundamentally changed the characteristics of the PPO algorithm (we acknowledge that the surrogate objective with ratio clipping remains the preferred choice in most cases), nor have they effectively constrained the KL divergence between the old and current policies. We propose Simple Policy Optimization (SPO) algorithm to compensate for these gaps, SPO does not even require the clipping of probability ratio, instead of a novel clipping method for the KL divergence between the old and current policies. Using SPO, we highlight three important observations about it:

- It can effectively limit the average KL divergence between the current policy and the old policy in almost all environments, while the PPO algorithm inevitably results in a high value of KL divergence between the old and current policies.
- It requires the calculation of the KL divergence between the old and current policies, which does not increase much computational overhead, and still, it maintains the simplicity of an unconstrained first-order algorithm.
- The aforementioned characteristics of the SPO algorithm are robust to the increase of network depth or complexity. In contrast, the increase in KL divergence during the training process of the PPO algorithm becomes more evident as the network deepens.

Based on the above three points, SPO can achieve better monotonic improvement than PPO, and its first-order property makes it easily scalable to distributed training with many workers, delivering strong performance as well.

2 Background

In this section, we introduce the background knowledge of policy gradients in reinforcement learning, as well as the TRPO and PPO algorithms.

2.1 Policy Gradient

Reinforcement learning is generally defined by a tuple $(\mathcal{S}, \mathcal{A}, r, \mathcal{P}, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state space and action space, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the probability distribution of the state transition function, $\rho_0 : \mathcal{S} \mapsto \mathbb{R}$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor.

For each trajectory

$$\tau = (s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T), \tag{1}$$

the objective function can be written as

$$J(\theta) = \mathbb{E}_{\tau \sim p(\cdot)} [R(\tau)] = \sum_{\tau} R(\tau) \cdot p(\tau), \tag{2}$$

where $R(\tau) = \sum_{i=1}^{T-1} \gamma^{i-1} r_i$, the policy gradient can be estimated as

$$\nabla J(\theta) \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n - 1} R(\tau^n) \cdot \nabla \log \pi_\theta(a_t^n | s_t^n). \tag{3}$$

$\nabla J(\theta)$ can be estimated through the Monte Carlo (MC) method (Williams, 1992).

2.2 Trust Region Policy Optimization

In TRPO, the following monotonic improvement theorem is given:

Theorem 2.1. *(Schulman et al., 2015a) Let* $\alpha = D_{\mathrm{TV}}^{\max}(\pi, \tilde{\pi})$ *and* $\epsilon = \max_s |\mathbb{E}_{a \sim \tilde{\pi}} [A^\pi(s, a)]|$*, the following bound holds:*

$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} \cdot \alpha^2,$$

where L_π *matches* η *to first order if the policy* π *is a differentiable function of a parameter vector (Kakade & Langford, 2002).*

Noting the relationship between total variation divergence and the KL divergence $D_{\mathrm{TV}}(\pi, \tilde{\pi})^2 \leq D_{\mathrm{KL}}(\pi, \tilde{\pi})$, we have the following corollary.

Corollary 2.2. *Let* $D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{\mathrm{KL}}(\pi, \tilde{\pi})$*, the following bound holds:*

$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} \cdot D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}).$$

Through a heuristic approximation, the maximum KL divergence $D_{\mathrm{KL}}^{\max}$ in the Corollary 2.2 is approximated as the average KL divergence and used as a constraint to obtain the formalization of TRPO algorithm, that is,

$$\begin{aligned} \max_\theta \ & \mathbb{E}_{(s_t, a_t) \sim \pi_{\theta_{\mathrm{old}}}} \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t) \right] \\ \text{s.t.} \ & \mathbb{E} \left[D_{\mathrm{KL}}(\pi_{\theta_{\mathrm{old}}}, \pi_\theta) \right] \leq \delta. \end{aligned} \tag{4}$$

This constrained optimization problem can be solved using the conjugate gradient algorithm followed by a line search.

Algorithm 1 Simple Policy Optimization (SPO)

Input: Policy network parameters θ; value network parameters w; threshold of KL divergence $d_{\max}$
Output: Optimal policy network parameters θ^*
while not converged **do**
  # Data collection
  Collect data $X = \{(s_i, a_i, r_i)\}_{i=1}^{N}$ using the current policy network π_θ
  for each training epoch **do**
    # Estimate advantage function
    Use GAE (Schulman et al., 2015b) technique to estimate the advantage $\hat{A}(s_t, a_t)$
    # Calculate KL divergence
    For each data (s_t, a_t, r_t) calculate $d \leftarrow \sum_{a\in\mathcal{A}} \pi_{\theta_{\text{old}}}(a|s_t) \cdot \log \frac{\pi_{\theta_{\text{old}}}(a|s_t)}{\pi_\theta(a|s_t)}$ and $d_{\text{clip}} \leftarrow \text{clip}(d, 0, d_{\max})$
    # Calculate policy loss l_p, value loss l_v and policy entropy l_e
    Policy loss

$$l_p \leftarrow \begin{cases} -\dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t), & \theta = \theta_{\text{old}} \\ -\dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t) \cdot \left\{ \dfrac{d_{\text{clip}}}{d} + \text{sign}\left[\hat{A}(s_t, a_t)\right] - 1 \right\} \cdot \text{sign}\left[\hat{A}(s_t, a_t)\right], & \theta \neq \theta_{\text{old}} \end{cases}$$

    and total loss $l \leftarrow l_p + c_1 \cdot l_v - c_2 \cdot l_e$
    # Update parameters
    Update parameters θ and w through backpropagation
  end for
  Current policy network parameters $\theta_{\text{old}} \leftarrow \theta$
end while

2.3 Proximal Policy Optimization

PPO is an algorithm that attains the data efficiency and reliable performance of TRPO, while using only first-order optimization. PPO algorithm has two main variants, one of them is called PPO-PENALTY, that is,

$$\max_\theta \mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \left\{ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t) - \beta \left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}, \pi_\theta) \right] \right\}, \tag{5}$$

where β is adaptive KL penalty coefficient. Another one is called PPO-CLIP, which can be formalized as

$$\max_\theta \mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \left\{ \min \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t), \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \cdot \hat{A}(s_t, a_t) \right] \right\}. \tag{6}$$

PPO-CLIP has been widely adopted by the academic community due to its simple form and reliable performance.

3 Method

In TRPO, trust region are imposed by limiting the expectation of the KL divergence between the old and current policies. Here, we denote the KL divergence as

$$\begin{aligned} d &= D_{\text{KL}}\left[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right] \\ &= \sum_{a\in\mathcal{A}} \pi_{\theta_{\text{old}}}(a|s_t) \cdot \log \frac{\pi_{\theta_{\text{old}}}(a|s_t)}{\pi_\theta(a|s_t)}, \end{aligned} \tag{7}$$

and the surrogate objective

$$\mathfrak{O}^\theta_{\theta_{\text{old}}}(s_t, a_t) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t). \tag{8}$$

According to the Corollary 2.2, we hope to optimize the surrogate objective $\mathfrak{O}^\theta_{\theta_{\text{old}}}(s_t, a_t)$ with the KL divergence d as small as possible, as it optimizes the monotonic improvement lower bound in policy iterations. This minimax problem seems to be equivalent to the following form:

$$\max_\theta \mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \left[\frac{\mathfrak{O}^\theta_{\theta_{\text{old}}}(s_t, a_t)}{d} \right]. \tag{9}$$

However, note that if we simply using the above form as the objective for the policy network, this can lead to significant problems. Now Assume that the estimated value of the advantage $\hat{A}(s_t, a_t)$ is positive, the policy network would strive to minimize the KL divergence in order to maximize the objective. This tendency would result in the network

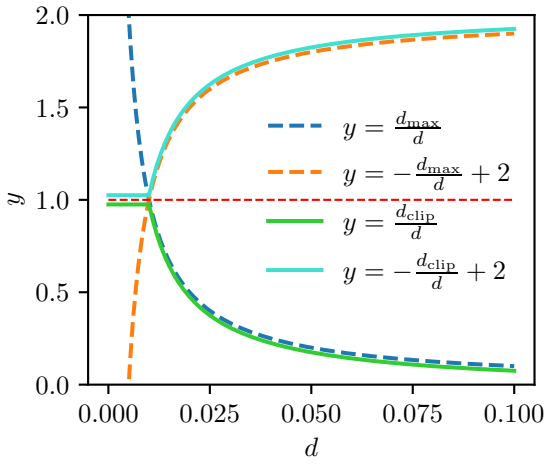


Figure 2. Function curves of $(d_{\text{clip}}/d + \sigma - 1) \cdot \sigma$ when $d_{\max}$ is set to 0.01.

preferring to make only minor adjustments to the parameters, thereby hindering its ability to learn effectively from historical data.

To address this issue, we aim to guide the policy network to avoid situations where the KL divergence d becomes excessively small. For example, we artificially set a threshold $d_{\max}$. When $d \leq d_{\max}$, the objective is the surrogate objective $\mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t)$. Otherwise, if $d > d_{\max}$, then the objective becomes the ratio form of (9). The formalization will be

$$\mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \left[\mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \frac{d_{\text{clip}}}{d} \right], \tag{10}$$

where $d_{\text{clip}} = \text{clip}(d, 0, d_{\max})$, $d_{\max}$ is a hyperparameter that constrains the upper bound of KL divergence. As we expected:

$$\mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \frac{d_{\text{clip}}}{d} = \begin{cases} \mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t), & d \leq d_{\max}; \\ \mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \frac{d_{\max}}{d}, & d > d_{\max}. \end{cases} \tag{11}$$

Objective (10) only considers the case where the estimated value of the advantage $\hat{A}(s_t, a_t)$ is positive. However, when $\hat{A}(s_t, a_t)$ is negative and d exceeds the threshold, we need to increase the absolute value of objective as a penalty instead of decreasing it. Therefore, by combining both cases, we formalize the objective:

$$\mathop{\mathbb{E}}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}} \left[\mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \left(\frac{d_{\text{clip}}}{d} + \sigma - 1 \right) \cdot \sigma \right], \tag{12}$$

where

$$\sigma = \text{sign}\left[\hat{A}(s_t, a_t)\right] = \begin{cases} +1, & \hat{A}(s_t, a_t) > 0; \\ 0, & \hat{A}(s_t, a_t) = 0; \\ -1, & \hat{A}(s_t, a_t) < 0. \end{cases} \tag{13}$$

We now refer to objective (12) as $J(\theta)$. Obviously, we have

$$J(\theta) = \begin{cases} \mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \frac{d_{\text{clip}}}{d}, & \hat{A}(s_t, a_t) > 0; \\ \mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \left(-\frac{d_{\text{clip}}}{d} + 2 \right), & \hat{A}(s_t, a_t) < 0. \end{cases} \tag{14}$$

Furthermore, if $d \leq d_{\max}$, we have $(d_{\text{clip}}/d + \sigma - 1) \cdot \sigma \equiv 1$, which means that the KL divergence satisfies the trust region constraints, and the goal $J(\theta)$ will be the surrogate objective $\mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t)$. Otherwise,

$$J(\theta) = \begin{cases} \mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \frac{d_{\max}}{d}, & \hat{A}(s_t, a_t) > 0; \\ \mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t) \cdot \left(-\frac{d_{\max}}{d} + 2 \right), & \hat{A}(s_t, a_t) < 0. \end{cases} \tag{15}$$

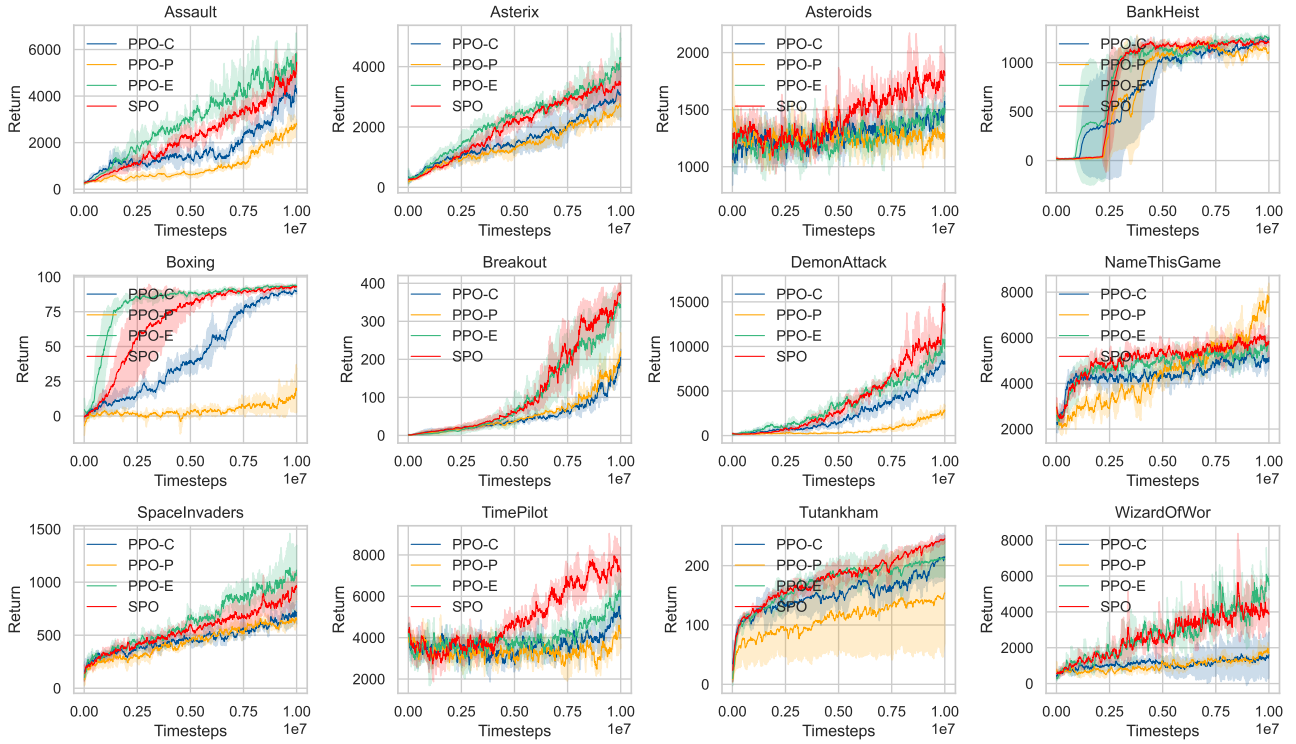
We will provide intuition about the objective (12), suppose that $d_{\max} = 0.01$, and the function curves is shown in Figure 2. If $\hat{A}(s_t, a_t) > 0$ and $d > d_{\max}$, the objective $J(\theta)$ will decrease rapidly (see curve d_{clip}/d), this can force the KL divergence to satisfy the trust region constraints. Otherwise, when $\hat{A}(s_t, a_t) < 0$ and $d > d_{\max}$, the function curve $-d_{\text{clip}}/d + 2$ and d_{clip}/d are symmetric about the line $y = 1$, and $J(\theta)$ will also decrease. Therefore, to maximize the objective $J(\theta)$, SPO will secretly optimize the surrogate objective $\mathfrak{O}^{\theta}_{\theta_{\text{old}}}(s_t, a_t)$ within the trust region $d \leq d_{\max}$ as much as possible, but it is still an unconstrained first-order algorithm.

The pseudo-code of SPO is shown in Algorithm 1. When $\theta = \theta_{\text{old}}$ (this means that the policy network has not been updated yet), we use the original surrogate objective to update the policy network parameters θ. Once the update is performed, the parameters of the current policy network change, then we optimize the objective (12).

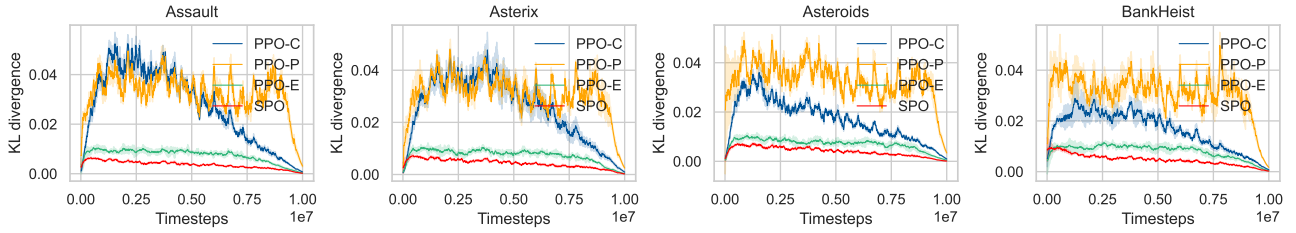
Considering that the hyperparameter $d_{\max}$ can be set manually and even dynamically changed at any moment during the training process, SPO can benefit significantly from the Corollary 2.2. In summary, the characteristics mentioned above enable SPO to potentially achieve better monotonic policy improvement than PPO, and its performance depends on the value of the hyperparameter $d_{\max}$, which corresponds to the degree of conservativeness in policy iterations.

4 Experiments

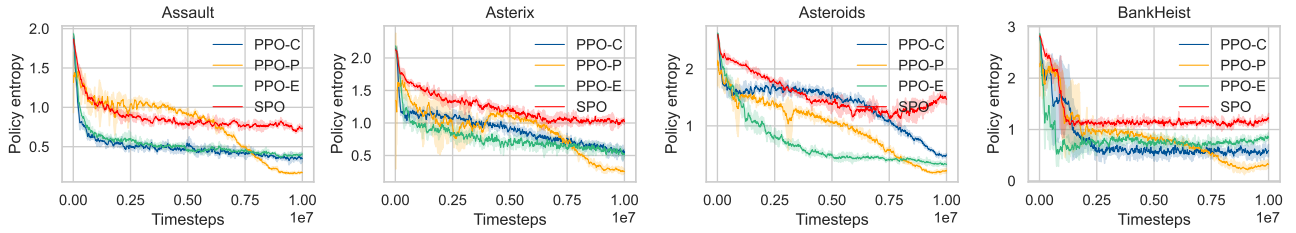
We report results on the Atari 2600 environment (Bellemare et al., 2013; Machado et al., 2018). The Atari 2600 environment is a popular benchmark in the field of reinforcement learning. It consists of a set of video game environments designed for the Atari 2600 gaming console. These environments serve as testing grounds for developing and evaluating reinforcement learning algorithms and



(a) Training curves of return.



(b) Training curves of average KL divergence.



(c) Training curves of policy entropy.

Figure 3. Training curves of different PPO variants and SPO algorithm in Atari 2600 environments. Among them, PPO-C represents PPO-CLIP, PPO-P represents PPO-PENALTY, and PPO-E represents PPO-CLIP with an early stopping strategy.

have been widely used because of its diverse set of games, ranging from simple classics like Pong to more complex titles like SpaceInvaders and Breakout. This diversity allows researchers to assess the generalization capabilities of reinforcement learning algorithms across different tasks and complexities. We preprocessed the Atari 2600 environment, for example, the input to the network is the stacking of the last four frames resized to $84 \times 84 \times 4$, reward clipping and so on (Mnih et al., 2013; Huang et al., 2022). In all experiments, we use the hyperparameters provided in the Appendix A unless otherwise specified. For each environment in Atari 2600, we compute and visualize the standard deviation of the algorithm's performance across 3 separate runs with different random seeds.

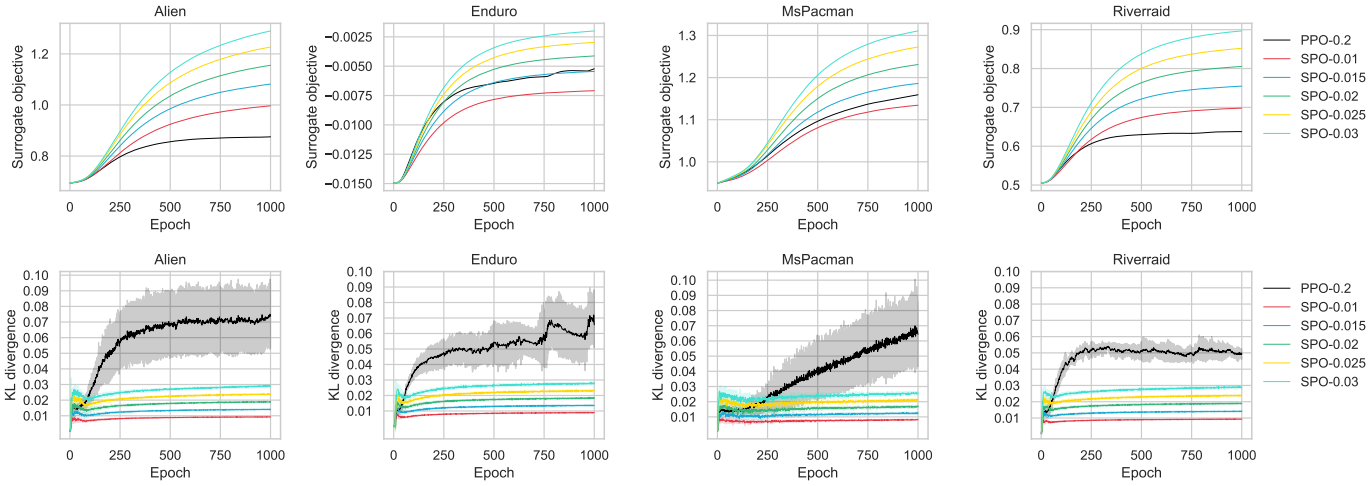


Figure 4. We over-optimize the objectives of PPO and SPO for 1000 epochs in the early stages of training, where $d_{\max}$ is set to 0.01, 0.015, 0.02, 0.025 and 0.03.

Our experiments are divided into four parts. The first part compares the performance of several PPO variants and the SPO algorithm in the Atari 2600 environment, as well as the changes in KL divergence. In the second part, we increase the number of update epochs from 8 to 1000 in a certain training stage to over-optimize the respective objectives of PPO and SPO, and observe their optimization of surrogate objectives and the growth trend of KL divergence. In the third part, we try to increase the depth of the neural network and observe how the complexity of the network structure affects the performance of PPO and SPO in the training process. Finally, we conducted a sensitivity analysis on the hyperparameter $d_{\max}$ of the SPO algorithm.

4.1 Comparison to PPOs

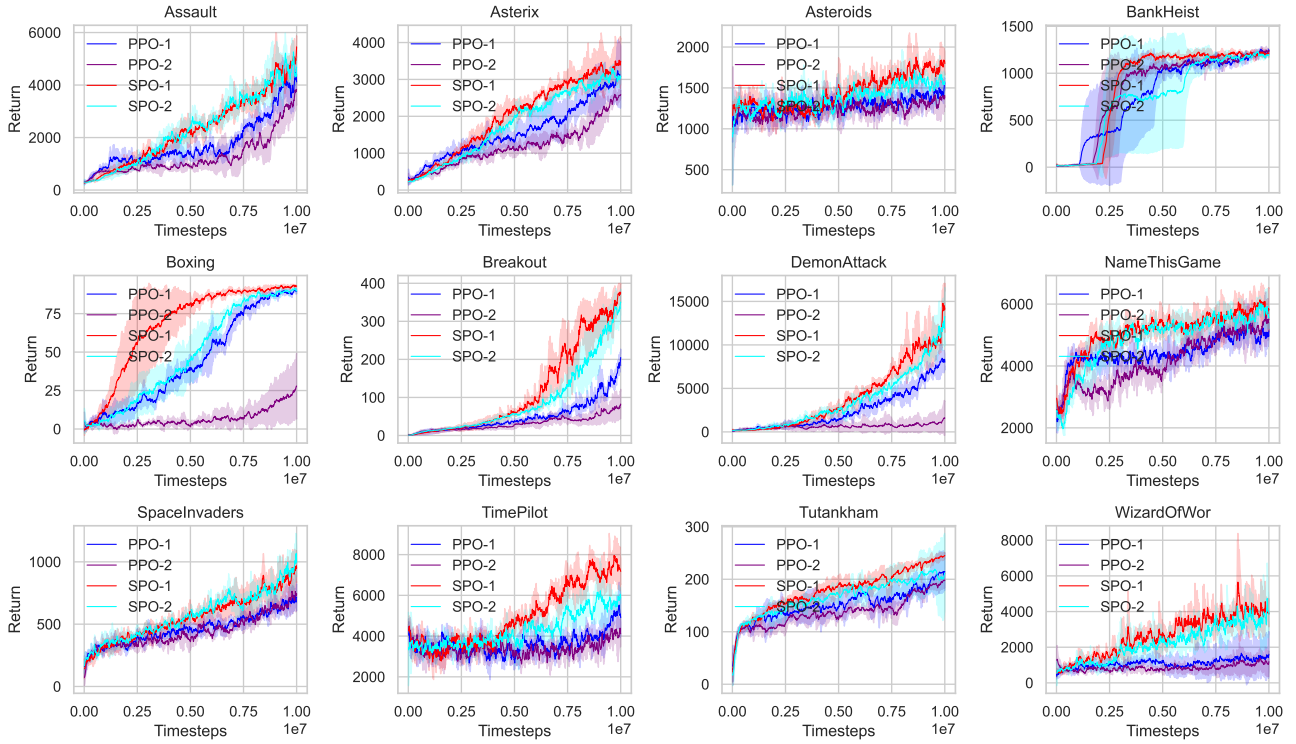
For the sake of experimental fairness, the policy network and value network structures of both the SPO and PPO algorithms are entirely identical, see Appendix C. They share a common convolutional layer designed to compress high-dimensional image inputs into a low-dimensional representation of 512 dimensions. Subsequently, a fully connected layer is connected separately to generate outputs for the policy network (with the output dimensionality corresponding to the action space) and the value network (with a 1-dimensional output).

Partial experimental results are shown in Figure 3, with more detailed results provided in the Appendix B.4. Here, PPO-PENALTY (PPO-P) and PPO-CLIP with an early stopping strategy (PPO-E) are two main variants introduced to limit the excessive increase of KL divergence of PPO during the training process. We maintain consistency in the corresponding KL divergence thresholds for these two variants and the threshold $d_{\max}$ for SPO, all set to 0.02. It can be seen that SPO nearly achieves significantly better sample efficiency than other PPO variants in most environments, ranking second only to SPO-E in a few cases, while maintaining the lowest KL divergence throughout the entire training process. In contrast, both PPO-C and PPO-P consistently exhibit significantly high KL divergence. However, PPO-E is also able to effectively limit KL divergence, trailing only slightly behind SPO.

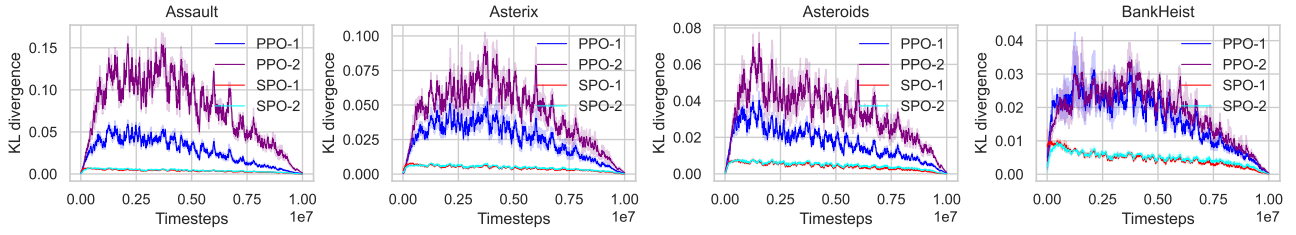
We also notice that SPO exhibits the highest policy entropy during training, which indicates that SPO has a better exploration capability in the environment. We speculate that this may be a by-product of limiting the KL divergence. In fact, we conducted a sensitivity analysis on the hyperparameter $d_{\max}$ in Subsection 4.4, and we found that a lower value of $d_{\max}$ leads to a higher policy entropy.

4.2 Over Optimization

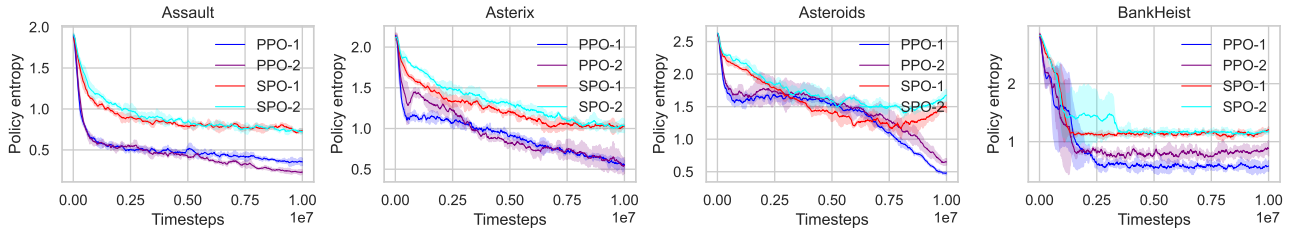
To further evaluate whether SPO can satisfy trust region constraints under extreme conditions, we select four environments and over-optimize the objectives of PPO and SPO with different $d_{\max}$ for 1000 epochs in the early stages of training, results are shown in Figure 4. Where SPO-x represents the SPO algorithm with the maximum KL divergence $d_{\max}$ set to x, PPO-x represents the PPO algorithm with the ratio clipping parameter ϵ set to x. We can see that even in the case of over-optimization, SPO still has a very stable KL divergence, which remains below the threshold, and the surrogate objective gradually increases as the constraints become weaker ($d_{\max}$ increases). In contrast, PPO leads to excessively high average KL divergence and has relatively poor optimization performance for surrogate objective. Therefore, to a certain extent, this indicates that SPO optimizes the surrogate objective within the trust region as



(a) Training curves of return.



(b) Training curves of average KL divergence.



(c) Training curves of policy entropy.

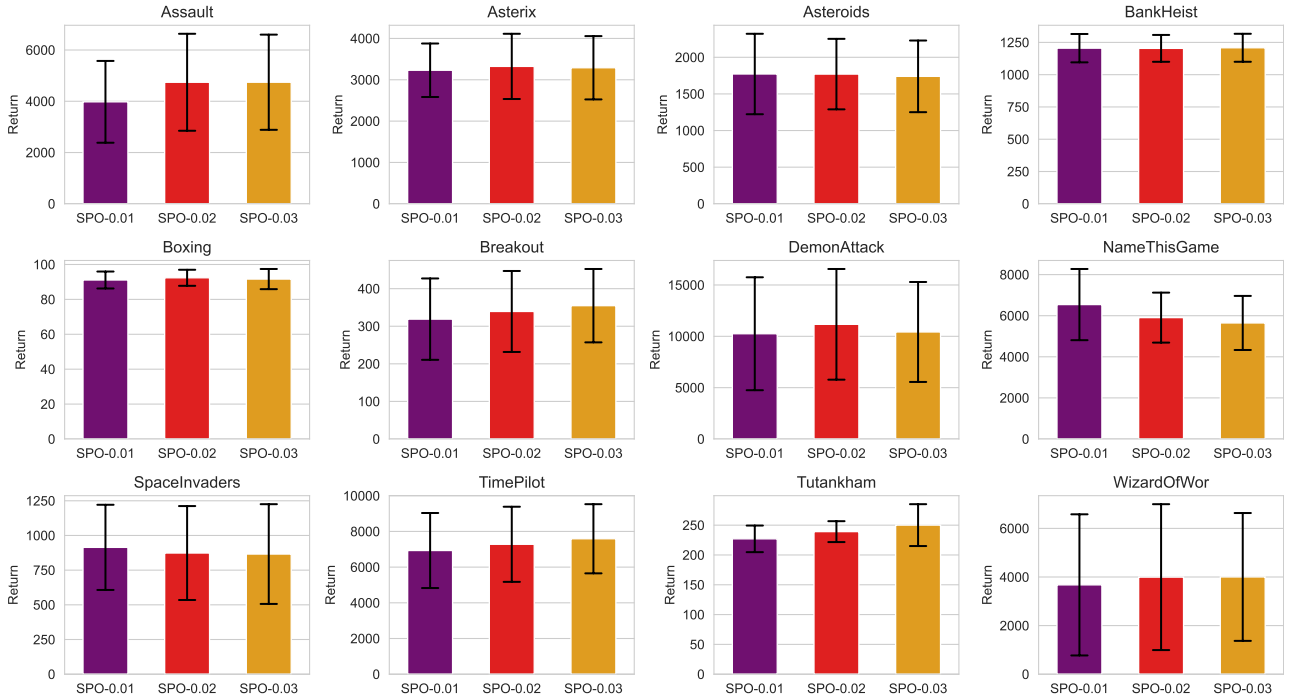
Figure 5. Training curves of PPO and SPO algorithms using networks with different depths in Atari 2600 environments, where PPO-1 and SPO-1 represent the use of shallow networks, while PPO-2 and SPO-2 represent the use of deeper networks.

much as possible, and performs better than PPO.

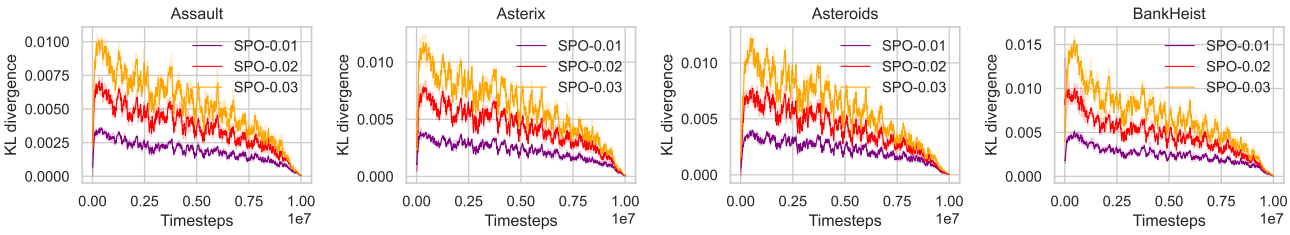
4.3 Increasing Network Complexity

We are interested in understanding how the depth or complexity of the neural networks[1] affects the performance of both PPO and SPO algorithms in the training process, as well as their KL divergence. This subsection aims to answer the following questions: When the network structure becomes more complex, will the ability of the SPO algorithm to limit KL divergence disappear? Will the phenomenon of PPO's inability to limit KL divergence become more evident?

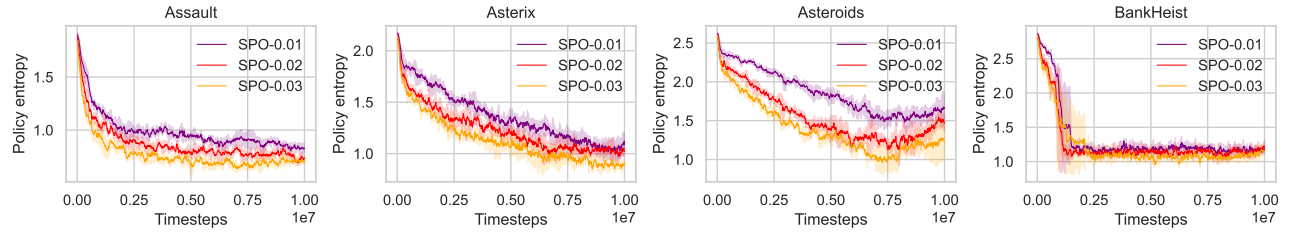
[1]Two neural network structures are shown in the appendix C.



(a) Error bar charts of return of SPO in Atari 2600 environments during the last 10% training steps, with their $d_{\max}$ set to 0.01, 0.02, and 0.03, respectively.



(b) Training curves of average KL divergence.



(c) Training curves of policy entropy.

Figure 6. Training curves of SPO in Atari 2600 environments, with their corresponding KL divergence thresholds $d_{\max}$ set to 0.01, 0.02, and 0.03, respectively.

We did not make any changes to the hyperparameters or other settings, except for increasing the number of network layers. PPO-1 and SPO-1 represent the use of shallow networks, while PPO-2 and SPO-2 represent the use of deeper networks. The results are shown in Figure 5. As the network becomes deeper, we can see that the KL divergence of PPO during the training process becomes larger than that in the shallow networks, sometimes leading to a sharp decline in its performance. In some environments, PPO almost loses its ability to learn effectively (e.g., Breakout and DemonAttack). In contrast, SPO demonstrates strong robustness to changes in network structure. Furthermore, we observed that the average KL divergence curves of SPO-1 and SPO-2 nearly overlap throughout the entire training process, which once again validates the effectiveness of SPO in limiting KL divergence. Additionally, this suggests that SPO is more

likely to play a pivotal role in complex, large-scale neural network models, such as the Transformers (Vaswani et al., 2017) or DM (Diffusion Models) (Ho et al., 2020).

4.4 Sensitivity Analysis

We conducted a sensitivity analysis on the hyperparameter $d_{\max}$ of SPO, setting it to 0.01, 0.02, and 0.03, respectively. The experimental results are shown in Figure 6.

We can see that the performance of the SPO algorithm is not highly sensitive to the selection of the hyperparameter $d_{\max}$, and the average KL divergence during the training process gradually increases as $d_{\max}$ increases. Notably, as $d_{\max}$ gradually decreases, the policy entropy of the SPO algorithm gradually increases. This result seems to be expected because the KL divergence d satisfies

$$\begin{aligned} d &= \sum_{a\in\mathcal{A}} \pi_{\theta_{\text{old}}}(a|s_t)\cdot\log\frac{\pi_{\theta_{\text{old}}}(a|s_t)}{\pi_{\theta}(a|s_t)}, \\ &= \underbrace{H\left(\pi_{\theta_{\text{old}}}(\cdot|s_t),\pi_{\theta}(\cdot|s_t)\right)}_{\text{Cross Entropy}} - \underbrace{H\left(\pi_{\theta_{\text{old}}}(\cdot|s_t)\right)}_{\text{Policy Entropy}}. \end{aligned} \tag{16}$$

The cross entropy is non-negative, and it will be a good choice for the agent to reduce the KL divergence by maintaining a higher policy entropy in the training process.

We believe that there is a fundamental difference between SPO and PPO-E, the latter of which introduces an early stopping strategy to forcibly limit the KL divergence, as SPO always exhibits a higher policy entropy than PPO-E when they have a similar level of KL divergence.

5 Discussion

We propose Simple Policy Optimization (SPO), a first-order algorithm for policy optimization in reinforcement learning. Experimental results in the Atari 2600 environment demonstrate that compared to the traditional PPO algorithm, SPO significantly improves sample efficiency and effectively limits the average KL divergence between the old and current policies, while exhibiting higher policy entropy. SPO strikes a favorable balance between sample complexity, simplicity and wall-time.

In this work, we do not advocate the use of the surrogate objective with ratio clipping. We show that the ratio clipping method is not effective in limiting the average KL divergence between the old and current policies, and this phenomenon becomes more pronounced as the network deepens. In contrast, SPO achieves higher sample efficiency with lower KL divergence and exhibits robustness to the depth or complexity of the network.

We have noticed that some techniques employ reinforcement learning to allow LLMs (Large Language Models) to learn human preferences (Ouyang et al., 2022; Ziegler et al., 2019; Stiennon et al., 2020; Lee et al., 2023), called RLHF (Reinforcement Learning from Human Feedback). A crucial step in this process is to fine-tune the LLMs using policy optimization algorithms such as PPO. To prevent the "catastrophic forgetting" of LLMs, efforts have been focused on developing new algorithms that minimize the KL divergence with the base reference model during the fine-tuning process (Ziegler et al., 2019; Rafailov et al., 2024; Tunstall et al., 2023). It would be intriguing to apply SPO to this process, as the key difference lies in the fact that the old policy is no longer the policy network at the end of the previous training phase, but instead refers to the base reference policy.

References

Achiam, J. Spinning Up in Deep Reinforcement Learning. 2018.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Chatzilygeroudis, K., Vassiliades, V., Stulp, F., Calinon, S., and Mouret, J.-B. A survey on policy search algorithms for learning robot controllers in a handful of trials. *IEEE Transactions on Robotics*, 36(2):328–347, 2019.

Chen, G., Peng, Y., and Zhang, M. An adaptive clipping approach for proximal policy optimization. *arXiv preprint arXiv:1804.06461*, 2018.

Cobbe, K. W., Hilton, J., Klimov, O., and Schulman, J. Phasic policy gradient. In *International Conference on Machine Learning*, pp. 2020–2027. PMLR, 2021.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

Garg, S., Zhanson, J., Parisotto, E., Prasad, A., Kolter, Z., Lipton, Z., Balakrishnan, S., Salakhutdinov, R., and Ravikumar, P. On proximal policy optimization's heavy-tailed gradients. In *International Conference on Machine Learning*, pp. 3610–3619. PMLR, 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., and Wang, W. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 267–274, 2002.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23 (6):4909–4926, 2021.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T., Bishop, C., Carbune, V., and Rastogi, A. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Markowitz, J. and Staley, E. W. Clipped-objective policy gradients for pessimistic policy optimization. *arXiv preprint arXiv:2311.05846*, 2023.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Queeney, J., Paschalidis, Y., and Cassandras, C. G. Generalized proximal policy optimization with sample reuse. *Advances in Neural Information Processing Systems*, 34: 11909–11919, 2021.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Sun, M., Kurin, V., Liu, G., Devlin, S., Qin, T., Hofmann, K., and Whiteson, S. You may not need ratio clipping in ppo. *arXiv preprint arXiv:2202.00079*, 2022.

Teng, S., Hu, X., Deng, P., Li, B., Li, Y., Ai, Y., Yang, D., Li, L., Xuanyuan, Z., Zhu, F., et al. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Transactions on Intelligent Vehicles*, 2023.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Rasul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier, C., Habib, N., et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Wang, K., Zhou, D., Feng, J., and Mannor, S. Ppg reloaded: An empirical study on what matters in phasic policy gradient. 2023.

Wang, Y., He, H., and Tan, X. Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence*, pp. 113–122. PMLR, 2020.

Wen, B., Lian, W., Bekris, K., and Schaal, S. You only demonstrate once: Category-level manipulation from single visual demonstration. *arXiv preprint arXiv:2201.12716*, 2022.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H., Yang, S., Wu, X., Guo, Q., et al. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6672–6679, 2020.

Yu, C. and Wang, P. Dexterous manipulation for multi-fingered robotic hands with reinforcement learning: a review. *Frontiers in Neurorobotics*, 16:861825, 2022.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Hyperparameters Setting

Table 1. Detailed hyperparameters in Atari 2600 environments.

Algorithm	PPO-CLIP	PPO-PENALTY	PPO-EarlyStop	SPO
Number of actors	8	8	8	8
Horizon	128	128	128	128
Learning rate	2.5×10^{-4}	2.5×10^{-4}	2.5×10^{-4}	2.5×10^{-4}
Learning rate decay	Linear	Linear	Linear	Linear
Optimizer	Adam	Adam	Adam	Adam
Total steps	1×10^7	1×10^7	1×10^7	1×10^7
Batch size	1024	1024	1024	1024
Update epochs	8	8	8	8
Mini-batch size	256	256	256	256
Mini-batches	4	4	4	4
GAE parameter λ	0.95	0.95	0.95	0.95
Discount factor γ	0.99	0.99	0.99	0.99
Clipping parameter ϵ	0.2	\	0.2	\
Value loss coeff. c_1	1	1	1	1
Entropy loss coeff. c_2	0.01	0.01	0.01	0.01
Threshold of KL divergence	\	0.02	0.02	\
Maximum KL divergence $d_{\max}$	\	\	\	0.02

Table 2. Detailed hyperparameters in MuJoCo environments.

Algorithm	PPO	SPO
Number of actors	8	8
Horizon	256	256
Learning rate	3×10^{-4}	3×10^{-4}
Learning rate decay	Linear	Linear
Optimizer	Adam	Adam
Total steps	3×10^6, 1×10^7	3×10^6, 1×10^7
Batch size	2048	2048
Update epochs	10	10
Mini-batch size	512	512
Mini-batches	4	4
GAE parameter λ	0.95	0.95
Discount factor γ	0.99	0.99
Clipping parameter ϵ	0.2	\
Value loss coeff. c_1	0.5	0.5
Entropy loss coeff. c_2	0.0	0.0
Maximum KL divergence $d_{\max}$	\	0.005, 0.01, 0.03

B More Experiment Results

B.1 Training Curves of Return

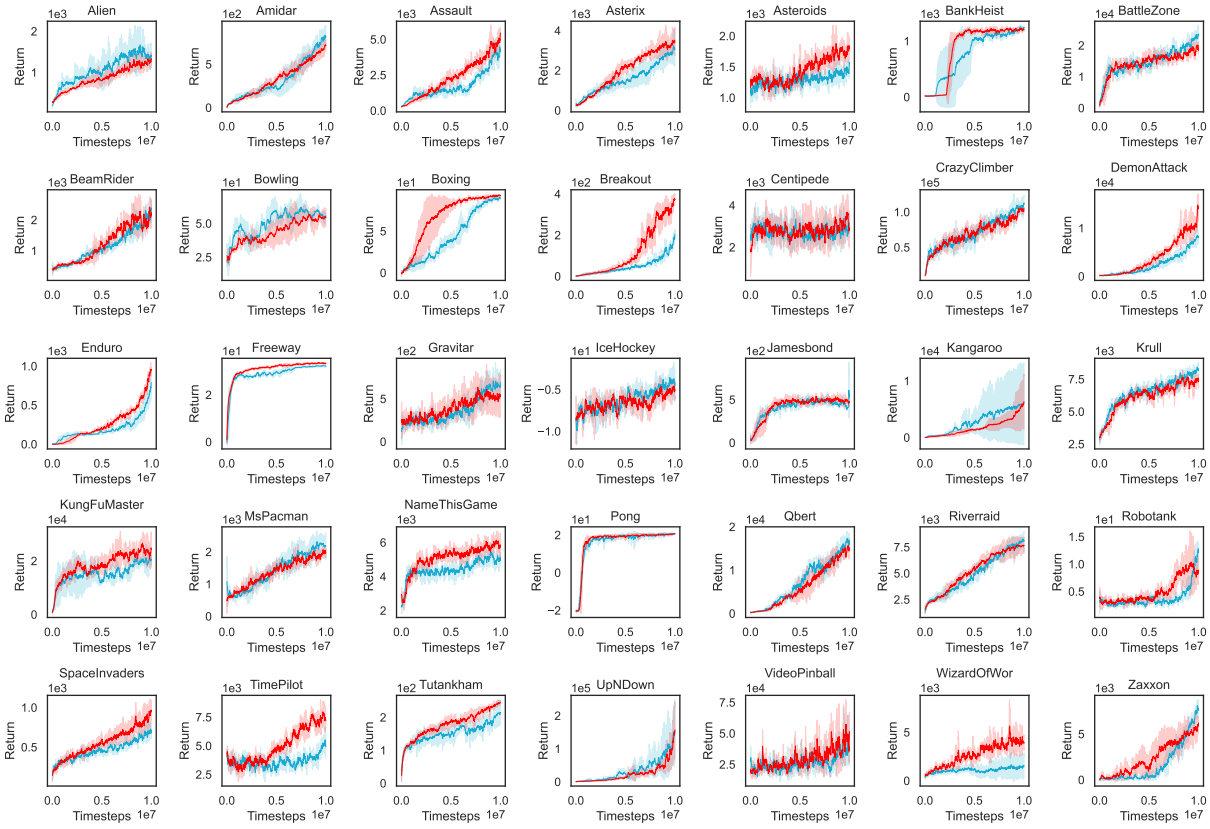


Figure 7. Training curves of return of PPO and SPO in Atari 2600 environments.

B.2 Training Curves of Average KL Divergence

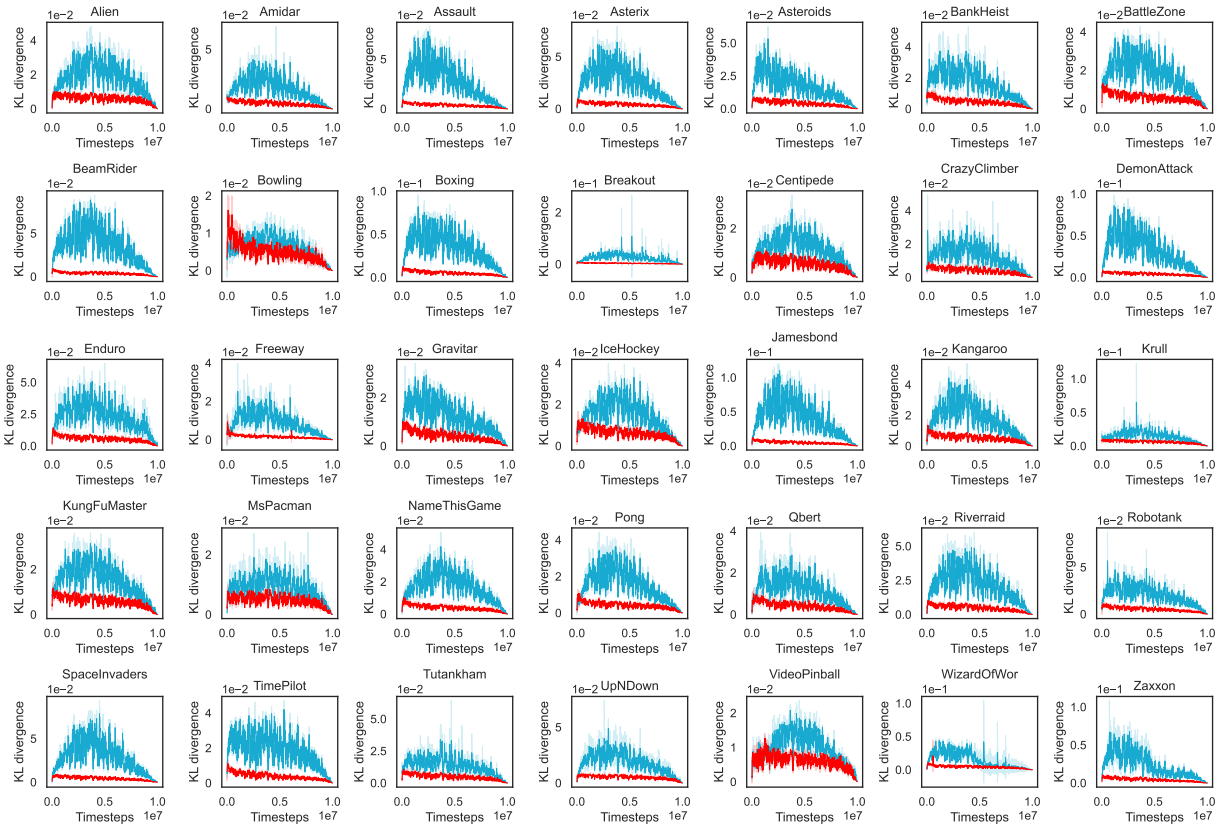


Figure 8. Training curves of average KL divergence in Atari 2600 environments.

B.3 Training Curves of Policy Entropy

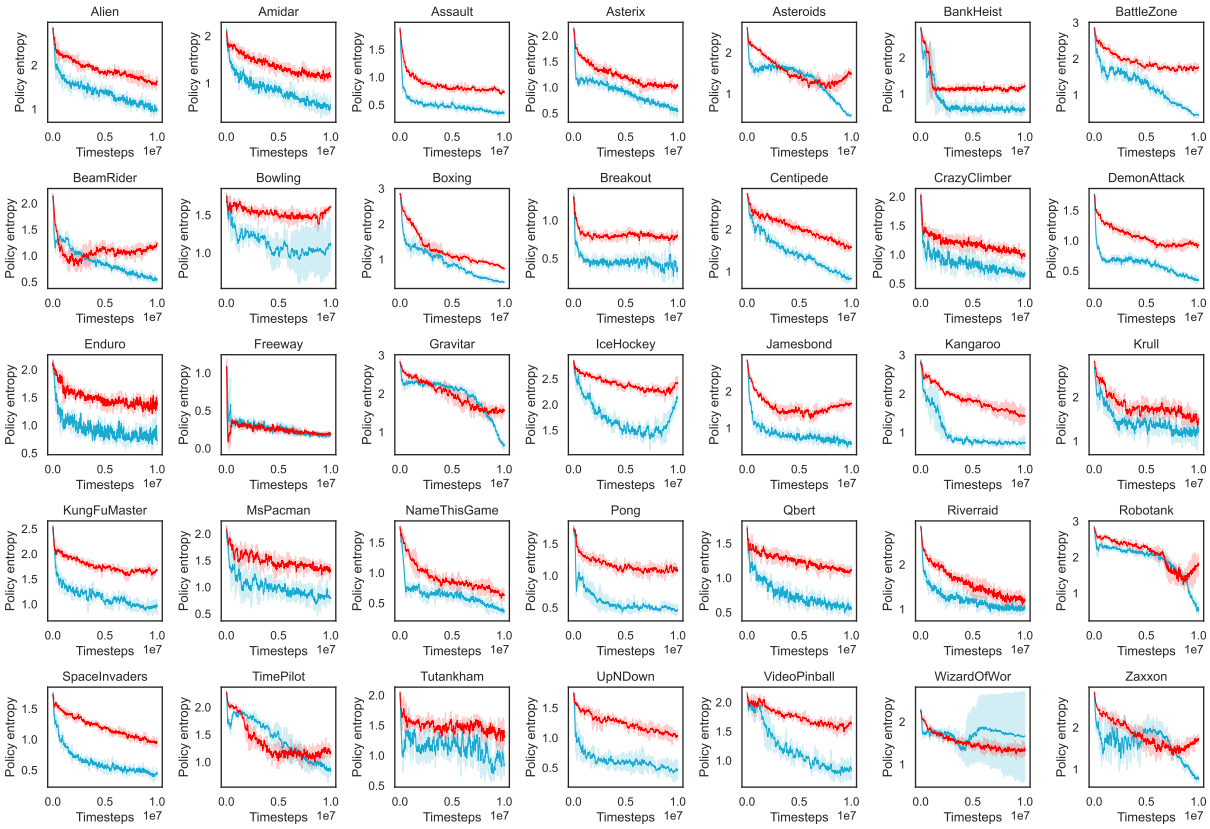
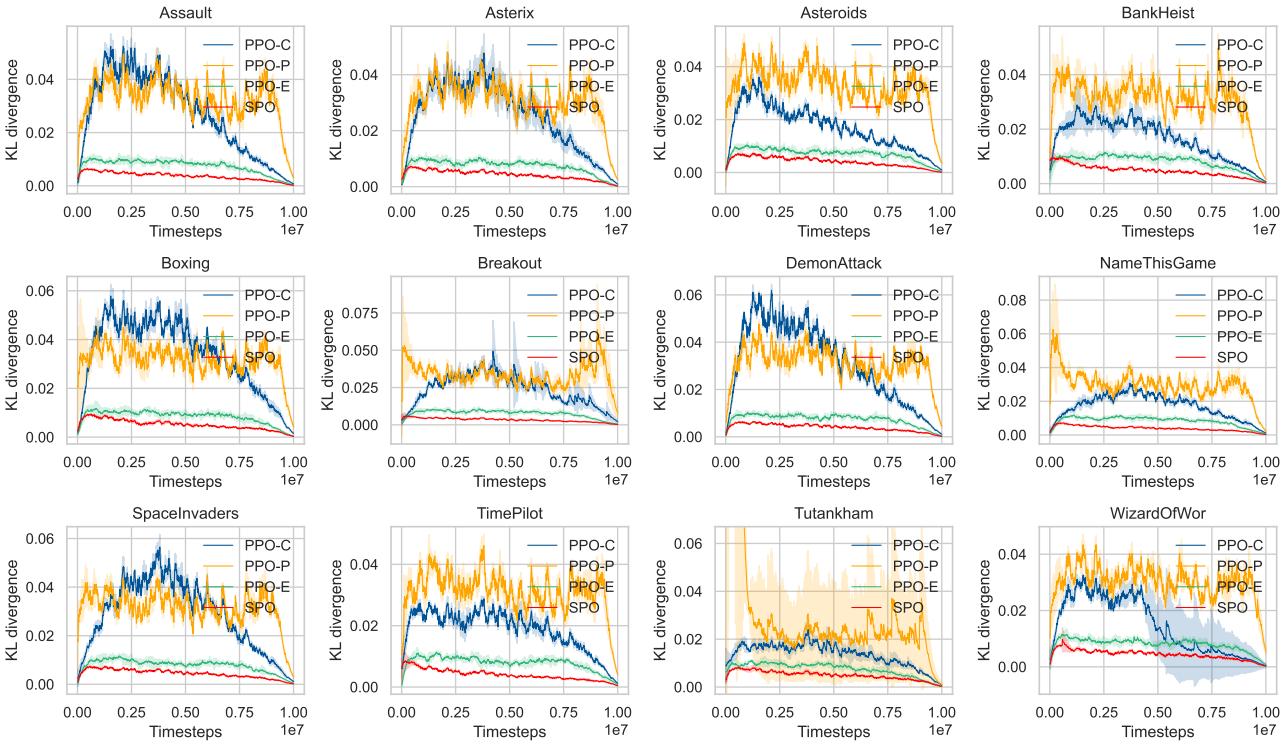
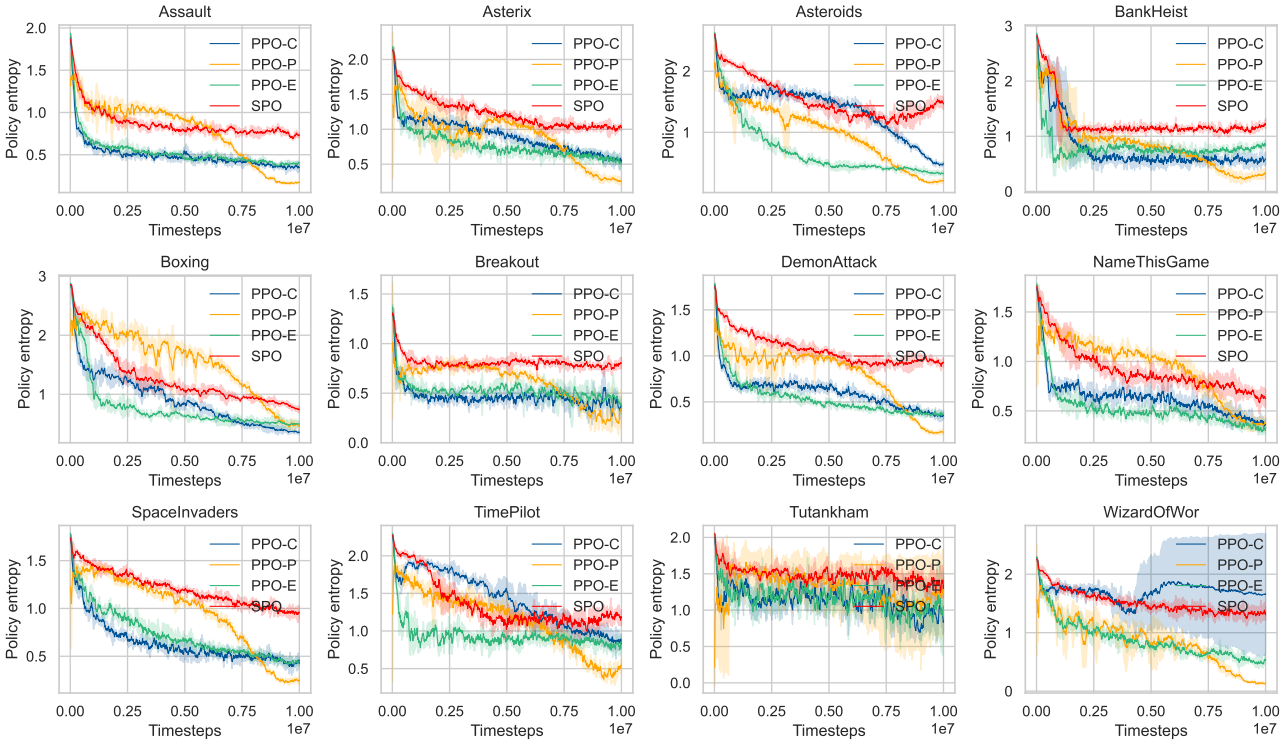


Figure 9. Training curves of policy entropy in Atari 2600 environments.

B.4 Comparison to PPOs



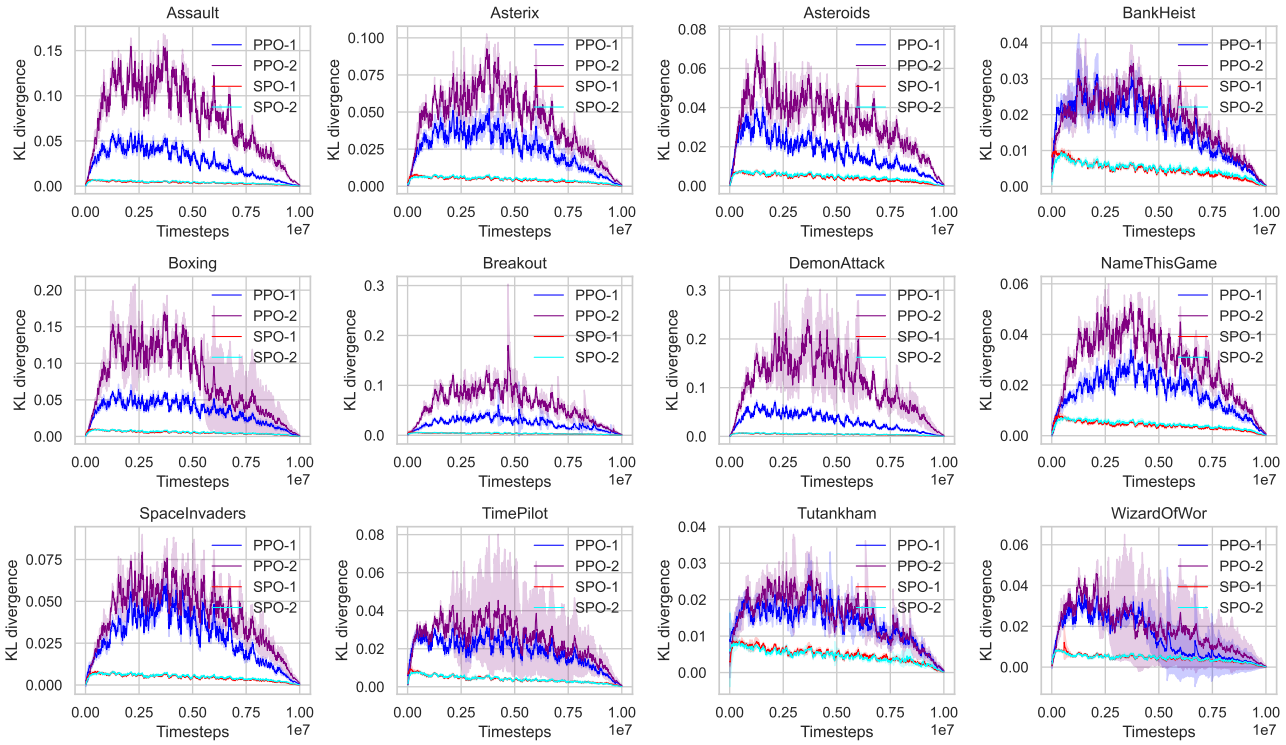
(a) Training curves of average KL divergence.



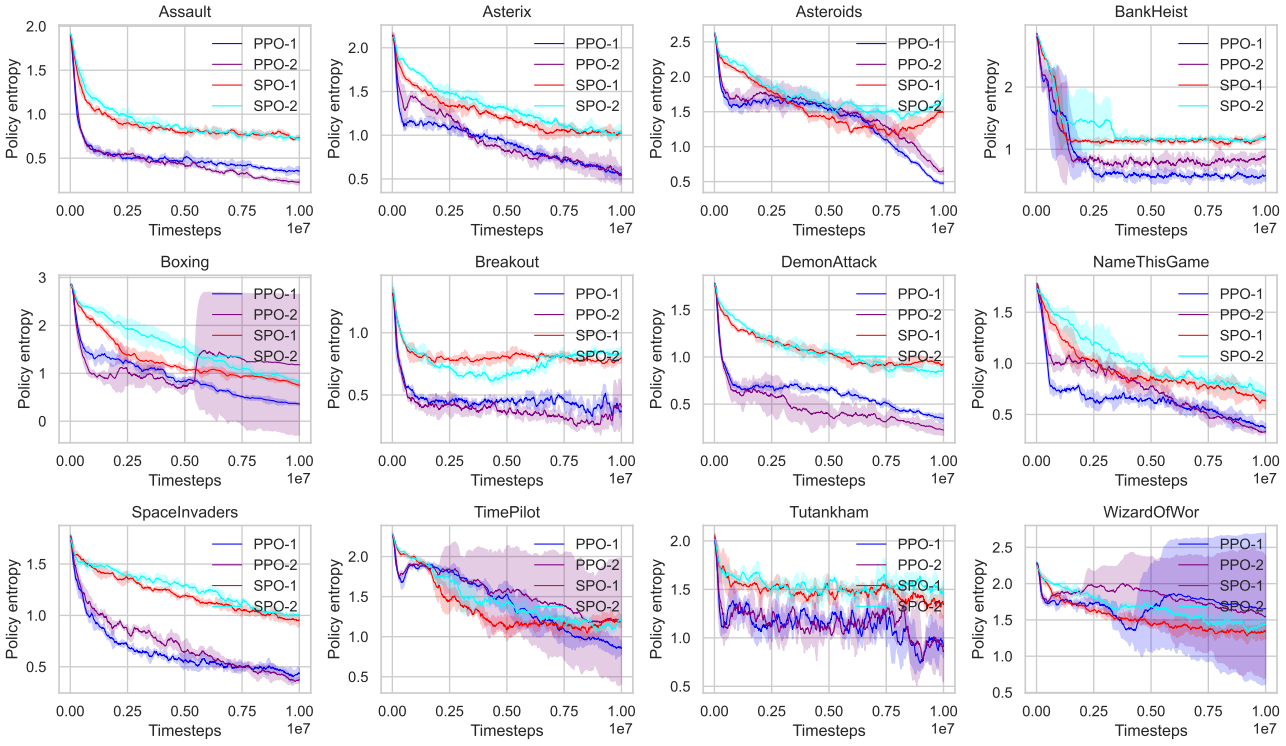
(b) Training curves of policy entropy.

Figure 10. Training curves of average KL divergence and policy entropy in Atari 2600 environments.

B.5 Increasing Network Complexity



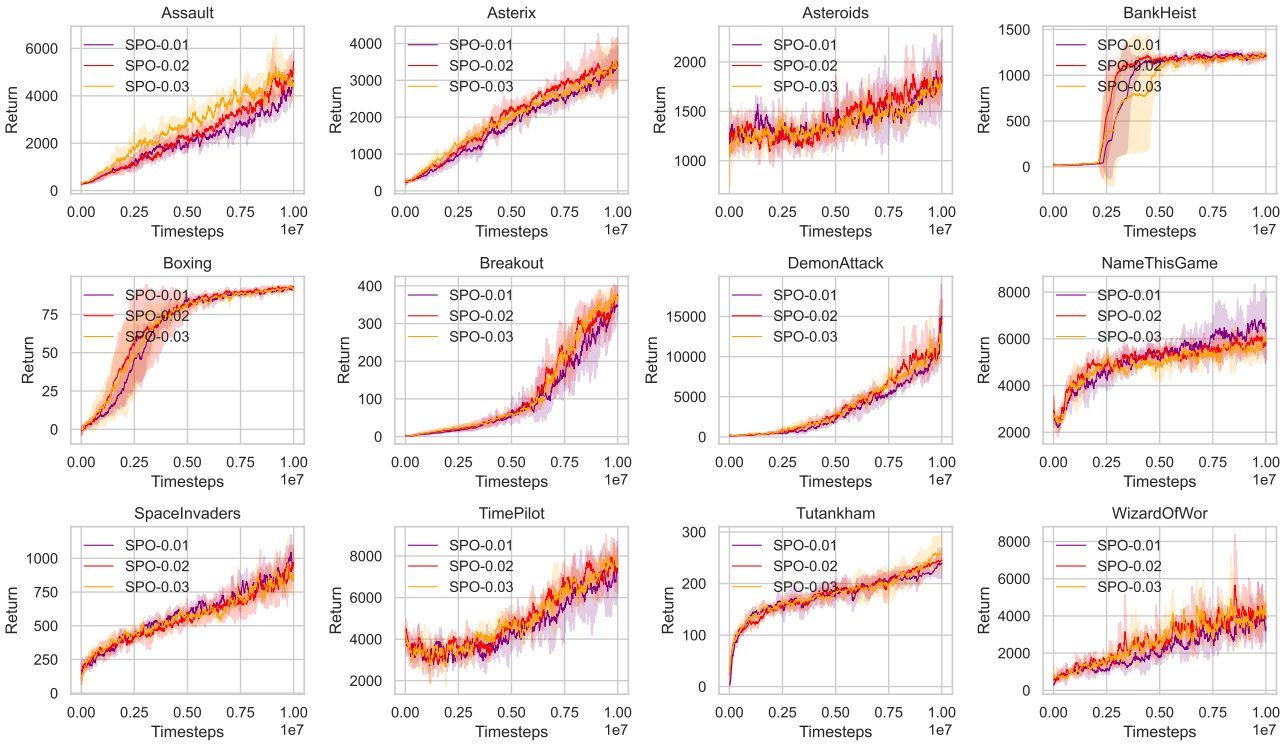
(a) Training curves of average KL divergence.



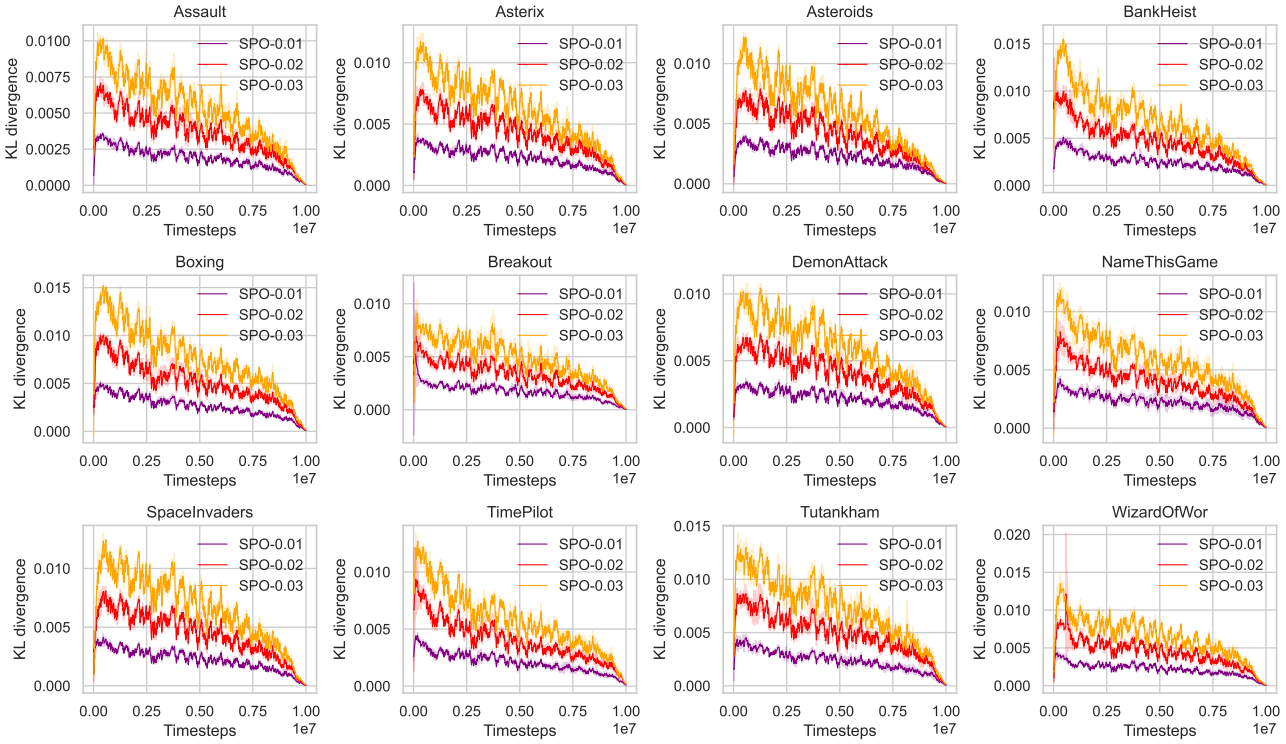
(b) Training curves of policy entropy.

Figure 11. Training curves of average KL divergence and policy entropy in Atari 2600 environments.

B.6 Sensitivity Analysis



(a) Training curves of return.



(b) Training curves of average KL divergence.

Figure 12. Training curves of return and average KL divergence in Atari 2600 environments.

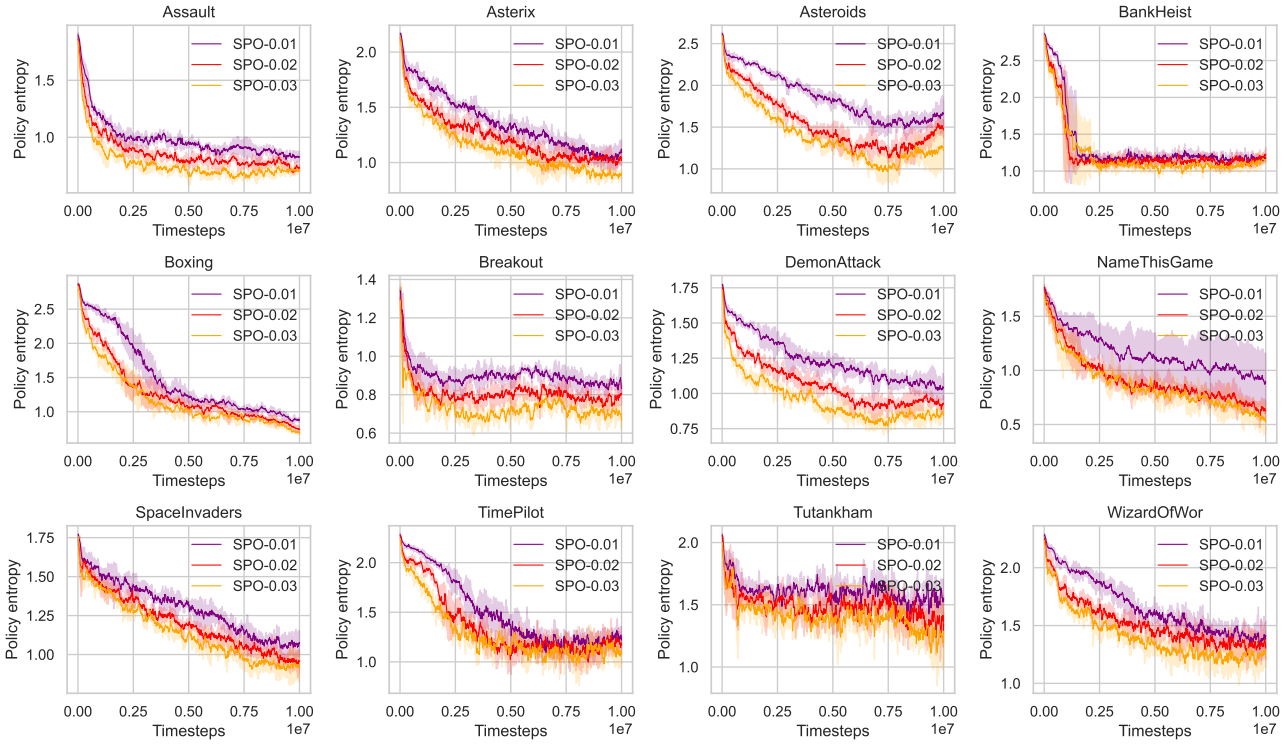


Figure 13. Training curves of policy entropy in Atari 2600 environments.

C Network Structure

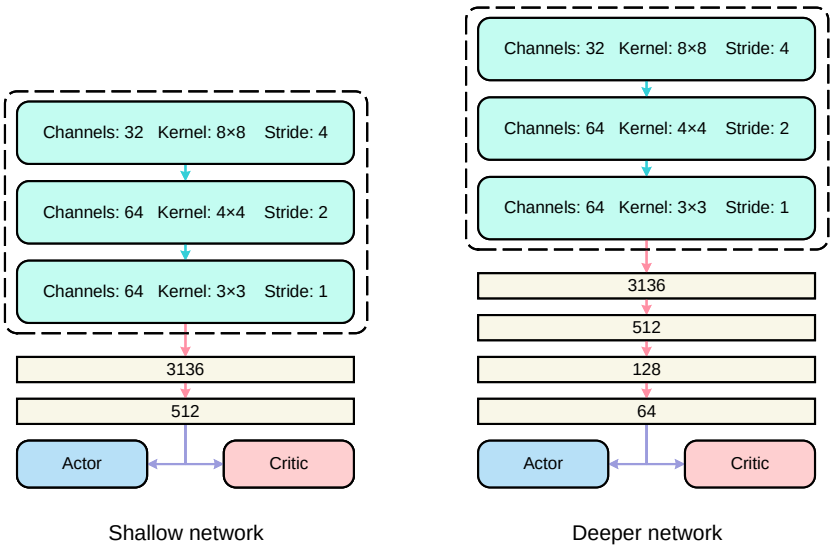


Figure 14. Shallow network and deeper network structure in Atari 2600 environments.

D The Scalability of SPO for Continuous Action Space

We further tested the scalability of the SPO algorithm in a continuous action space environment, i.e., MuJoCo (Todorov et al., 2012), and surprisingly, SPO still performed exceptionally well.

Typically, in the continuous action space, we assume that the output of the policy network follows a multivariate Gaussian distribution, and we further assume that the components are mutually independent. Therefore, the KL divergence between them is the sum of the KL divergences of the individual components.

Now assuming that two random variables follow Gaussian distributions: $p(x) = \mathcal{N}(\mu_1, \sigma_1^2)$ and $q(x) = \mathcal{N}(\mu_2, \sigma_2^2)$, then their KL divergence is

$$D_{\mathrm{KL}}(p, q) = \log \frac{\sigma_2}{\sigma_1} + \frac{(\mu_1 - \mu_2)^2 + \sigma_1^2 - \sigma_2^2}{2\sigma_2^2}. \tag{17}$$

Proof. The KL divergence between two given random distributions $p(x)$ and $q(x)$ is defined as

$$\begin{aligned} D_{\mathrm{KL}}(p, q) &= \int p(x) \cdot \log \frac{p(x)}{q(x)} \mathrm{d}x \\ &= \int p(x) \cdot \log \left[\frac{\sigma_2}{\sigma_1} \cdot \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2} \right) \right] \mathrm{d}x \\ &= \int p(x) \cdot \log \frac{\sigma_2}{\sigma_1} \mathrm{d}x + \int p(x) \cdot \left(-\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2} \right) \mathrm{d}x \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 - \sigma_2^2}{2\sigma_1^2\sigma_2^2} \cdot \underbrace{\int x^2 p(x) \mathrm{d}x}_{M} + \frac{\mu_1\sigma_2^2 - \mu_2\sigma_1^2}{\sigma_1^2\sigma_2^2} \cdot \underbrace{\int x p(x) \mathrm{d}x}_{N} + \frac{\mu_2^2\sigma_1^2 - \mu_1^2\sigma_2^2}{2\sigma_1^2\sigma_2^2} \cdot \underbrace{\int p(x) \mathrm{d}x}_{=1}, \end{aligned} \tag{18}$$

where

$$M = \mathbb{E}_{x\sim p(x)}(x^2) = \mathbb{D}_{x\sim p(x)}(x) + \mathbb{E}_{x\sim p(x)}(x)^2 = \sigma_1^2 + \mu_1^2, \quad N = \mathbb{E}_{x\sim p(x)}(x) = \mu_1, \tag{19}$$

so we have

$$\begin{aligned} D_{\mathrm{KL}}(p, q) &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 - \sigma_2^2}{2\sigma_1^2\sigma_2^2} \cdot (\sigma_1^2 + \mu_1^2) + \frac{\mu_1\sigma_2^2 - \mu_2\sigma_1^2}{\sigma_1^2\sigma_2^2} \cdot \mu_1 + \frac{\mu_2^2\sigma_1^2 - \mu_1^2\sigma_2^2}{2\sigma_1^2\sigma_2^2} \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{(\mu_1 - \mu_2)^2 + \sigma_1^2 - \sigma_2^2}{2\sigma_2^2}. \end{aligned} \tag{20}$$

□

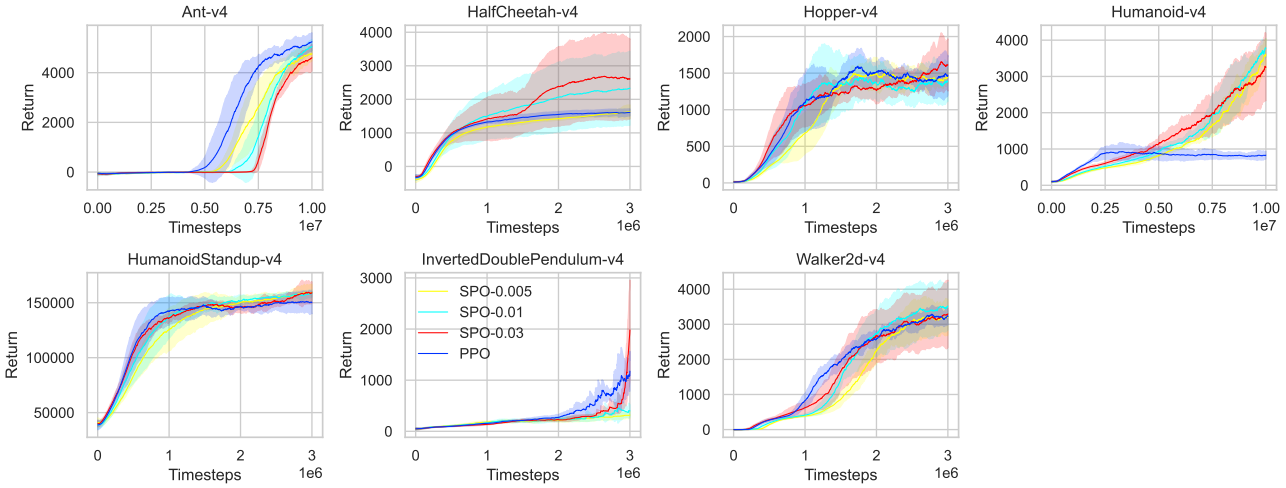
For each environment in MuJoCo, we compute and visualize the standard deviation of the algorithm's performance across 5 separate runs with different random seeds, the results are shown in Table 3, Table 4 and Figure 15.

Table 3. Average return of the last 10% training steps in MuJoCo environments.

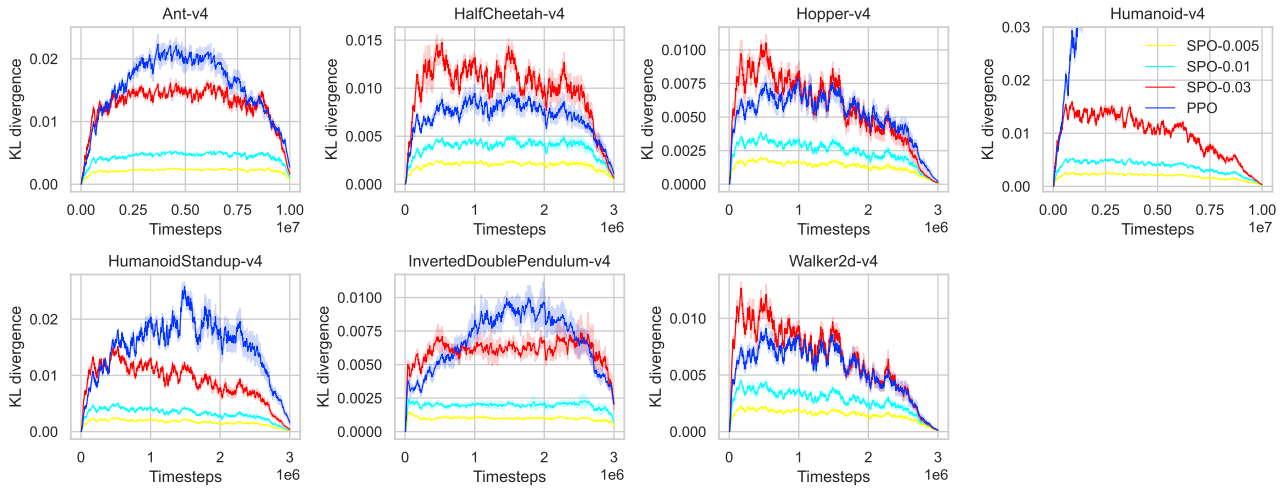
Algorithm	Ant-v4	HalfCheetah-v4	Hopper-v4	Humanoid-v4	HumanoidStandup-v4	InvertedDoublePendulum-v4	Walker2d-v4
SPO-0.005	4775.95	1623.82	1434.74	3587.51	159441.18	317.34	3187.8
SPO-0.01	5066.14	2328.6	1377.1	**3772.78**	158407.45	416.64	**3508.62**
SPO-0.03	4628.01	**2619.4**	**1632.63**	3168.32	**159841.75**	**1485.7**	3286.5
PPO	**5216.36**	1607.55	1471.81	822.57	150445.3	1133.99	3217.51

Table 4. Average return throughout the entire training process in MuJoCo environments.

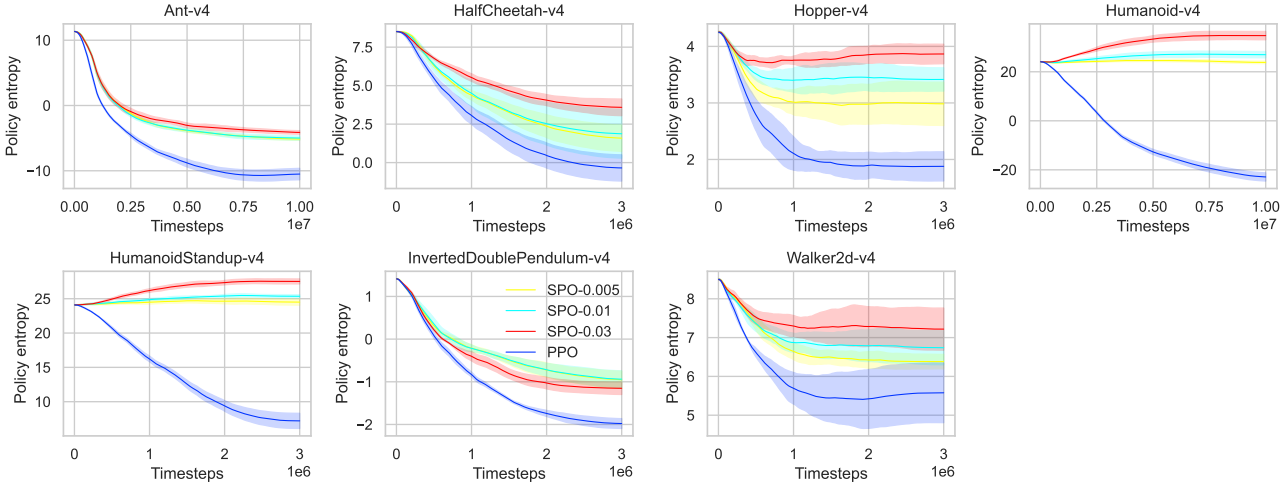
Algorithm	Ant-v4	HalfCheetah-v4	Hopper-v4	Humanoid-v4	HumanoidStandup-v4	InvertedDoublePendulum-v4	Walker2d-v4
SPO-0.005	1466.71	1229.05	1046.8	1318.69	132000.34	206.88	1530.99
SPO-0.01	1249.85	1705.9	1088.38	1457.89	136563.02	222.5	1803.99
SPO-0.03	1035.62	**1861.74**	1132.27	**1501.02**	**136769.98**	333.29	1788.93
PPO	**1991.05**	1351.08	**1155.63**	793.51	136209.27	**363.97**	**1893.36**



(a) Training curves of return.



(b) Training curves of average KL divergence.



(c) Training curves of policy entropy.

Figure 15. Training curves in MuJoCo environments.