Training PPO-Clip with Parallelized Data Generation: A Case of Fixed-Point Convergence

Homayoun Honari[1], **Roger Creus Castanyer** [1,2], **Pablo Samuel Castro**[1,2,3], **Glen Berseth**[1,2]

{homayoun.honari,roger.creus-castanyer,glen.berseth}@mila.quebec, psc@google.com

[1]**Mila-Quebec AI Institute**
[2]**Université de Montréal**
[3]**Google DeepMind**

Abstract

In recent years, with the increase in the compute power of GPUs, parallelized data collection has become the dominant approach for training reinforcement learning (RL) agents. Proximal Policy Optimization (PPO) is one of the widely-used on-policy methods for training RL agents. In this paper, we focus on the training behavior of PPO-Clip with the increase in the number of parallel environments. In particular, we show that as we increase the amount of data used to train PPO-Clip, the optimized policy would converge to a fixed distribution. We use the results to study the behavior of PPO-Clip in two case studies: the effect of change in the minibatch size and the effect of increase in the number of parallel environments versus the increase in the rollout lengths. The experiments show that settings with high-return PPO runs result in slower convergence to the fixed-distribution and higher consecutive KL divergence changes. Our results aim to offer a better understanding for the prediction of the performance of PPO with the scaling of the parallel environments.

1 Introduction

Reinforcement Learning (RL) constitutes a fundamental paradigm in machine learning, enabling intelligent agents to interact with an environment and learn optimal behavior. In recent years, advances in computing power and parallel simulation have elevated distributed training to a central topic in machine learning research. As models grow larger and environments become more complex, model inference increasingly becomes the primary bottleneck, further motivating a shift toward distributed training. In distributed RL training, multiple actors generate trajectories in parallel, aggregating experiences for a central learner to update the policy. A key question is whether policy training still converges as the number of actors increases and rollouts become longer.

Parallelized data generation offers advantages such as enhanced exploration Gallici et al. (2024), faster data collection, and greater data diversity Kapturowski et al. (2018). However, there are inherent limitations to the performance gains obtainable simply by scaling up data generation; the nature and significance of these limitations depend on the specific RL formulation employed.

There are two main RL optimization paradigms: off-policy and on-policy methods. Generally, the off-policy methods rely on a replay buffer to approximate the stationary state visitation in the MDP. On the other hand, on-policy methods aim to optimize the policy using trajectories close to the current policy. As a result, in the context of scaling these methods, off-policy methods require relatively larger storage to store the transitions and have a good performance. On the other hand, on-policy methods offer simpler implementations and lower storage requirements at the cost of lower

data reusability. In other words, the data efficiency of on-policy methods suffer greatly when the data becomes too far from the optimized policy Huang et al. (2024); Espeholt et al. (2018) (which is characterized as the off-policiness of the data).

In the context of scaling, off-policy methods have been well-studied Schaul et al. (2015); Kapturowski et al. (2018); Rybkin et al. (2025); Badia et al. (2020); Kapturowski et al. (2022) since they provide a better framework to understand their scaling behavior. This is due to the fact that all transitions are stored in the replay buffer. However, scaling of on-policy algorithms pose a more challenging problem. The main reason is the fact that the real distributed setup of training these methods require certain assumptions regarding the environments and the distance between the behavior policies and the current learner policy (which is named as the policy lag Huang et al. (2022b)). Policy lag is generally caused when we perform multiple steps of optimization on the data causing the policy to be more and more distanced from the behaviour policy. As a result, while more gradient updates may improve the sample efficiency of the algorithms, it inevitably causes lag in later epochs leading to instabilities in the learning process. Policy lag is further aggravated when we consider the real-world distributed implementation issues such as delayed communications and asynchrony of the generated trajectories with respect to the current policy. This leads to discrepancies inside the batches of data causing even more instabilities.

This paper aims to provide a theoretical analysis on the scaling behavior of one of the most commonly-used on-policy methods proximal policy optimization (PPO). In particular, we show that when using PPO in continuous action environments with Gaussian distribution action parametrization, the increase in the number of actors generating trajectories results in the convergence to a fixed policy. Moreover, we study the performance of PPO with respect to two important parameters, namely minibatch size and rollout length vs number of parallel actors and examine them through the lens of the KL-divergence characteristic. The study shows that PPO would benefit from slower convergence to the fixed policy and higher consecutive KL-divergence values in terms of the overall performance.

2 Related Work

There has been a growing number of studies on the global convergence of RL methods in various settings. One of the well-studied topics in that regard is the convergence analysis of RL methods under Policy Gradient (PG) loss. For example, Yuan et al. (2020); Zhang et al. (2020; 2021); Wang et al. (2019); Xu et al. (2020) explored the conditions under which the policy can converge to a global solution. More importantly, Zhang et al. (2020); Fazel et al. (2018) aim to analyze the stationary point convergence of general PG policies. Moreover, PPO-Clip theoretical analysis has also been generally focused on its global convergence. To this end, Jin et al. (2023); Yao et al. (2022); Huang et al. (2021) are the major works focusing on the global convergence of methods under PPO-Clip loss. However, the major focus of the works have been on the theoretical global convergence of the policy in the training process. In contrast, our work aims to characterize the behaviour of PPO-Clip policies in the parallelized setting rather than their global convergence performance.

Regarding the parallelization of DeepRL algorithms, the focus has been mainly on improving their empirical performance. To this end, off-policy distributed methods Kapturowski et al. (2018); Schaul et al. (2015) mainly utilize a specialized replay buffer to allow and account for the asychrony of the data. For that purpose, Gallici et al. (2024) theoretically demonstrated that using LayerNorm can allow efficient and stable training of Deep Q-Networks Mnih et al. (2013) without using a target network and replay buffer allowing for better parallelization. Furthermore, in the context of on-policy methods, most attempts Zhang et al. (2019); Wijmans et al. (2019); Espeholt et al. (2018); Kapturowski et al. (2022) has been to increase the robustness of the method to off-policy data. For example, Espeholt et al. (2018) proposed v-trace which uses truncated importance sampling to estimate the value of the learner policy from the trajectories generated from a behaviour policy. Kapturowski et al. (2022) also added certain architectural enhancement, such as a trust region

method for training of the value networks and temporal difference-error normalization, to achieve better overall performance.

3 Preliminaries

3.1 MDP

A Markov Decision Process (MDP) is defined using the tuple $< \mathcal{S}, \mathcal{A}, T, r, \mu, \gamma >$ where $\mathcal{S}$ and $\mathcal{A}$ are the state and action space spaces, respectively, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition probability, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, μ is the initial state distribution, and γ is the discount factor.

The main objective of an RL policy is to maximize the expected discounted return $\eta(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. To this end, the state and state-action value functions are defined as conditioning them for the expected return:

$$V^{\pi}(s) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s] \tag{1}$$

$$Q^{\pi}(s, a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s, a_0 = a] \tag{2}$$

Kakade (2001) derived the performance difference between two policies as:

$$V^{\pi'}(s_0) - V^{\pi}(s_0) = \frac{1}{1-\gamma} \mathbb{E}_{\tau\sim\pi'}[A^{\pi}(s, a)] \tag{3}$$

where $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$ is the advantage function.

3.2 Proximal Policy Optimization (PPO)

One of the early successful attempts at optimizing the policy using Eq. 3 was the seminal paper Trust Region Policy Optimization (TRPO) Schulman et al. (2015a) which used a constrained surrogate objective specified as:

$$\max_{\theta\in\Theta} \mathbb{E}_{\tau\sim\pi_{\theta_{old}}}[\frac{\pi_\theta(s, a)}{\pi_{\theta_{old}}(s, a)} \hat{A}^{\pi_{\theta_{old}}}(s, a)] \tag{4}$$

$$subject\ to\ \mathbb{E}_{\tau\sim\pi_{\theta_{old}}}[D_{KL}[\pi_{\theta_{old}}(.|s)||\pi_\theta(.|s)]] \leq \delta \tag{5}$$

where $\pi_{\theta_{old}}$ is the policy used to generate the trajectories before updating the parameters. The main difference between Eq. 4 and Eq. 3 is the approximation that the trajectories are generated from the behavior policy $\pi_{\theta_{old}}$ instead of the current optimized policy π_θ.

The main challenge faced when optimizing the loss function in Eq. 4 is mainly regarding the importance sampling ratio $\frac{\pi_\theta(s,a)}{\pi_{\theta_{old}}(s,a)}$. With more and more gradient steps the ratio converges to zero or unboundedly grows. Hence, various algorithms aim to control the ratio so that the two policy distribution can be close to each other. In addition to that, to keep the aforementioned approximation accurate, the two distributions need to be close enough to each other in terms of the total variation divergence or the KL divergence Schulman et al. (2015a). For that reason, TRPO imposes the KL divergence constraint to the optimization process.

While TRPO provides an effective framework to optimize the policy, evaluating the KL divergence and maintaining the constraint is challenging. To address this issue, PPO Schulman et al. (2017) proposed a clipped surrogate objective to optimize the policy.

PPO objective is defined as:

$$\mathcal{L}_{CLIP}(\theta) = \mathbb{E}_{\tau\sim\pi_{\theta_{old}}}[\min(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\hat{A}^{\pi_{\theta_{old}}}(s,a), clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\theta_{old}}})] \quad (6)$$

where ϵ is the trust region hyperparameter. Basically, PPO maintains a trust region by removing the gradients with $\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ out of the $[1-\epsilon, 1+\epsilon]$ range.

4 Analysis

4.1 Problem Formulation

We frame the problem in as a distributed setting with a set of actors generating trajectories. Let us denote π_{θ_T} as the final updated policy which the actors use to generate trajectories. Furthermore, D_N denotes the trajectory buffer from N actors. D_N will be the primary source of data used to update π_{θ_T}. For that purpose, we denote $\pi^N_{\theta_{T+1}}$ as the next updated policy using the data from N actors. It is important to note that while in the practical implementation PPO uses a value function and Generalized Advantage Estimation (GAE) Schulman et al. (2015b) to estimate the advantage function, we avoid considering that in our analysis and assume that to be fixed (even though it might not be optimal).

Furthermore, the PPO policy parametrizes the action using a Gaussian distribution:

$$\pi(a|s) = \mathcal{N}(\mu_\theta(s), \sigma_\theta) = \frac{1}{\sigma_\theta\sqrt{2\pi}}\exp(-\frac{(a-\mu_\theta(s))}{2\sigma^2_\theta}) \quad (7)$$

It is important to note that the parametrization of the standard deviation σ is state-independent as a common practice to promote stability of the training process Huang et al. (2022b); Andrychowicz et al. (2021); Huang et al. (2022a).

Moreover, we make certain structural assumptions based on the practical implementations of PPO:

Assumption 1. *The policy parameters θ are ν-Lipschitz continuous w.r.t. to the outputs in the l_∞ norm. In other words, the first-order gradient of μ_θ and σ_θ w.r.t. θ are bounded: $\|\nabla_\theta\mu(s)\|_\infty, \|\nabla_\theta\sigma\|_\infty \leq \nu\ \forall s \in \mathcal{S}$*

It is important to note that in addition to the fact that Lipschitz continuity is a common and practical assumption for the neural networks using common activation functions (such as ReLU and Tanh which are 1-Lipschitz continuous themselves), the boundedness of the gradient allows us to formulate the clipping process of PPO better since in the clipped data points we would set gradient to zero. Hence, it allows us to analyze PPO with the unconstrained surrogate loss function 4.

Assumption 2. *The policy parameters are α-smooth w.r.t the outputs. In other words, the second-order gradient of the policy parameters θ w.r.t. the outputs are bounded: $|\nabla^2_\theta\mu(s)| \leq \alpha, |\nabla^2_\theta\sigma| = 0\ \forall s \in \mathcal{S}$*

It is noteworthy that the reason for assuming $\|\nabla^2\sigma_\theta\|_\infty = 0$ is due to assuming the standard deviation is state-independent.

Assumption 3. *The ratio $r(s,a) = \frac{A_\beta(s,a)}{\pi_{\theta_{old}}(a|s)}$ is bounded: $|r(s,a)| \leq R$.*

While the assumption may not be explicitly used in the standard implementations, we can see details that imply this fact. For example, the practice of normalizing the advantage estimation aims to bound it as much as possible.Moreover, while the ratio may still be unbounded due to $\pi_{\theta_{old}}(a|s)$ being too small, in many cases we filter out the outliers using the clipping in PPO.

4.2 Fixed Point Analysis

To prove the convergence of the policy to a fixed point, we first show that the PPO loss function is L-smooth:

Lemma 1. *Fixing the current policy parameters as θ (which may generally be different from the trajectory-generating policy parameters θ_{old}), the PPO-Clip loss function in Eq. 6 is locally L-smooth where* $L = \max(10\eta\nu^2\|\sigma\|_{-\infty}^{-2}|\mathcal{A}|, 2\eta R|\mathcal{A}|\left[\frac{\nu^2\sqrt{\pi}}{\sqrt{2}\|\sigma\|_{-\infty}^2} + \frac{\alpha}{\|\sigma\|_{-\infty}} + \frac{\nu^2}{\sqrt{2}\|\sigma\|_{-\infty}^2}\right])$.

The proofs to all the lemmas and theorems are included in Appendix A.

Furthermore, it is important to note that by generally assuming that the value of σ is lower bounded throughout the optimization process (for example by clipping its value), we can conclude that the loss function is globally L-smooth. Empirically, the lower bound can be specified as: $\sigma_{\min} = \min_{k=1,\dots,\infty}\{\sigma_k\}$ where σ_i is the result of the optimization step at iteration i.

Furthermore, the smoothness of the loss function allows us to use the classic ε-critical convergence of the parameters to a fixed point Agarwal et al. (2019):

Theorem 1. *Denote $\mathcal{L}_\beta^*$ as $\min_{\theta\in\Theta}\mathcal{L}_{CLIP}^\beta(\theta)$ with the trajectories generated from a fixed behavior policy $\beta(.|s)$ and the starting policy parameters as θ_0. By having a fixed step size η such that $\eta - \frac{\eta^2 L}{2} > 0$, for $\forall\varepsilon > 0$, we can achieve $\|\nabla_\theta\mathcal{L}_\beta(\theta)\|_2 \leq \varepsilon$ in at most $\frac{\mathcal{L}_\beta^* - \mathcal{L}_\beta(\theta_0)}{\varepsilon^2(\eta - \frac{\eta^2 L}{2})}$ iterations.*

Theorem 1 implies that on the limit, the policy parameters converge to a fixed policy. We denote this policy as π^∞ to indicate that with infinite number of actors and long enough updating iterations, it is achievable. It is noteworthy while Theorem 1 shows the convergence of the PPO loss function to a fixed distribution, it does not argue for the superiority of the performance of the final policy compared to the previous policies.

Next, we make the observation that by increasing the number of actors in the distributed setup, we gain a more accurate approximation of the gradient of Eq. 6. Hence, the policy must be closer to π^∞. Therefore, we explicitly assume the aforementioned convergence in terms of the KL divergence:

Assumption 4. *Denote the policy π_T used to generate trajectories at timestep T. The KL-divergence of the resultant policy π_{T+1}^N from N actors compared with the fixed policy distribution π_{T+1}^∞ is finite and $\gamma_D \in [0, 1)$ contraction with the increase in the number of actors $\forall s \in \mathcal{S}$:*

$$D_{KL}[\pi_{T+1}^\infty(s)||\pi_{T+1}^N(s)] \leq \gamma_D^N D_{KL}[\pi_{T+1}^\infty(s)||\pi_{T+1}^1(s)] < \infty \tag{8}$$

While Assumption 4 has been stated as a consequence of Theorem 1, the bottom rows of Fig. 2 and Fig. 3 aim to showcase the empirical evidence for Assumption 4 by assuming the final policy in the optimization iterations as π_{T+1}^∞ and comparing the previous policies with the final one.

Moreover, we can use Assumption 4 to analyze the sequential KL divergence between N-actor policies:

Theorem 2. *For any small enough $\varepsilon > 0$ there exists a number of actors n such that $\forall N > n$ and $\forall s \in \mathcal{S}$:*

$$D_{KL}[\pi_{T+1}^N(s)||\pi_{T+1}^n(s)] \leq \mathcal{O}(\gamma_D^{N-n}, \varepsilon) \tag{9}$$

Intuitively, Theorem 2 aims to show that with enough increase in the number of actors, the resultant policy would have fewer and fewer changes in its distribution as a result of more data.

5 Experiments

In this section we aim to study the behavior of PPO with various hyperparameters and parallelized data generation settings. To this end, we aim to answer two main question:

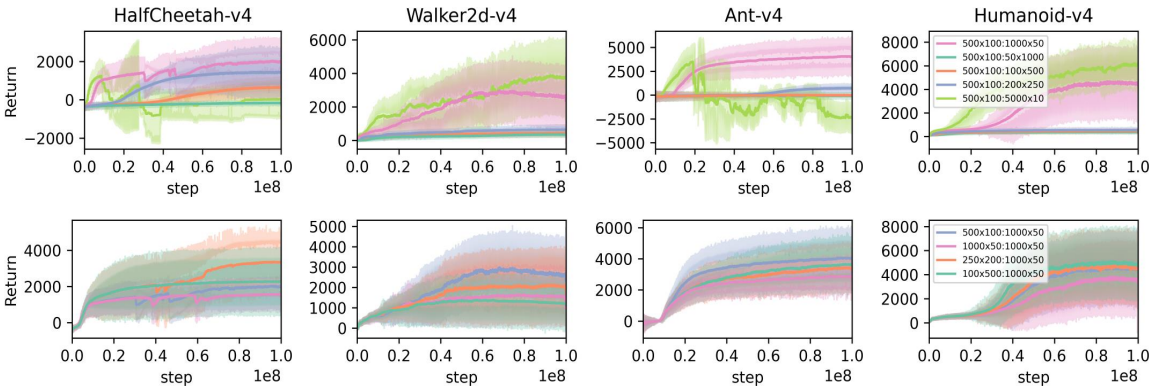

Figure 1: The labels in the legends show $(N_{env} \times N_{rollout} : N_{minibatch} \times minibatch\ size)$. The results represent the average across four independent seeds and the shaded areas showcase the standard deviation of the runs.
(Top Row) The effect of minibatch size on the performance of PPO.
(Bottom Row) The performance of PPO in various $(N_{env} \times N_{rollout})$ settings.

- Does the minibatch size affect the training performance of PPO?
- Does increasing the number of parallel actors have effect on the performance of PPO compared to the increase in the rollout length?

In addition to the study of PPO in terms of the overall return, this section aims to study these questions from the lens of Theorem 2 and Assumption 4. To this end, we utilize the CleanRL Huang et al. (2022b) codebase to adopt the continuous-action PPO implementation. Furthermore, we use four commonly-used MuJoCo environments, namely Humanoid, HalfCheetah, Ant, and Walker2d, from the Farama Gymnasium Towers et al. (2024). To have consistent results, the standard default parameters used in CleanRL were used. During the training phase, the agent would first interact with N_{env} parallel environments for $N_{rollout}$ steps each. After the data generation process, PPO is then updated by estimating the advantage function using Generalized Advantage Estimation method Schulman et al. (2015b) and then fixed-sized minibatches are sampled from the trajectories to train the policy. The policy optimization process of PPO can effectively and efficiently imitate the process of scaling. This is due to the fact that each policy in each updating iteration can be regarded as a policy from some lower number of actors and rollout lengths. Hence, we effectively regard the final policy in the series of updating iterations starting from π_T as π^{∞}_{T+1} which was defined in Section 4.1.

5.1 Effect of Change in the Minibatch Size

In this section we study the effect of minibatch size on the training of PPO. To do so, we set $N_{env} = 500$ parallelized environments with $N_{rollout} = 100$ rollout length. Furthermore, out of the $N_{env} \times N_{rollout}$ data, we split them into minibatches of various sizes.

As evident in Fig. 1, the size of the minibatches has a noticable effect on the overall performance of the PPO. While small minibatch size is a common practice originating back to the original PPO paper Schulman et al. (2017); Huang et al. (2022a) and was studied in Hilton et al. (2022); Andrychowicz et al. (2021), the minibatch size becomes an important factor in the scaling process of PPO. This is due to the fact that, while smaller minibatch sizes significantly affect the performance of the algorithm, they may also introduce instabilities in the training process. The experiments suggest that making the minibatches too small can introduce significant instability and result in the collapse of the algorithm in addition to the slow training process.

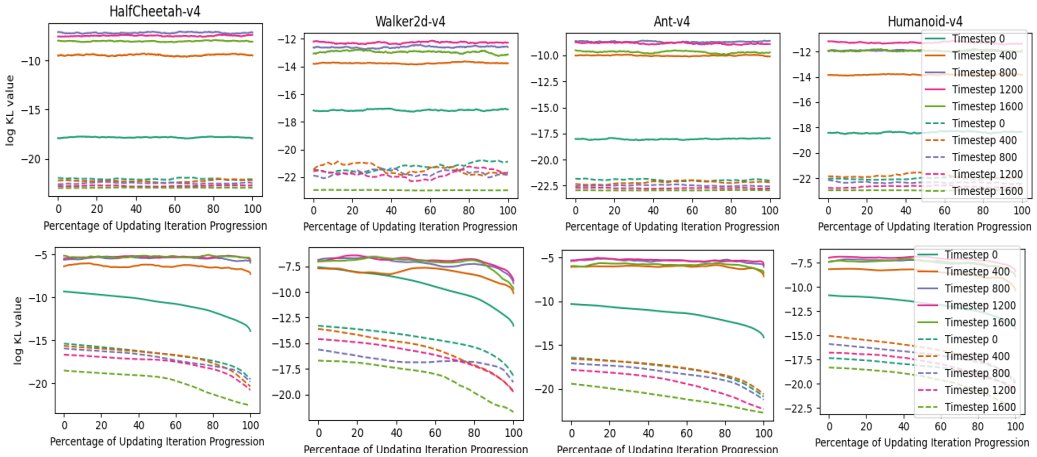

Figure 2: KL divergence values for the minibatch size experiments. **(Top Row)** Sequential log KL divergence values ($\mathbb{E}_{s\sim\pi_T^\infty}[D_{KL}[\pi_{T+1}^{i+1}\|\pi_{T+1}^{i}]]$) between policies in each optimization iteration. **(Bottom Row)** Reference log KL divergence values ($\mathbb{E}_{s\sim\pi_T^\infty}[D_{KL}[\pi_{T+1}^{\infty}\|\pi_{T+1}^{i}]]$) between the policies in each iteration and the final policy in the optimization iterations. The straight and dashed lines represent the values for $500 \times 100 : 1000 \times 50$ and $500 \times 100 : 50 \times 1000$ settings, respectively (Fig 1).

Furthermore, out of the multiple runs for the $500 \times 100 : 1000 \times 50$ and $500 \times 100 : 50 \times 1000$ settings, two of the best runs were selected to examine the KL profile of a high-return and low-return PPO training process in various training phase timesteps. To this end, since the number of policy training iterations varies for these settings, the x-axis is normalized as the percentage of the data processed by PPO (100% represents the achievement of the final policy). Fig. 2 represents KL divergence values of two of the settings.

The bottom row of Fig. 2 confirms the monotonic reduction in the KL values in Assumption 4. Moreover, the monotonicity can be also observe as a special case of Theorem 2 with fixed N and variable n. An important observation in the sequential KL values in Fig. 2 is the higher values of the sequential KL even though the minibatch sizes are smaller. Hence, it could be hypothesized that with same amount of data, smaller minibatches provide more meaningful changes to the policy.

5.2 Effect of Change in the Number of Parallel Environments vs Rollout Length

In this section, we study whether increasing the number of parallel environments have any benefits over increasing the rollout length. Bottom row of Fig. 1 illustrates the performance of PPO for different variations of $N_{env} \times N_{rollout}$. The results indicate that the increase in the rollout length can have similar effects in the performance compared with the increase in the number of parallel environments. Hence, the increase in the number of environments can provide the added benefit of CPU/GPU parallelization during the data collection phase resulting in a lower wall-clock time. Furthermore, Fig. 3 compares the KL divergence progression of PPO in 1000×50 and 100×500 settings. The results show similar KL divergence profiles across the two settings. in the context of updating the policy, the similarity could indicate that the exploration data as a result of the increase in the number of parallel environments are qualitatively similar to the increase in the rollout length.

6 Conclusions

Scaling RL methods to higher number of parallel environments is an important topic with the improvements in GPU computing power. In this paper the parallelization scaling properties of PPO was discussed. It was first shown that, with certain assumptions, training the policy with the PPO-Clip

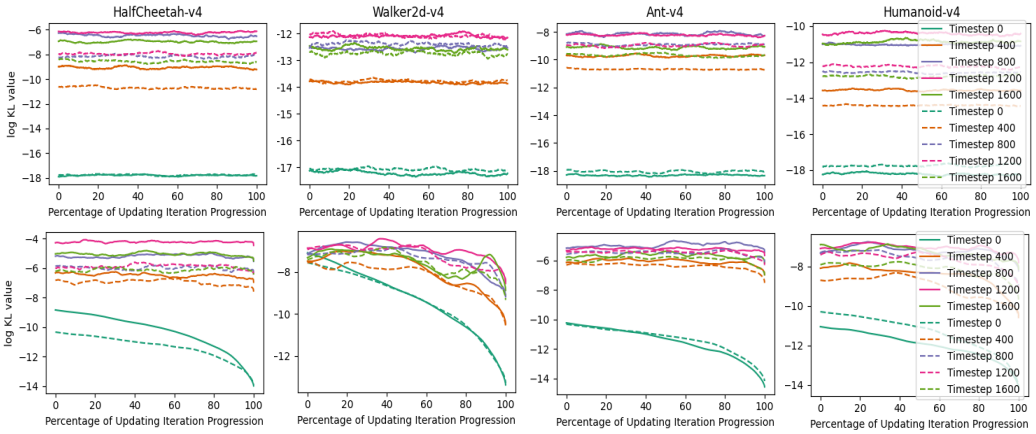

Figure 3: KL divergence values for the $N_{env} \times N_{rollout}$ experiments. **(Top Row)** Sequential log KL divergence values ($\mathbb{E}_{s\sim\pi_T^\infty}[D_{KL}[\pi_{T+1}^{i+1} \| \pi_{T+1}^{i}]]$) between policies in each optimization iteration. **(Bottom Row)** Reference log KL divergence values ($\mathbb{E}_{s\sim\pi_{T-1}^\infty}[D_{KL}[\pi_{T}^{\infty} \| \pi_{T}^{i}]]$) between the policies in each iteration and the final policy in the optimization iterations. The straight and dashed lines represent the values for 1000×50 and 100×500 settings, respectively (Fig 1).

loss will converge to a stationary distribution. Furthermore, the experiments with various number of minibatch sizes and different variations of the number of environments and the rollout lengths showed that faster convergence of the agent to the stationary distribution results in the degradation of the performance of the agent. In other words, the KL profiling in this paper offers a tradeoff between stability and performance. Having lower sequential KL divergence can result in more stability at the cost of performance. Hence, the sequential comparison of the KL divergence of the policies can offer an approach to monitor the stabillity and the optimality of training PPO-Clip as the available data is increased.

References

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32:96, 2019.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International conference on learning representations*, 2021.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pp. 507–517. PMLR, 2020.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International conference on machine learning*, pp. 1467–1476. PMLR, 2018.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098, 2022.

Nai-Chieh Huang, Ping-Chun Hsieh, Kuo-Hao Ho, Hsuan-Yu Yao, Kai-Chun Hu, Liang-Chun Ouyang, I Wu, et al. Neural ppo-clip attains global optimality: A hinge loss perspective. *arXiv preprint arXiv:2110.13799*, 2021.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*, 2022a.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022b.

Shengyi Huang, Jiayi Weng, Rujikorn Charakorn, Min Lin, Zhongwen Xu, and Santiago Ontanon. Cleanba: A reproducible and efficient distributed reinforcement learning platform. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Diq6urt3lS.

Ruinan Jin, Shuai Li, and Baoxiang Wang. On stationary point convergence of ppo-clip. In *The Twelfth International Conference on Learning Representations*, 2023.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Steven Kapturowski, Víctor Campos, Ray Jiang, Nemanja Rakićević, Hado van Hasselt, Charles Blundell, and Adria Puigdomenech Badia. Human-level atari 200x faster. *arXiv preprint arXiv:2209.07550*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Oleh Rybkin, Michal Nauman, Preston Fu, Charlie Snell, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Value-based deep rl scales predictably. *arXiv preprint arXiv:2502.04327*, 2025.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural policy gradient methods: Global optimality and rates of convergence. *arXiv preprint arXiv:1909.01150*, 2019.

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

Pan Xu, Felicia Gao, and Quanquan Gu. An improved convergence analysis of stochastic variance-reduced policy gradient. In *Uncertainty in Artificial Intelligence*, pp. 541–551. PMLR, 2020.

Hsuan-Yu Yao, Ping-Chun Hsieh, Kuo-Hao Ho, Kai-Chun Hu, Liang Chun Ouyang, and I-Chen Wu. Hinge policy optimization: Rethinking policy improvement and reinterpreting PPO, 2022. URL https://openreview.net/forum?id=gex-2G2bLdh.

Huizhuo Yuan, Xiangru Lian, Ji Liu, and Yuren Zhou. Stochastic recursive momentum for policy gradient methods. *arXiv preprint arXiv:2003.04302*, 2020.

Junyu Zhang, Chengzhuo Ni, Csaba Szepesvari, Mengdi Wang, et al. On the convergence and sample efficiency of variance-reduced policy gradient method. *Advances in Neural Information Processing Systems*, 34:2228–2240, 2021.

Kaiqing Zhang, Alec Koppel, Hao Zhu, and Tamer Basar. Global convergence of policy gradient methods to (almost) locally optimal policies. *SIAM Journal on Control and Optimization*, 58(6): 3586–3612, 2020.

Yufeng Zhang, Jialu Pan, Li Ken Li, Wanwei Liu, Zhenbang Chen, Xinwang Liu, and Ji Wang. On the properties of kullback-leibler divergence between multivariate gaussian distributions. *Advances in Neural Information Processing Systems*, 36:58152–58165, 2023.

Zhenyu Zhang, Xiangfeng Luo, Tong Liu, Shaorong Xie, Jianshu Wang, Wei Wang, Yang Li, and Yan Peng. Proximal policy optimization with mixed distributed training. In *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)*, pp. 1452–1456. IEEE, 2019.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Proofs

A.1 Proof to Lemma 1

As mentioned in Section 4.1, using the assumption of the Lipschitz continuity of the policy parameters (Assumption 1) we can ignore the clipping mechanism of PPO. Hence, to perform the analysis general in a general form, we consider the base policy loss function as $\mathcal{L}(\theta) = \mathbb{E}_{s\sim\beta}[\frac{\pi_\theta(.|s)}{\beta(.|s)}A_\beta(s,.)]$ where β is the behaviour policy used to generate the trajectories.

Therefore, we would like to maximize the loss function using the loss function parameterized by a Gaussian action distribution. Hence, the policy parameters are updated as:

$$\theta_{t+1} \leftarrow \theta_t + \eta\mathbb{E}_{s\sim\beta}[\frac{\nabla_\theta\pi_\theta(.|s)}{\beta(.|s)}A_\beta(s,.)] \tag{10}$$

$$= \theta_t + \eta\int_{s\in\mathcal{S}}\rho_\beta(s)\nabla_\theta\int_{-\infty}^{+\infty}\frac{r(s,a)}{\sigma(s)}\exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{11}$$

where $\eta = lr/\sqrt{2\pi}$ is the learning rate, $r(s,a) = A_\beta(s,a)/\beta(a|s)$, and $\rho_\beta(s)$ is the normalized state visitation distribution following the behaviour policy $\beta(a|s)$.

In the following , we assume the parameters of μ and σ are independent of each other. Hence, we analyze each of them separately.

Action mean value μ_θ

Let us expand the loss function assuming that σ is constant. The update rule follows (we annotate $b = -1/2\sigma^2(s)$):

$$\theta_{t+1} \leftarrow \theta_t + \eta\int_{s\in\mathcal{S}}\rho_\beta(s)\frac{1}{\sigma(s)}\int_{-\infty}^{+\infty}2b.r(s,a)(\mu-a)\nabla_\theta\mu\exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{12}$$

Hence, for the second-order gradient $\nabla^2_\theta\mathcal{L}(\theta)$ we have:

$$\begin{aligned}&\nabla^2\mathcal{L}(\theta)\\ &= \int_{s\in\mathcal{S}}\rho_\beta(s)\frac{1}{\sigma(s)}\int_{-\infty}^{+\infty}2b.r(s,a)[(\nabla_\theta\mu)^2 + (\mu-a)\nabla^2_\theta\mu + 2b(\nabla\mu)^2(\mu-a)^2]\\ &\qquad\qquad \exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads\end{aligned} \tag{13}$$

The bound can then be derived as:

$$|\nabla^2_\theta \mathcal{L}(\theta)| \leq 2\int_{s\in\mathcal{S}} \rho_\beta(s)\frac{|b|}{\sigma(s)}\int_{-\infty}^{+\infty} |r(s,a)|.|(\nabla_\theta\mu)^2 + (\mu-a)\nabla^2_\theta\mu + 2b(\nabla\mu)^2(\mu-a)^2|\exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{14}$$

$$\leq 2R\int_{s\in\mathcal{S}} \rho_\beta(s)\frac{|b|}{\sigma(s)}\int_{-\infty}^{+\infty} [\nu^2 + \alpha|\mu-a| + 2\nu^2|b|(\mu-a)^2] \exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{15}$$

$$= 2R\int_{s\in\mathcal{S}} \rho_\beta(s)\frac{|b|}{\sigma(s)}\left[\frac{\nu^2\sqrt{\pi}}{\sqrt{|b|}} + \frac{\alpha}{|b|} + \frac{\nu^2\sqrt{\pi}}{\sqrt{|b|}}\right] ds \tag{16}$$

$$= 2R\int_{s\in\mathcal{S}} \rho_\beta(s)\left[\frac{\nu^2\sqrt{\pi}}{\sqrt{2}\sigma^2} + \frac{\alpha}{\sigma} + \frac{\nu^2}{\sqrt{2}\sigma^2}\right] \tag{17}$$

$$= 2R\left[\frac{\nu^2\sqrt{\pi}}{\sqrt{2}\sigma^2} + \frac{\alpha}{\sigma} + \frac{\nu^2}{\sqrt{2}\sigma^2}\right] \tag{18}$$

Since the bound is element-wise applied to the absolute value of the second-order gradient, we can then apply the same bound to the l_∞ norm. Therefore, the Lipschitz constant of $\nabla_\theta\mathcal{L}$ w.r.t. the parameters used to output μ_θ will be:

$$L_\mu = 2R\left[\frac{\nu^2\sqrt{\pi}}{\sqrt{2}\sigma^2} + \frac{\alpha}{\sigma} + \frac{\nu^2}{\sqrt{2}\sigma^2}\right] \tag{19}$$

Action standard deviation value σ_θ

Using the same definition of Gradient ascent we have:

$$\theta_{t+1} \leftarrow \theta_t + \eta\int_{s\in\mathcal{S}} \rho_\beta(s)\int_{-\infty}^{+\infty} \frac{[(a-\mu)^2-\sigma^2]\nabla\sigma}{\sqrt{2\pi}\sigma^4}\exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{20}$$

Therefore, for $\nabla^2\mathcal{L}(\theta)$ we have:

$$\nabla^2\mathcal{L}(\theta) \tag{21}$$

$$= \int_{s\in\mathcal{S}} \rho_\beta(s)\int_{-\infty}^{+\infty} \frac{(\sigma^5-(a-\mu)^2\sigma^3)\nabla^2\sigma - (2\sigma^4+(a-\mu)^4-5(a-\mu)^2\sigma^2)(\nabla\sigma)^2}{\sqrt{2\pi}\sigma^7} \exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{22}$$

Bounding the gradient we have:

$$|\nabla^2 f(\theta)| \leq \int_{s\in\mathcal{S}} \rho_\beta(s) \int_{-\infty}^{+\infty} \frac{|2\sigma^4 + (a-\mu)^4 - 5(a-\mu)^2\sigma^2|\nu^2}{\sqrt{2\pi}\sigma^7} \exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{23}$$

$$\leq \int_{s\in\mathcal{S}} \rho_\beta(s) \int_{-\infty}^{+\infty} \frac{(2\sigma^4 + (a-\mu)^4 + 5(a-\mu)^2\sigma^2)\nu^2}{\sqrt{2\pi}\sigma^7} \exp(-\frac{(\mu(s)-a)^2}{2\sigma^2(s)})dads \tag{24}$$

$$= \int_{s\in\mathcal{S}} \rho_\beta(s)[2\sigma^{-2} + 3\sigma^{-2} + 5\sigma^{-2}]\nu^2 ds \tag{25}$$

$$= 10\nu^2\sigma^{-2} \tag{26}$$

where, again, we used the assumption that σ is state-independent. Therefore, the Lipschitz smoothness constant will be:

$$L_\sigma = 10\nu^2\sigma^{-2} \tag{27}$$

Finally, since μ and σ are updated together, $\mathcal{L}(\theta)$ is locally L-smooth with the constant:

$$L = \max(10\nu^2\sigma^{-2}, 2R\left[\frac{\nu^2\sqrt{\pi}}{\sqrt{2}\sigma^2} + \frac{\alpha}{\sigma} + \frac{\nu^2}{\sqrt{2}\sigma^2}\right]) \tag{28}$$

A.1.1 Multi-Dimensional Action Spaces

It is important to note that the cases considered in the previous sections involved action spaces of size one $|\mathcal{A}| = 1$. While in general case extending that to vector outputs requires analyzing the Jacobian of θ, we can avoid that in this setting. Considering that the final loss function is $\mathcal{L}(\theta) = \sum_{i=1}^{|\mathcal{A}|} \mathcal{L}_i(\theta)$ where i is the index of the action, we can write the operation as:

$$\theta + \eta\nabla f(\theta) = \theta + \eta \sum_{i=1}^{|\mathcal{A}|} \nabla f_i(\theta) \tag{29}$$

Hence, for the second-order gradients we have:

$$\nabla^2 f(\theta) = \sum_{i=1}^{|\mathcal{A}|} \nabla^2 f_i(\theta) \tag{30}$$

$$\Rightarrow |\nabla^2 f(\theta)| \leq |\mathcal{A}| \max_{i=1,\ldots,|\mathcal{A}|} (|\nabla^2 f_i|)(\theta) \tag{31}$$

Incorporating the bound onto μ and σ, we have:

$$\max(10\eta\nu^2\|\sigma\|_{-\infty}^{-2}|\mathcal{A}|, 2\eta R|\mathcal{A}|\left[\frac{\nu^2\sqrt{\pi}}{\sqrt{2}\|\sigma\|_{-\infty}^2} + \frac{\alpha}{\|\sigma\|_{-\infty}} + \frac{\nu^2}{\sqrt{2}\|\sigma\|_{-\infty}^2}\right]) \tag{32}$$

A.2 Proof of Theorem 1

The L-smoothness of the loss function has the following property for $\forall x, y \in \mathbb{R}^n$:

$$|\mathcal{L}(y) - (\mathcal{L}(x) + \nabla\mathcal{L}(x)^T(y-x))| \leq \frac{L}{2}\|x-y\|_2^2 \tag{33}$$

Hence, for consecutive parameters following the gradient ascent setup we have:

$$|\mathcal{L}(\theta_{k+1}) - (\mathcal{L}(\theta_k) + \eta_k\|\nabla_\theta\mathcal{L}(\theta_k)\|_2^2)| \leq \frac{\eta_k^2 L}{2}\|\nabla_\theta\mathcal{L}(\theta_k)\|_2^2 \tag{34}$$

where η_k is the learning rate at iteration k and the two parameters follow the gradient ascent setup $\theta_{k+1} \leftarrow \theta_k + \eta_k \nabla_\theta \mathcal{L}(\theta_k)$.

Hence, we have:

$$\mathcal{L}(\theta_k) + (\eta_k - \frac{\eta_k^2 L}{2}) \|\nabla_\theta \mathcal{L}(\theta_k)\|_2^2 \leq \mathcal{L}(\theta_{k+1}) \tag{35}$$

Therefore, in k gradient steps we have:

$$\sum_{i=0}^{k-1} (\eta_i - \frac{\eta_i^2 L}{2}) \|\nabla_\theta \mathcal{L}(\theta_i)\|_2^2 \leq \mathcal{L}(\theta_k) - \mathcal{L}(\theta_0) \leq \mathcal{L}^* - \mathcal{L}(\theta_0) \tag{36}$$

Hence, by assuming a fixed step size $\eta_k = \eta$ such that $\eta - \frac{\eta^2 L}{2} > 0$ we have:

$$\frac{1}{k} \sum_{i=0}^{k-1} \|\nabla_\theta \mathcal{L}(\theta_i)\|_2^2 \leq \frac{\mathcal{L}^* - \mathcal{L}(\theta_0)}{k(\eta - \frac{\eta^2 L}{2})} \tag{37}$$

Therefore, there exists a gradient $\|\nabla_\theta \mathcal{L}(\theta_i)\|_2^2 \leq \frac{\mathcal{L}^* - \mathcal{L}(\theta_0)}{k(\eta - \frac{\eta^2 L}{2})}$ and, in order to have $\|\nabla_\theta \mathcal{L}(\theta_i)\|_2 \leq \varepsilon$, we would need at least $\frac{\mathcal{L}^* - \mathcal{L}(\theta_0)}{\varepsilon^2(\eta - \frac{\eta^2 L}{2})}$ iterations.

A.3 Proof of Theorem 2

Assumption 4 allows us to conclude that within the series $\varepsilon_1, \varepsilon_2, \varepsilon_3, \ldots$ (where for ε_i we have $\max_{s \in \mathcal{S}} D_{\text{KL}}[\pi^{\infty}_{T+1}(s) || \pi^{i}_{T+1}(s)] < \varepsilon_i$) there exists a number of actors n such that $D_{\text{KL}}[\pi^{\infty}_{T+1}(s) || \pi^{n}_{T+1}(s)] < \varepsilon_n \leq \varepsilon \; \forall s \in \mathcal{S}$. Hence following that, for any $N > n$ we have that:

$$D_{\text{KL}}[\pi^{\infty}_{T+1}(s) || \pi^{N}_{T+1}(s)] \leq \gamma_D^{N-n} \varepsilon \tag{38}$$

Using the results from Zhang et al. (2023), since ϵ is small enough, the reverse KL for multivariate Gaussian distributions can be bounded as (annotating $n' = N - n$):

$$D_{\text{KL}}[\pi^{N}_{T+1}(s) || \pi^{\infty}_{T+1}(s)] \leq \varepsilon' = \gamma_D^{n'} \varepsilon + 2\gamma_D^{1.5n'} \varepsilon^{1.5} + \mathcal{O}(\gamma_D^{2n'} \varepsilon^2) \tag{39}$$

Using the relaxed triangle inequality of KL divergence, we have:

$$D_{\text{KL}}[\pi^{N}_{T+1}(s) || \pi^{n}_{T+1}(s)] \leq 3\varepsilon + 3\varepsilon' + 2\sqrt{\varepsilon\varepsilon'} + o(\varepsilon) + o(\varepsilon') \leq \mathcal{O}(\gamma_D^{N-n}, \varepsilon) \tag{40}$$

Therefore, with the decrease ε and the increase in the reference number of actors n, the KL divergence gets closer to zero for the subsequent N actors.