
Simple Policy Optimization

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
Anonymous Authors¹

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.

¹Anonymous 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.

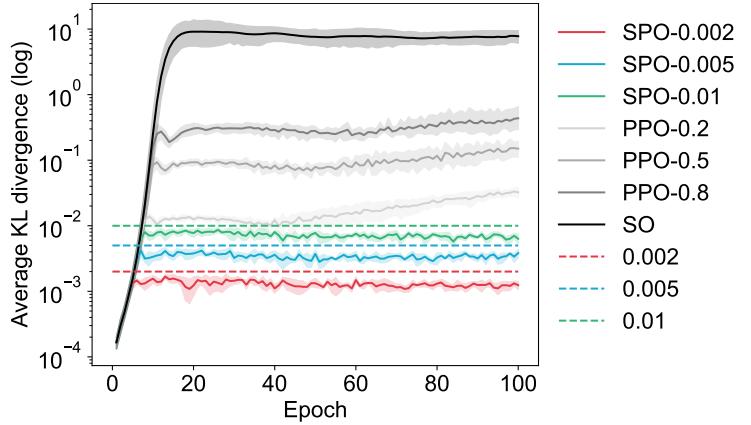


Figure 1. We over-optimize the objective in the Amidar-v5 environment for 100 epochs and select five random seeds for experiments. SPO- x represents the SPO algorithm with the maximum KL divergence set to x , PPO- x represents the PPO algorithm with the ratio clipping parameter set to x , and SO represents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

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

the objective function can be written as

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

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

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

$\nabla J(\theta)$ 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 $\alpha = D_{\text{TV}}^{\max}(\pi, \tilde{\pi})$ and $\epsilon = \max_s |\mathbb{E}_{a \sim \tilde{\pi}} [A^{\pi}(s, a)]|$, the following bound holds:

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

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 $D_{\text{TV}}(\pi, \tilde{\pi})^2 \leq D_{\text{KL}}(\pi, \tilde{\pi})$, we have the following corollary.

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

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

Through a heuristic approximation, the maximum KL divergence D_{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,

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

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

110 **Algorithm 1** Simple Policy Optimization (SPO)

111 1: **Input:** Policy network parameters θ ; value network parameters w ; threshold of KL divergence d_{\max}
 112 2: **Output:** Optimal policy network parameters θ^*
 113 3: **while** not converged **do**
 114 4: **# Data collection**
 115 5: Collect data $X = \{(s_i, a_i, r_i)\}_{i=1}^N$ using the current policy network π_θ
 116 6: **for** each training epoch **do**
 117 7: **# Estimate advantage function**
 118 8: Use GAE (Schulman et al., 2015b) technique to estimate the advantage $\hat{A}(s_t, a_t)$
 119 9: **# Calculate KL divergence**
 120 10: For each data (s_t, a_t, r_t) calculate $d \leftarrow \sum_{a \in \mathcal{A}} \pi_{\theta_{\text{old}}}(a|s_t) \cdot \log \frac{\pi_\theta(a|s_t)}{\pi_{\theta_{\text{old}}}(a|s_t)}$ and $d_{\text{clip}} \leftarrow \text{clip}(d, 0, d_{\max})$
 121 11: **# Calculate policy loss l_p , value loss l_v and policy entropy l_e**
 122 12: Policy loss
 123
 124
$$l_p \leftarrow \begin{cases} -\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t), & \theta = \theta_{\text{old}} \\ -\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \cdot \hat{A}(s_t, a_t) \cdot \left\{ \frac{d_{\text{clip}}}{d} + \text{sign}[\hat{A}(s_t, a_t)] - 1 \right\} \cdot \text{sign}[\hat{A}(s_t, a_t)], & \theta \neq \theta_{\text{old}} \end{cases}$$

 125
 126
 127
 128
 129 and total loss $l \leftarrow l_p + c_1 \cdot l_v - c_2 \cdot l_e$
 130 13: **# Update parameters**
 131 14: Update parameters θ and w through backpropagation
 132 15: **end for**
 133 16: Current policy network parameters $\theta_{\text{old}} \leftarrow \theta$
 134 17: **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,

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

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

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

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

$$d = D_{\text{KL}}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] = \sum_{a \in \mathcal{A}} \pi_{\theta_{\text{old}}}(a|s_t) \cdot \log \frac{\pi_{\theta_{\text{old}}}(a|s_t)}{\pi_\theta(a|s_t)}, \quad (7)$$

and the surrogate objective

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

According to the Corollary 2.2, we hope to optimize the surrogate objective $\mathfrak{O}_{\theta_{\text{old}}}^\theta(s_t, a_t)$ 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:

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

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

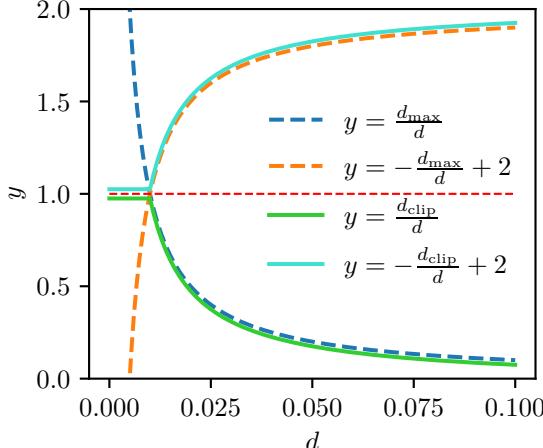


Figure 2. Function curves of $(d_{\text{clip}}/d + \sigma - 1) \cdot \sigma$ when d_{max} is set to 0.01.

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 $d \leq d_{\text{max}}$, the objective is the surrogate objective $\mathfrak{D}_{\theta_{\text{old}}}^{\theta}(s_t, a_t)$. Otherwise, if $d > d_{\text{max}}$, then the objective becomes the ratio form of (9). The formalization will be

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

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

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

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:

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

where

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

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

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

Furthermore, if $d \leq d_{\text{max}}$, 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 $\mathfrak{D}_{\theta_{\text{old}}}^{\theta}(s_t, a_t)$. Otherwise,

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

We will provide intuition about the objective (12), suppose that $d_{\text{max}} = 0.01$, and the function curves is shown in Figure 2. If $\hat{A}(s_t, a_t) > 0$ and $d > d_{\text{max}}$, 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 $d > d_{\text{max}}$, 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 $\mathfrak{D}_{\theta_{\text{old}}}^{\theta}(s_t, a_t)$ within the trust region $d \leq d_{\text{max}}$ 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

Simple Policy Optimization

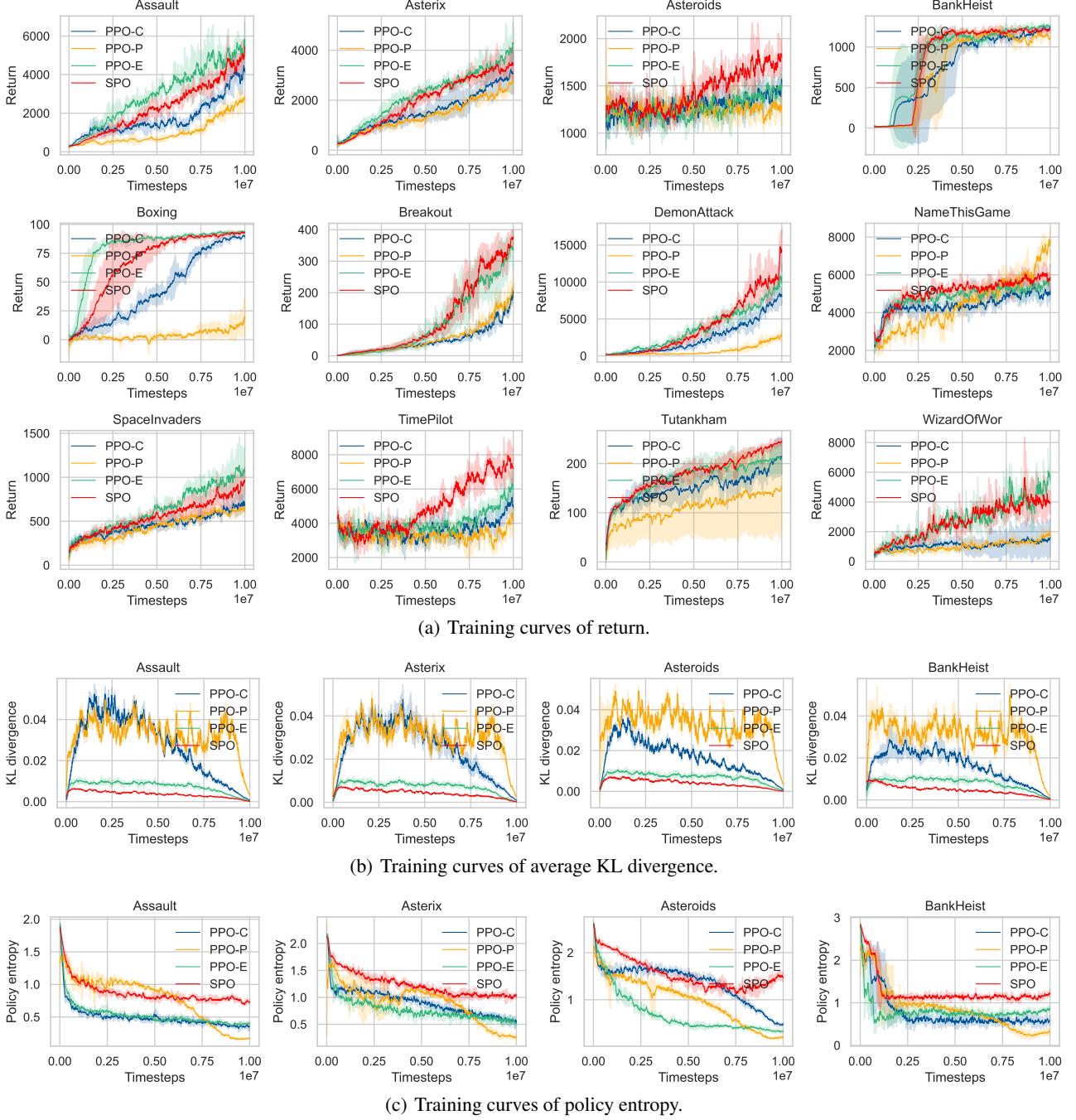


Figure 3. Training curves of different PPO variants and SPO algorithm in Atari 2600 environments. Among them, PPO-C represents PPO-CLIP, PPO-P represents PPO-PENALTY, and PPO-E represents PPO-CLIP with an early stopping strategy.

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.

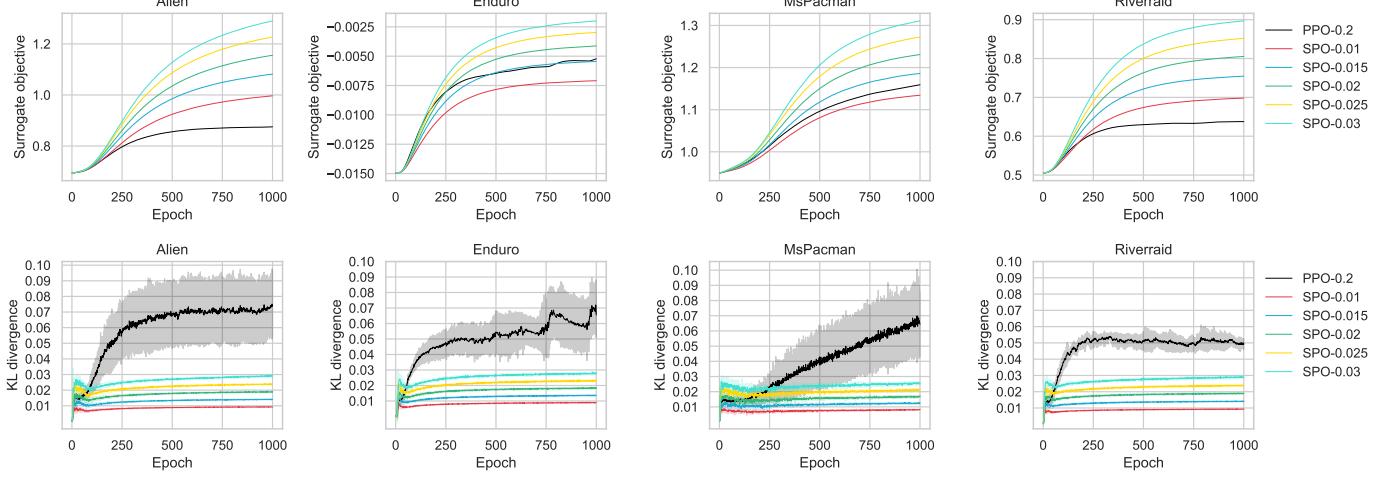


Figure 4. We over-optimize the objectives of PPO and SPO for 1000 epochs in the early stages of training, where d_{\max} is set to 0.01, 0.015, 0.02, 0.025 and 0.03.

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 F. 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 C. 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

Simple Policy Optimization

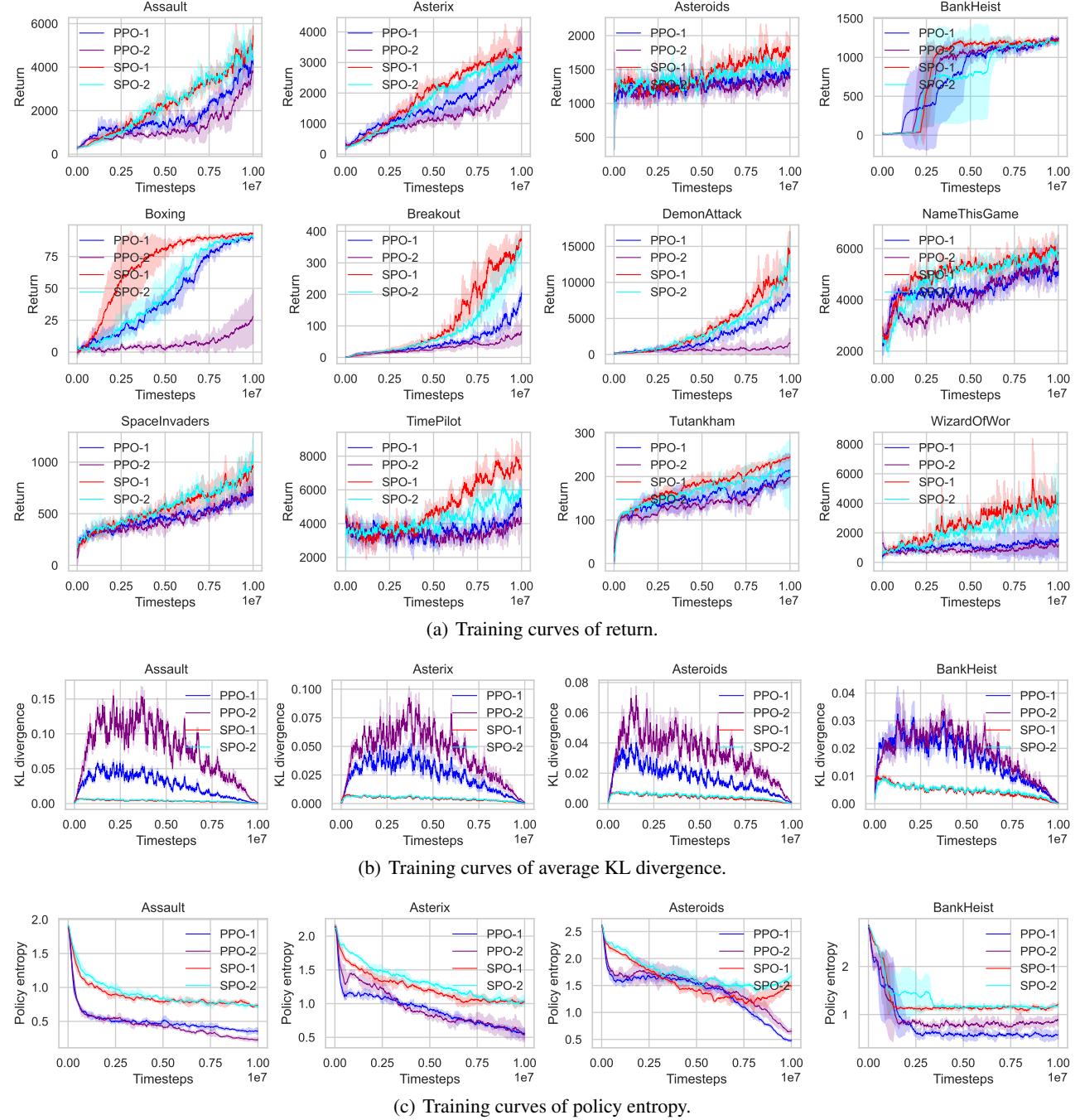


Figure 5. Training curves of PPO and SPO algorithms using networks with different depths in Atari 2600 environments, where PPO-1 and SPO-1 represent the use of shallow networks, while PPO-2 and SPO-2 represent the use of deeper networks.

much as possible, and performs better than PPO.

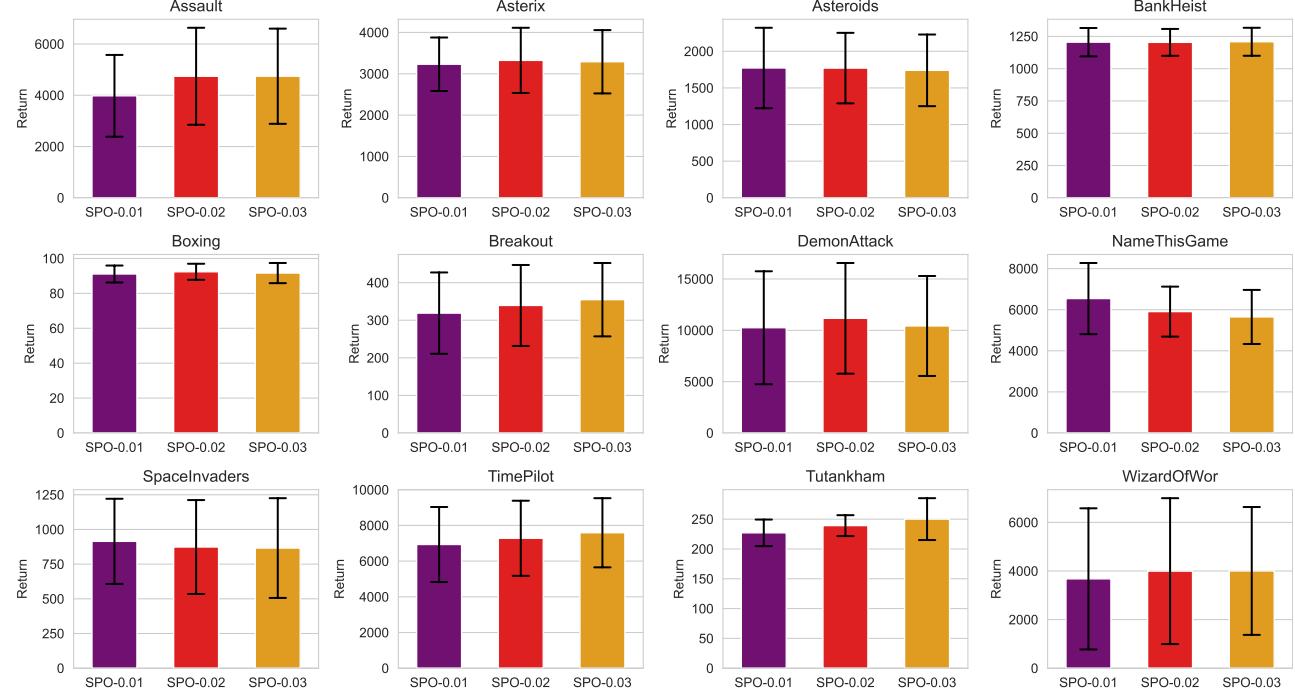
4.3 Increasing Network Complexity

We are interested in understanding how the depth or complexity of the neural networks¹ affects the performance of

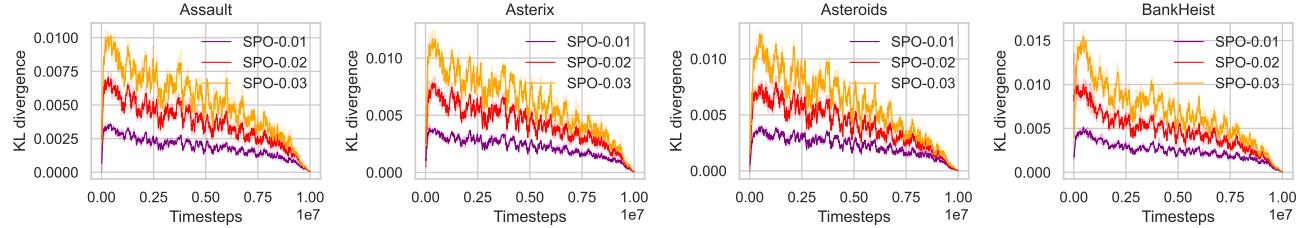
¹Two neural network structures are shown in the appendix F.

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?

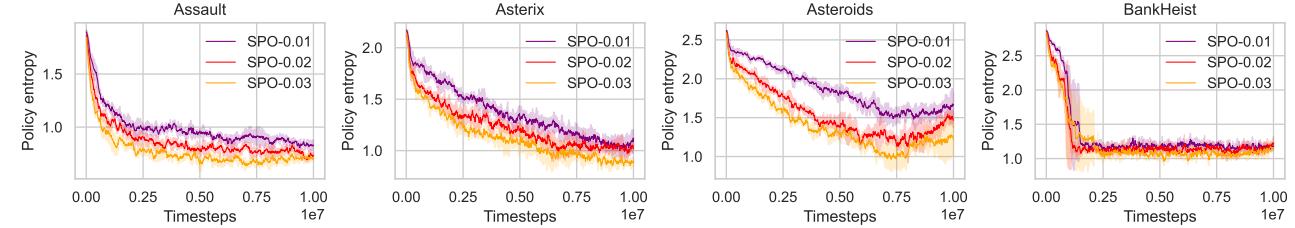
Simple Policy Optimization



(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

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

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.

- 495 Ho, J., Jain, A., and Abbeel, P. Denoising diffusion proba-
 496 bility models. *Advances in neural information process-*
 497 *ing systems*, 33:6840–6851, 2020.
 498
- 499 Huang, S., Dossa, R. F. J., Raffin, A., Kan-
 500 kervisto, A., and Wang, W. The 37 imple-
 501 mentation details of proximal policy optimiza-
 502 tion. In *ICLR Blog Track*, 2022. URL <https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/>.
 503
- 504
- 505 Kakade, S. and Langford, J. Approximately optimal ap-
 506 proximate reinforcement learning. In *Proceedings of the*
 507 *Nineteenth International Conference on Machine Learn-*
 508 *ing*, pp. 267–274, 2002.
 509
- 510 Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sal-
 511 lab, A. A., Yogamani, S., and Pérez, P. Deep reinforce-
 512 ment learning for autonomous driving: A survey. *IEEE*
 513 *Transactions on Intelligent Transportation Systems*, 23
 514 (6):4909–4926, 2021.
 515
- 516 Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Ad-*
 517 *vances in neural information processing systems*, 12,
 518 1999.
 519
- 520 Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T.,
 521 Bishop, C., Carbune, V., and Rastogi, A. Rlafif: Scal-
 522 ing reinforcement learning from human feedback with ai
 523 feedback. *arXiv preprint arXiv:2309.00267*, 2023.
 524
- 525 Machado, M. C., Bellemare, M. G., Talvitie, E., Veness,
 526 J., Hausknecht, M., and Bowling, M. Revisiting the
 527 arcade learning environment: Evaluation protocols and
 528 open problems for general agents. *Journal of Artificial*
 529 *Intelligence Research*, 61:523–562, 2018.
 530
- 531 Markowitz, J. and Staley, E. W. Clipped-objective pol-
 532 icy gradients for pessimistic policy optimization. *arXiv*
 533 *preprint arXiv:2311.05846*, 2023.
 534
- 535 Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A.,
 536 Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing
 537 atari with deep reinforcement learning. *arXiv preprint*
 538 *arXiv:1312.5602*, 2013.
 539
- 540 Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap,
 541 T., Harley, T., Silver, D., and Kavukcuoglu, K. Asyn-
 542 chronous methods for deep reinforcement learning. In
 543 *International conference on machine learning*, pp. 1928–
 544 1937. PMLR, 2016.
 545
- 546 Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.,
 547 Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A.,
 548 et al. Training language models to follow instructions
 549 with human feedback. *Advances in neural information*
 550 *processing systems*, 35:27730–27744, 2022.
 551
- 552 Queeney, J., Paschalidis, Y., and Cassandras, C. G. Gen-
 553 eralized proximal policy optimization with sample reuse.
 554 *Advances in Neural Information Processing Systems*, 34:
 555 11909–11919, 2021.
 556
- 557 Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Er-
 558 mon, S., and Finn, C. Direct preference optimization:
 559 Your language model is secretly a reward model. *Ad-*
 560 *vances in Neural Information Processing Systems*, 36,
 561 2024.
 562
- 563 Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz,
 564 P. Trust region policy optimization. In *International*
 565 *conference on machine learning*, pp. 1889–1897. PMLR,
 566 2015a.
 567
- 568 Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel,
 569 P. High-dimensional continuous control using generalized
 570 advantage estimation. *arXiv preprint arXiv:1506.02438*,
 571 2015b.
 572
- 573 Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and
 574 Klimov, O. Proximal policy optimization algorithms.
 575 *arXiv preprint arXiv:1707.06347*, 2017.
 576
- 577 Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L.,
 578 Van Den Driessche, G., Schrittwieser, J., Antonoglou, I.,
 579 Panneershelvam, V., Lanctot, M., et al. Mastering the
 580 game of go with deep neural networks and tree search.
 581 *nature*, 529(7587):484–489, 2016.
 582
- 583 Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I.,
 584 Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M.,
 585 Bolton, A., et al. Mastering the game of go without
 586 human knowledge. *nature*, 550(7676):354–359, 2017.
 587
- 588 Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai,
 589 M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Grae-
 590 pel, T., et al. A general reinforcement learning algorithm
 591 that masters chess, shogi, and go through self-play. *Sci-
 592 ence*, 362(6419):1140–1144, 2018.
 593
- 594 Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R.,
 595 Voss, C., Radford, A., Amodei, D., and Christiano,
 596 P. F. Learning to summarize with human feedback. *Ad-*
 597 *vances in Neural Information Processing Systems*, 33:
 598 3008–3021, 2020.
 599
- 600 Sun, M., Kurin, V., Liu, G., Devlin, S., Qin, T., Hofmann,
 601 K., and Whiteson, S. You may not need ratio clipping in
 602 ppo. *arXiv preprint arXiv:2202.00079*, 2022.
 603
- 604 Teng, S., Hu, X., Deng, P., Li, B., Li, Y., Ai, Y., Yang, D.,
 605 Li, L., Xuanyuan, Z., Zhu, F., et al. Motion planning
 606 for autonomous driving: The state of the art and future
 607 perspectives. *IEEE Transactions on Intelligent Vehicles*,
 608 2023.
 609

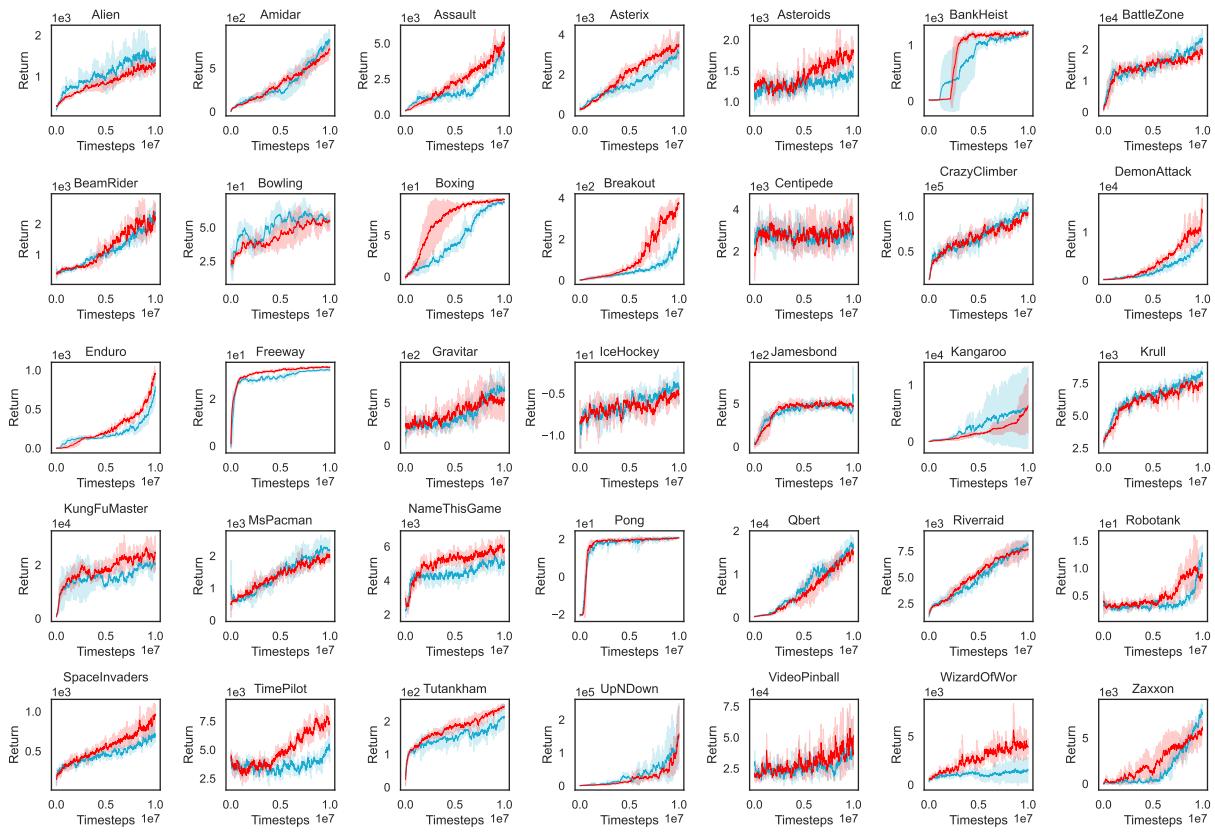
- 550 Tunstall, L., Beeching, E., Lambert, N., Rajani, N., Ra-
551 sul, K., Belkada, Y., Huang, S., von Werra, L., Fourrier,
552 C., Habib, N., et al. Zephyr: Direct distillation of lm
553 alignment. *arXiv preprint arXiv:2310.16944*, 2023.
- 554
- 555 Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones,
556 L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. At-
557 tention is all you need. *Advances in neural information*
558 *processing systems*, 30, 2017.
- 559 Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M.,
560 Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds,
561 T., Georgiev, P., et al. Grandmaster level in starcraft ii
562 using multi-agent reinforcement learning. *Nature*, 575
563 (7782):350–354, 2019.
- 564
- 565 Wang, K., Zhou, D., Feng, J., and Mannor, S. Ppg reloaded:
566 An empirical study on what matters in phasic policy gra-
567 dient. 2023.
- 568
- 569 Wang, Y., He, H., and Tan, X. Truly proximal policy op-
570 timization. In *Uncertainty in Artificial Intelligence*, pp.
571 113–122. PMLR, 2020.
- 572
- 573 Wen, B., Lian, W., Bekris, K., and Schaal, S. You
574 only demonstrate once: Category-level manipulation
575 from single visual demonstration. *arXiv preprint*
576 *arXiv:2201.12716*, 2022.
- 577
- 578 Williams, R. J. Simple statistical gradient-following algo-
579 rithms for connectionist reinforcement learning. *Machine*
580 *learning*, 8:229–256, 1992.
- 581
- 582 Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H.,
583 Yang, S., Wu, X., Guo, Q., et al. Mastering complex
584 control in moba games with deep reinforcement learn-
585 ing. In *Proceedings of the AAAI Conference on Artificial*
Intelligence, volume 34, pp. 6672–6679, 2020.
- 586
- 587 Yu, C. and Wang, P. Dexterous manipulation for multi-
588 fingered robotic hands with reinforcement learning: a
589 review. *Frontiers in Neurorobotics*, 16:861825, 2022.
- 590
- 591 Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford,
592 A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning
593 language models from human preferences. *arXiv preprint*
arXiv:1909.08593, 2019.
- 594
- 595
- 596
- 597
- 598
- 599
- 600
- 601
- 602
- 603
- 604

A Hyperparameters Setting
Table 1. Detailed hyperparameters

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
Max KL divergence d_{\max}	\	\	\	0.02
Threshold of KL divergence	\	0.02	0.02	\

660 B More Experiment Results

661 B.1 Training Curves of Return



690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714

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

B.2 Training Curves of Average KL Divergence

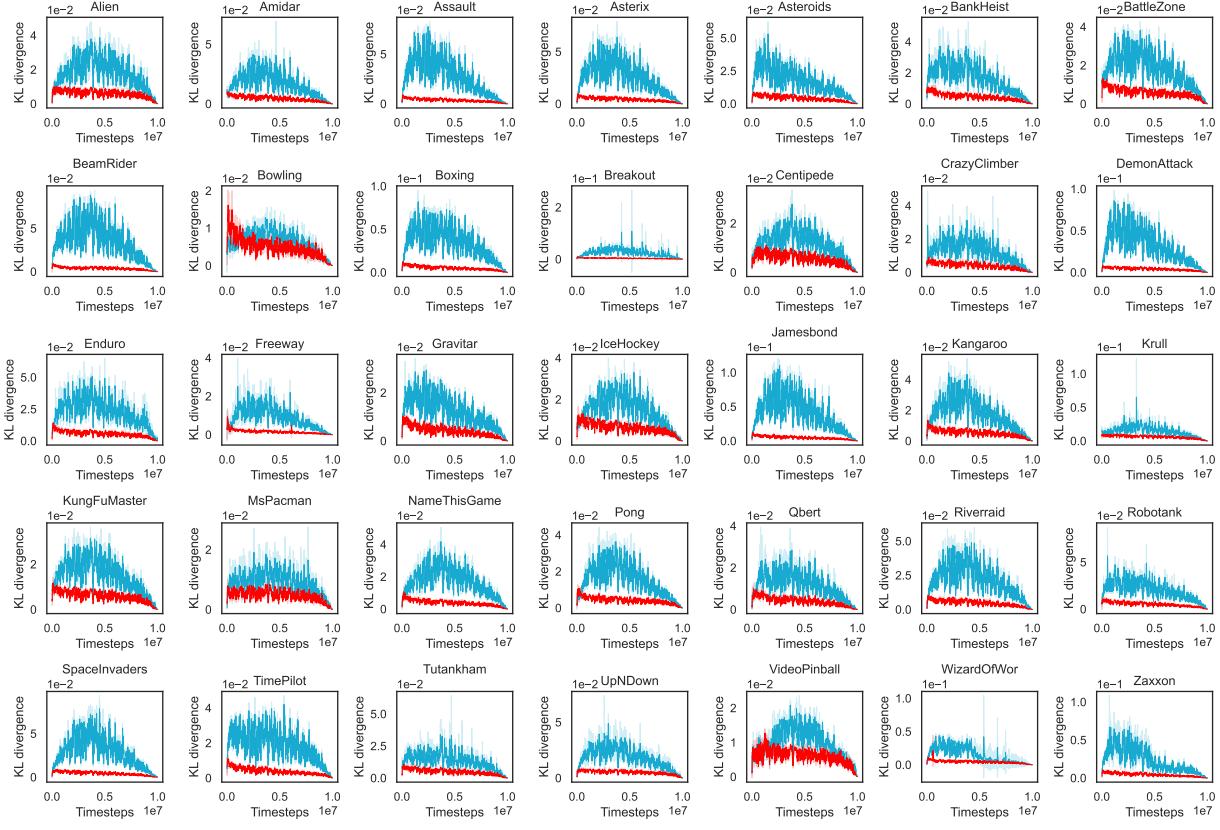


Figure 8. Training curves of average KL divergence in Atari 2600 environments.

B.3 Training Curves of Policy Entropy

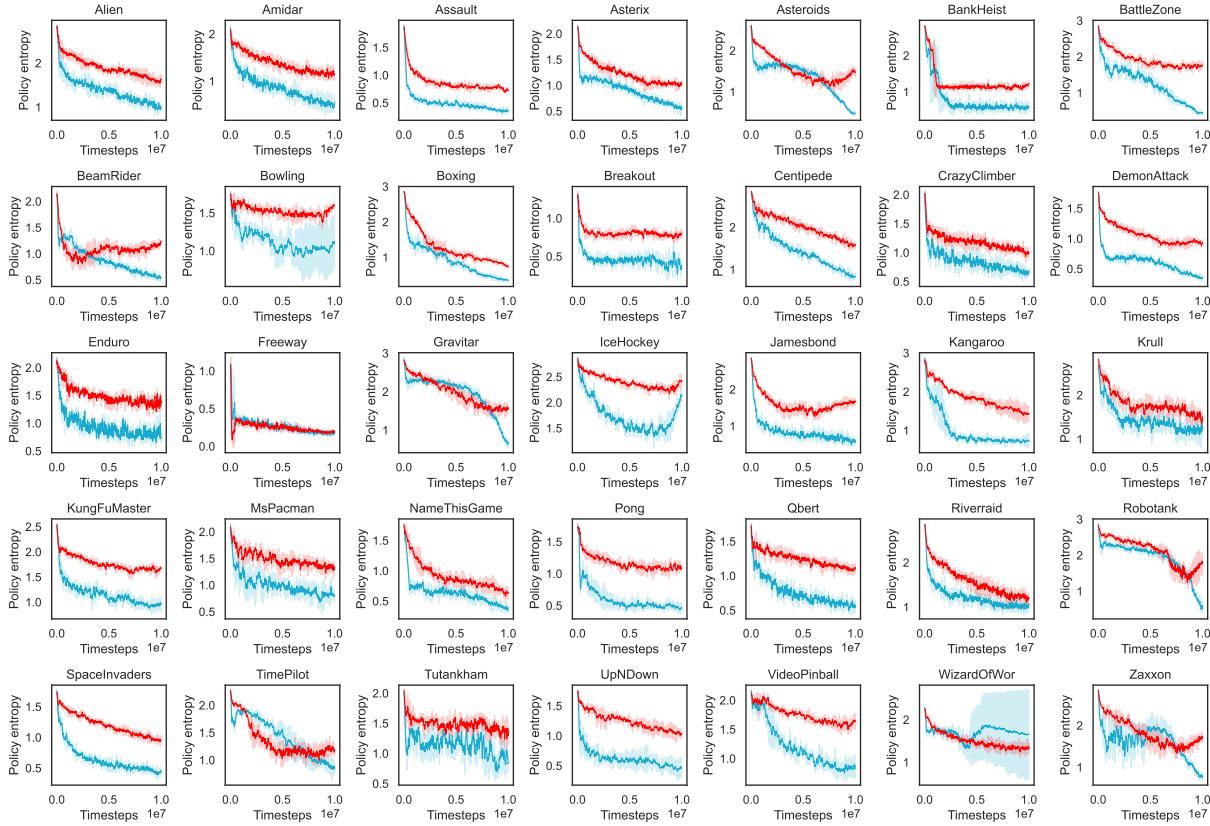


Figure 9. Training curves of policy entropy in Atari 2600 environments.

825
826
827 C Comparison to PPOs
828
829
830
831
832
833
834

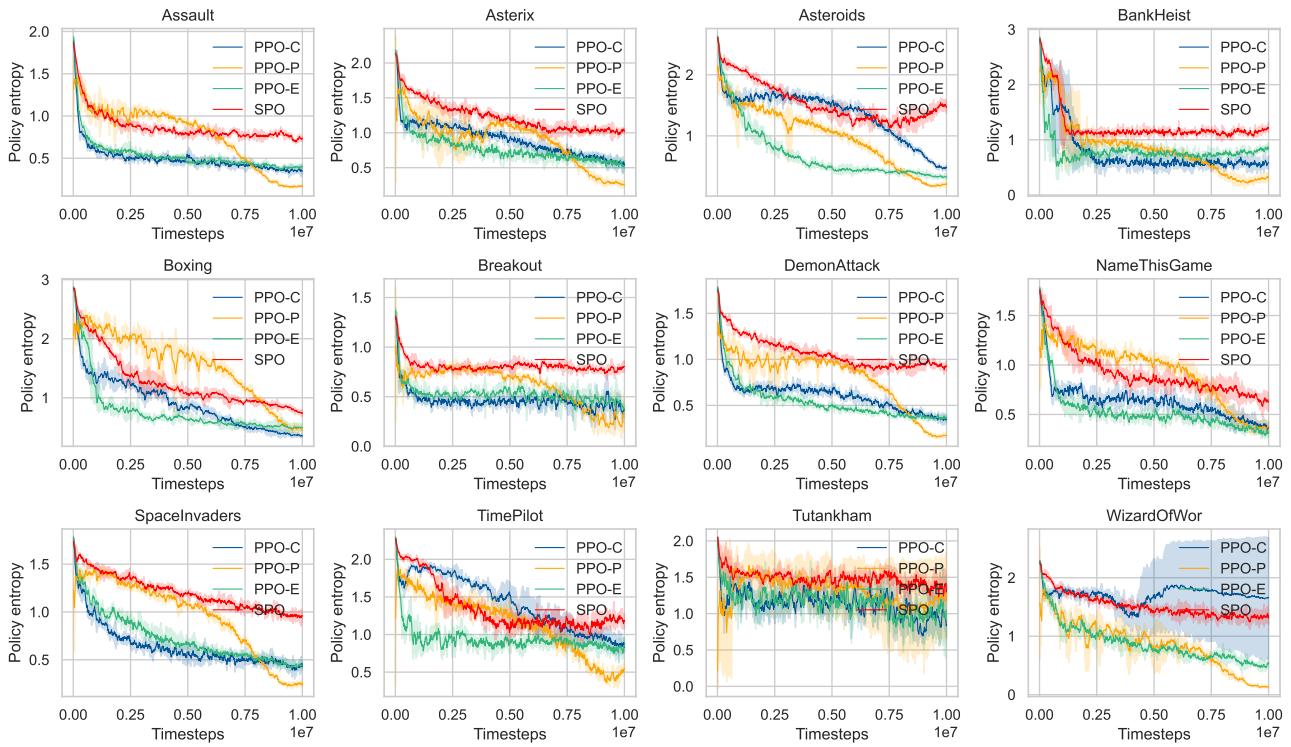
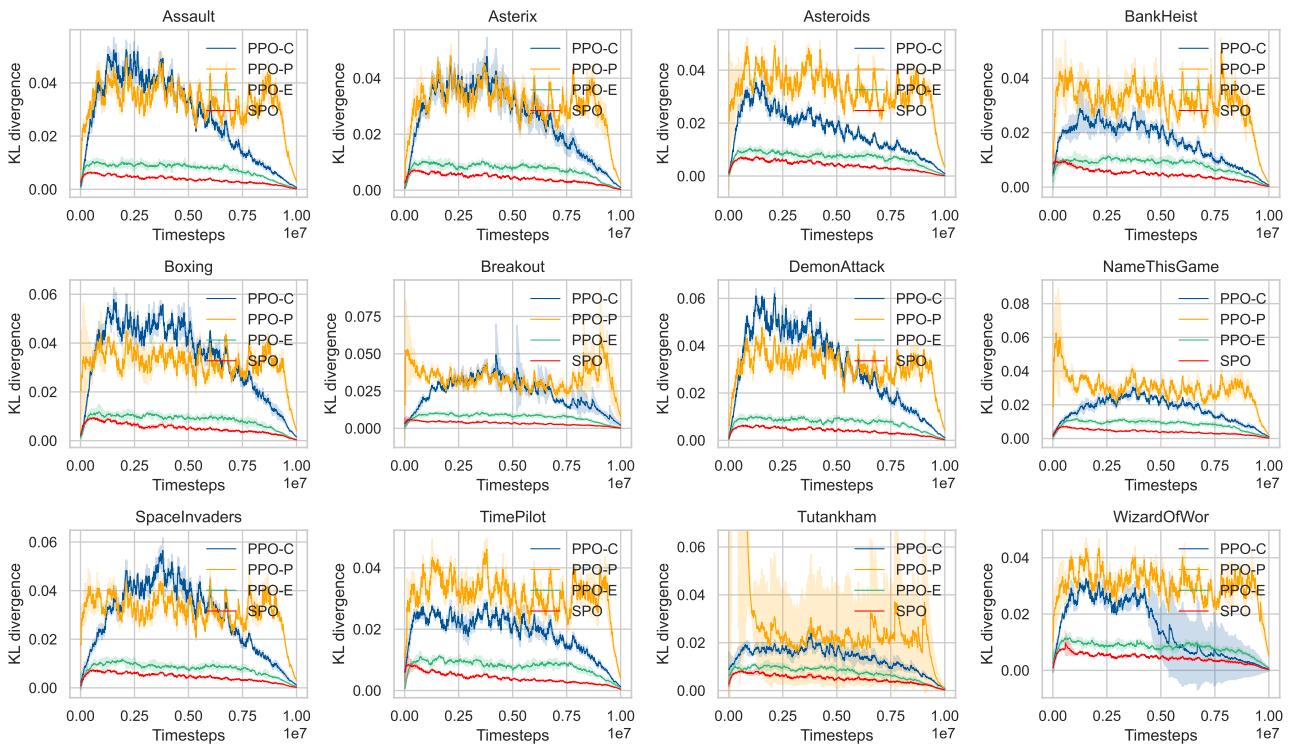
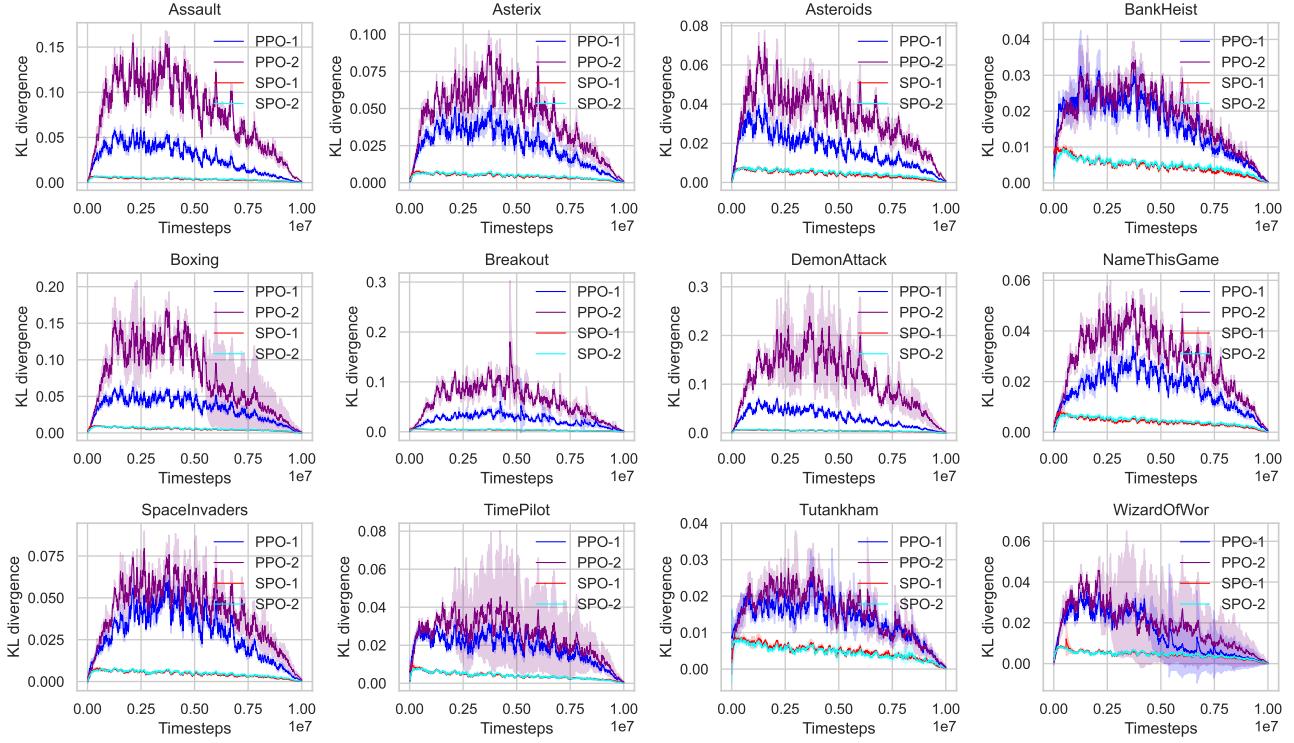
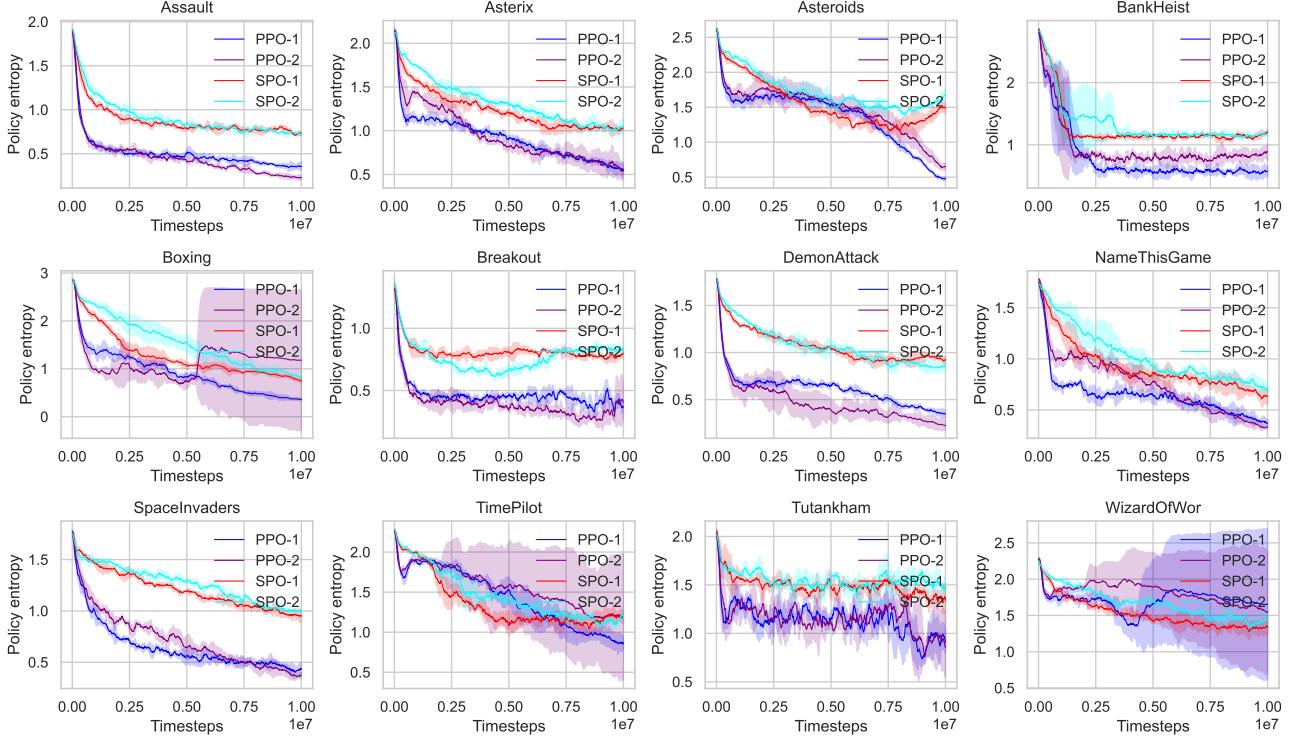


Figure 10. Training curves of average KL divergence and policy entropy in Atari 2600 environments.

D Increasing Network Complexity



(a) Training curves of average KL divergence.

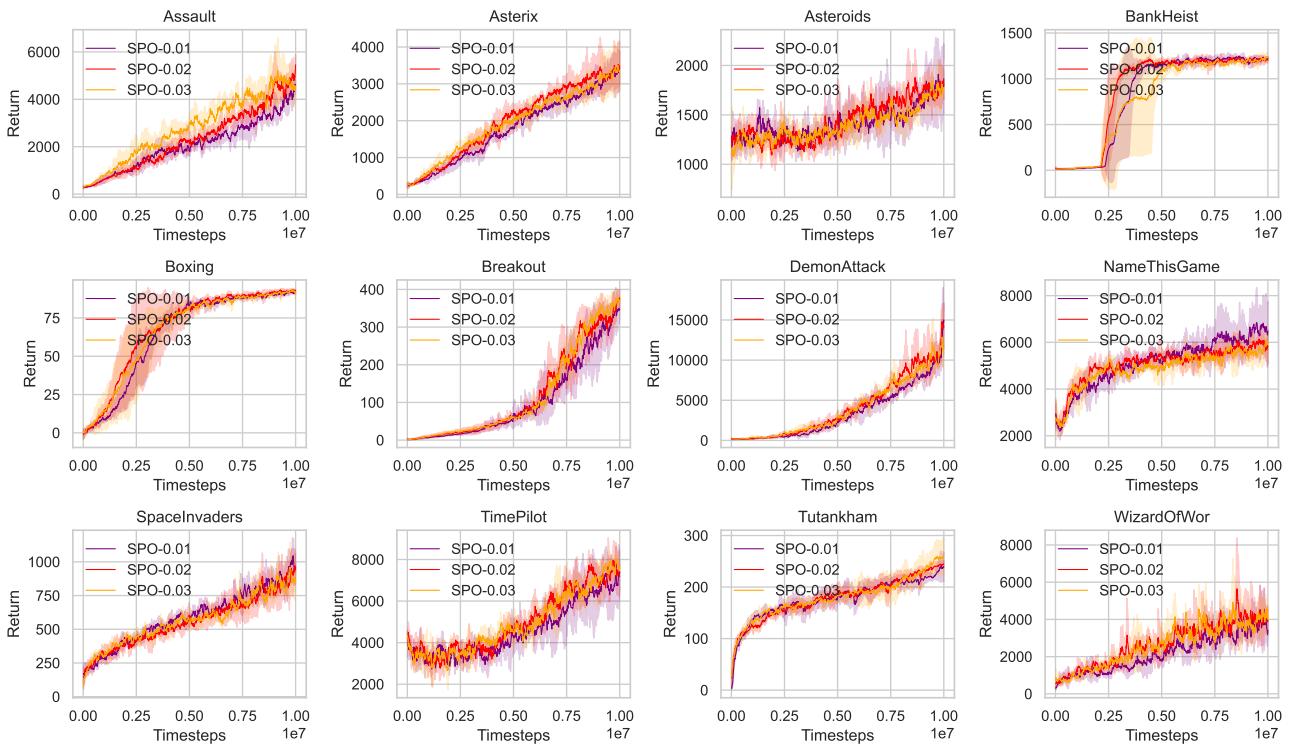


(b) Training curves of policy entropy.

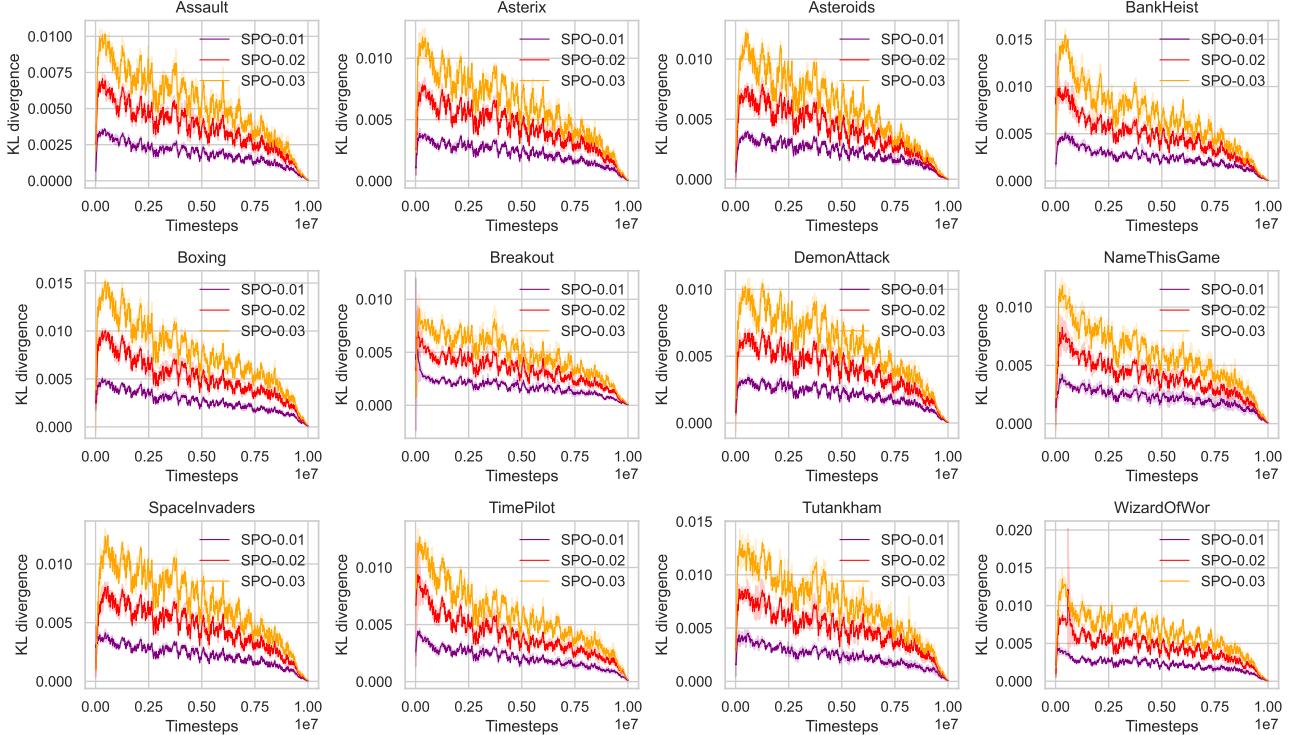
Figure 11. Training curves of average KL divergence and policy entropy in Atari 2600 environments.

935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989

E 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.

Simple Policy Optimization

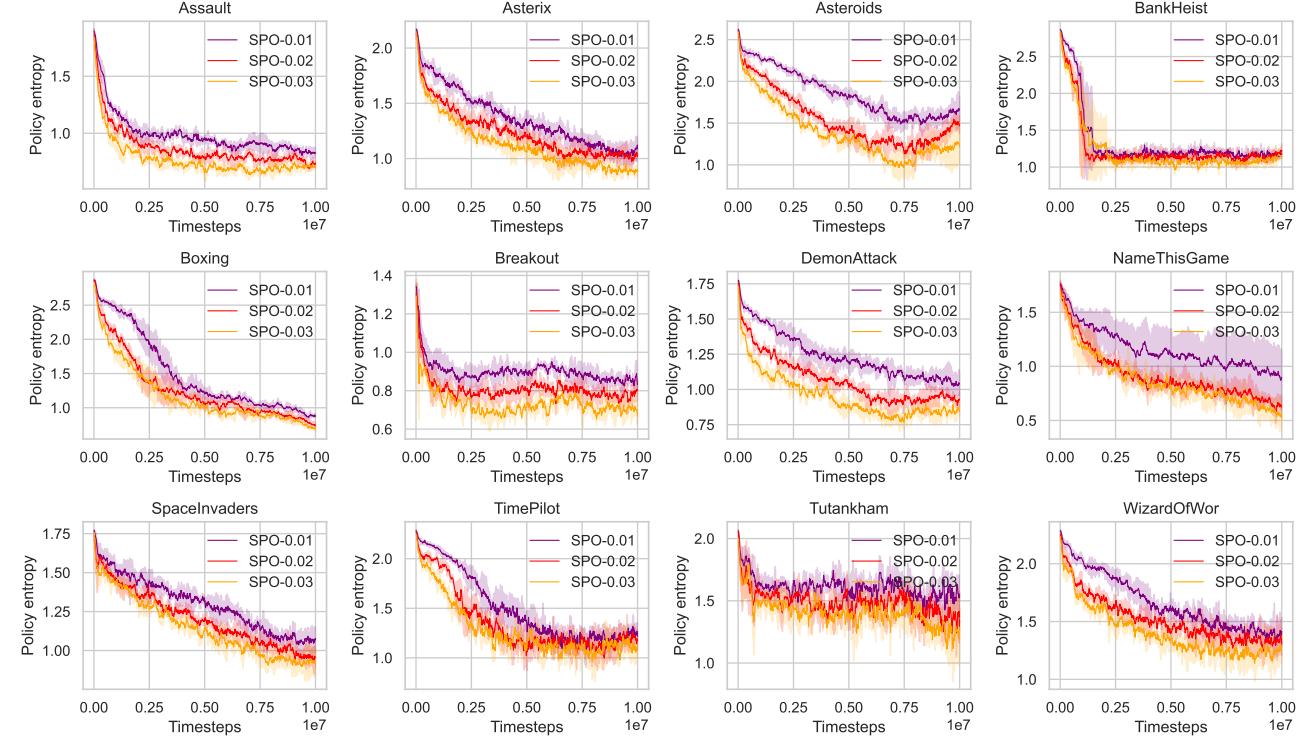


Figure 13. Training curves of policy entropy in Atari 2600 environments.

F Network Structure

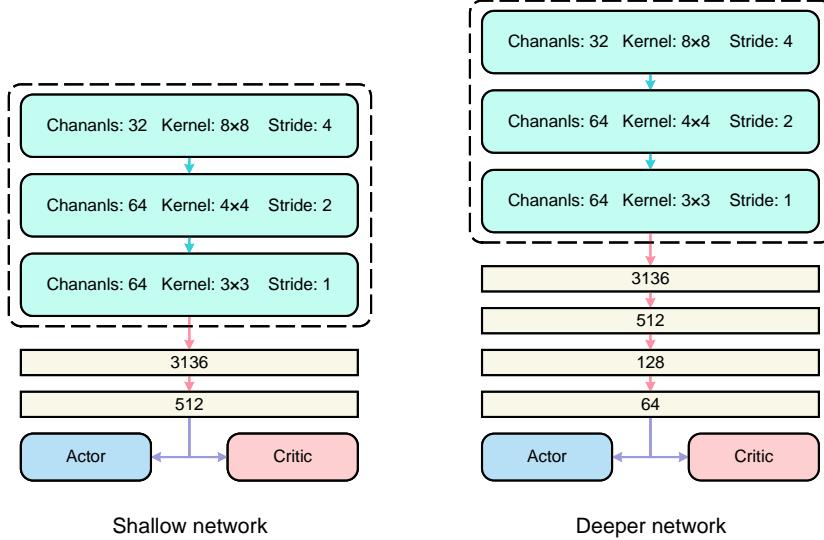


Figure 14. Shallow network and deeper network structure.