Divergence-Augmented Policy Optimization

Qing Wang *
Huya AI
Guangzhou, China

Yingru Li
The Chinese University of Hong Kong
Shenzhen, China

Jiechao Xiong
Tencent AI Lab
Shenzhen, China

Tong Zhang
The Hong Kong University of Science and Technology
Hong Kong, China

Abstract

In deep reinforcement learning, policy optimization methods need to deal with issues such as function approximation and the reuse of off-policy data. Standard policy gradient methods do not handle off-policy data well, leading to premature convergence and instability. This paper introduces a method to stabilize policy optimization when off-policy data are reused. The idea is to include a Bregman divergence between the behavior policy that generates the data and the current policy to ensure small and safe policy updates with off-policy data. The Bregman divergence is calculated between the state distributions of two policies, instead of only on the action probabilities, leading to a divergence augmentation formulation. Empirical experiments on Atari games show that in the data-scarce scenario where the reuse of off-policy data becomes necessary, our method can achieve better performance than other state-of-the-art deep reinforcement learning algorithms.

1 Introduction

In recent years, many algorithms based on policy optimization have been proposed for deep reinforcement learning (DRL), leading to great successes in Go, video games, and robotics (Silver et al., 2016; Mnih et al., 2016; Schulman et al., 2015, 2017b). Real-world applications of policy-based methods commonly involve function approximation and data reuse. Typically, the reused data are generated with an earlier version of the policy, leading to off-policy learning. It is known that these issues may cause premature convergence and instability for policy gradient methods (Sutton et al., 2000; Sutton and Barto, 2017).

A standard technique that allows policy optimization methods to handle off-policy data is to use importance sampling to correct trajectories from the behavior policy that generates the data to the target policy (e.g. Retrace (Munos et al., 2016) and V-trace (Espeholt et al., 2018)). The efficiency of these methods depends on the divergence between the behavior policy and the target policy. Moreover, to improve stability of training, one may introduce a regularization term (e.g. Shannon-Gibbs entropy in (Mnih et al., 2016)), or use a proximal objective of the original policy gradient loss (e.g. clipping in (Schulman et al., 2017b; Wang et al., 2016a)). Although the well-adopted method of entropy regularization can stabilize the optimization process (Mnih et al., 2016), this additional entropy regularization alters the learning objective, and prevent the algorithm from converging to the optimal action for each state. Even for the simple case of bandit problems, the monotonic diminishing regularization may fail to converge to the best arm (Cesa-Bianchi et al., 2017).

In this work, we propose a method for policy optimization by adding a Bregman divergence term, which leads to more stable and sample efficient off-policy learning. The Bregman divergence

*The work was done when the first author was at Tencent AI Lab.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

constraint is widely used to explore and exploit optimally in mirror descent methods (Nemirovsky and Yudin, 1983), in which specific form of divergence can attain the optimal rate of regret (sample efficiency) for bandit problems (Audibert et al., 2011; Bubeck and Cesa-Bianchi, 2012). In contrast to the traditional approach of constraining the divergence between target policy and behavior policy conditioned on each state (Schulman et al., 2015), we consider the divergence over the joint state-action space. We show that the policy optimization problem with Bregman divergence on state-action space is equivalent to the standard policy gradient method with divergence-augmented advantage. Under this view, the divergence-augmented policy optimization method not only considers the divergence on the current state but also takes into account the discrepancy of policies on future states, thus can provide a better constraint on the change of policy and encourage "deeper" exploration.

We experiment with the proposed method on the commonly used Atari 2600 environment from Arcade Learning Environment (ALE) (Bellemare et al., 2013). Empirical results show that divergence-augmented policy optimization method performs better than the state-of-the-art algorithm under data-scarce scenarios, i.e., when the sample generating speed is limited and samples in replay memory are reused multiple times. We also conduct a comparative study for the major effect of improvement on these games.

The article is organized as follows: we give the basic background and notations in Section 2. The main method of divergence-augmented policy optimization is presented in Section 3, with connections to previous works discussed in Section 4. Empirical results and studies can be found in Section 5. We conclude this work with a short discussion in Section 6.

2 Preliminaries

In this section, we state the basic definition of the Markov decision process considered in this work, as well as the Bregman divergence used in the following discussions.

2.1 Markov Decision Process

We consider a Markov decision process (MDP) with infinite-horizon and discounted reward, denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, d_0, \gamma)$, where $\mathcal{S}$ is the finite state space, $\mathcal{A}$ is the finite action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, where $\Delta(\mathcal{S})$ means the space of all probability distributions on $\mathcal{S}$. A reward function is denoted by $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The distribution of initial state s_0 is denoted by $d_0 \in \Delta(\mathcal{S})$. And a discount factor is denoted by $\gamma \in (0, 1)$.

A stochastic policy is denoted by $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The space of all policies is denoted by Π. We use the following standard notation of state-value $V^\pi(s_t)$, action-value $Q^\pi(s_t, a_t)$ and advantage $A^\pi(s_t, a_t)$, defined as $V^\pi(s_t) = \mathbb{E}_{\pi|s_t} \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})$, $Q^\pi(s_t, a_t) = \mathbb{E}_{\pi|s_t,a_t} \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})$, and $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$, where $\mathbb{E}_{\pi|s_t}$ means $a_l \sim \pi(a|s_l)$, $s_{l+1} \sim P(s_{l+1}|s_l, a_l)$, $\forall l \geq t$, and $\mathbb{E}_{\pi|s_t,a_t}$ means $s_{l+1} \sim P(s_{l+1}|s_l, a_l)$, $a_{l+1} \sim \pi(a|s_{l+1})$, $\forall l \geq t$. We also define the space of policy-induced state-action distributions under $\mathcal{M}$ as

$$\Delta_\Pi = \{\mu \in \Delta(\mathcal{S} \times \mathcal{A}) : \sum_{a'} \mu(s', a') = (1 - \gamma) d_0(s') + \gamma \sum_{s,a} P(s'|s, a)\mu(s, a), \forall s' \in \mathcal{S}\} \quad (1)$$

We use the notation μ_π for the state-action distribution induced by π. On the other hand, for each $\mu \in \Delta_\Pi$, there also exists a unique policy $\pi_\mu(a|s) = \frac{\mu(s,a)}{\sum_b \mu(s,b)}$ which induces μ. We define the state distribution d_π as $d_\pi(s) = (1 - \gamma)\mathbb{E}_{\tau|\pi} \sum_{t=0}^{\infty} \gamma^t \mathbf{1}(s_t = s)$. Then we have $\mu_\pi(s, a) = d_\pi(s)\pi(a|s)$. We sometimes write π_{μ_t} as π_t and d_{π_t} as d_t when there is no ambiguity.

In this paper, we mainly focus on the performance of a policy π defined as

$$J(\pi) = (1 - \gamma)\mathbb{E}_{\tau|\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) = \mathbb{E}_{d_\pi,\pi} r(s, a) \quad (2)$$

where $\mathbb{E}_{\tau|\pi}$ means $s_0 \sim d_0, a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t), t \geq 0$. We use the notation $\mathbb{E}_{d,\pi} = \mathbb{E}_{s\sim d(\cdot),a\sim\pi(\cdot|s)}$ for brevity.

2.2 Bregman Divergence

We define Bregman divergence (Bregman, 1967) as follows (e.g. Definition 5.3 in (Bubeck and Cesa-Bianchi, 2012)). For $\mathcal{D} \subset \mathbb{R}^d$ an open convex set, the closure of $\mathcal{D}$ as $\bar{\mathcal{D}}$, we consider a *Legendre* function $F : \bar{\mathcal{D}} \to \mathbb{R}$ defined as (1) F is strictly convex and admits continuous first partial derivatives on $\mathcal{D}$, and (2) $\lim_{x \to \bar{\mathcal{D}} \setminus \mathcal{D}} \|\nabla F\| = +\infty$. For function F, we define the Bregman divergence $D_F : \bar{\mathcal{D}} \times \mathcal{D} \to \mathbb{R}$ as

$$D_F(x, y) = F(x) - F(y) - \langle \nabla F(y), x - y \rangle.$$

The inner product is defined as $\langle x, y \rangle = \sum_i x_i y_i$. For $\mathcal{K} \subset \bar{\mathcal{D}}$ and $\mathcal{K} \cap \mathcal{D} \neq \emptyset$, the Bregman projection

$$z = \arg\min_{x \in \mathcal{K}} D_F(x, y)$$

exists uniquely for all $y \in \mathcal{D}$. Specifically, for $F(x) = \sum_i x_i \log(x_i) - \sum_i x_i$, we recover the Kullback-Leibler (KL) divergence as

$$D_{\mathrm{KL}}(\mu', \mu) \quad = \quad \sum_{s,a} \mu'(s, a) \log \frac{\mu'(s, a)}{\mu(s, a)}$$

for $\mu, \mu' \in \Delta(\mathcal{S} \times \mathcal{A})$ and $\pi, \pi' \in \Pi$. To measure the distance between two policies π and π', we also use the symbol for conditional "Bregman divergence"[2] associated with state distribution d denoted as

$$D_F^d(\pi', \pi) = \sum_s d(s) D_F(\pi'(\cdot|s), \pi(\cdot|s)). \tag{3}$$

3 Method

In this section, we present the proposed method from the motivation of mirror descent and then discuss the parametrization and off-policy correction we employed in the practical learning algorithm.

3.1 Policy Optimization and Mirror Descent

The mirror descent (MD) method (Nemirovsky and Yudin, 1983) is a central topic in the optimization and online learning research literature. As a first-order method for optimization, the mirror descent method can recover several interesting algorithms discovered previously (Sutton et al., 2000; Kakade, 2002; Peters et al., 2010; Schulman et al., 2015). On the other hand, as an online learning method, the online (stochastic) mirror descent method can achieve (near-)optimal sample efficiency for a wide range of problems (Audibert and Bubeck, 2009; Audibert et al., 2011; Zimin and Neu, 2013). In this work, following a series of previous works (Zimin and Neu, 2013; Neu et al., 2017), we investigate the (online) mirror descent method for policy optimization. We denote the state-action distribution at iteration t as μ_t, and $\ell_t(\mu) = \langle g_t, \mu \rangle$ as the linear loss function for μ at iteration t. Without otherwise noted, we consider the negative reward as the loss objective $\ell_t(\mu) = -\langle r, \mu \rangle$, which also corresponds to the policy performance $\ell_t(\mu) \equiv -J(\pi_\mu)$ by Formula (2). We consider the mirror map method associated with Legendre function F as

$$\nabla F(\tilde{\mu}_{t+1}) \quad = \quad \nabla F(\mu_t) - \eta g_t \tag{4}$$

$$\mu_{t+1} \quad \in \quad \Pi_{\Delta_\Pi}(\tilde{\mu}_{t+1}), \tag{5}$$

where $\tilde{\mu}_{t+1} \in \Delta(\mathcal{S} \times \mathcal{A})$ and $g_t = \nabla \ell_t(\mu_t)$. It is well-known (Beck and Teboulle, 2003) that an equivalent formulation of mirror map (4) is

$$\mu_{t+1} \quad = \quad \arg\min_{\mu \in \Delta_\Pi} D_F(\mu, \tilde{\mu}_{t+1}) \tag{6}$$

$$\quad = \quad \arg\min_{\mu \in \Delta_\Pi} D_F(\mu, \mu_t) + \eta \langle g_t, \mu \rangle, \tag{7}$$

The former formulation (6) takes the view of non-linear sub-gradient projection in convex optimization, while the later formulation (7) can be interpreted as a regularized optimization and is the usual definition of mirror descent (Nemirovsky and Yudin, 1983; Beck and Teboulle, 2003; Bubeck, 2015). In this work, we will mostly investigate the approximate algorithm in the later formulation (7).

[2]Note that D_F^d may not be a Bregman divergence.

3.2 Parametric Policy-based Algorithm

In the mirror descent view for policy optimization on state-action space as in Formula (7), we need to compute the projection of μ onto the space of Δ_Π. For the special case of KL-divergence on μ, the sub-problem of finding minimum in (7) can be done efficiently, assuming the knowledge of transition function P (See Proposition 1 in (Zimin and Neu, 2013)). However, for a general divergence and real-world problems with unknown transition matrices, the projection in (7) is non-trivial to implement. In this section, we consider direct optimization in the (parametric) policy space without explicit projection. Specifically, we consider μ_π as a function of π, and π parametrized as π_θ. The Formula (7) can be written as

$$\pi_{t+1} = \arg\min_{\pi} D_F(\mu_\pi, \mu_t) + \eta\langle g_t, \mu_\pi\rangle. \tag{8}$$

Instead of solving globally, we approximate Formula (8) with gradient descent on π. From the celebrated policy gradient theorem (Sutton et al., 2000), we have the following lemma:

Lemma 1. *(Policy Gradient Theorem (Sutton et al., 2000)) For d_π and μ_π defined previously, the following equation holds for any state-action function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:*

$$\sum_{s,a} f(s,a)\nabla_\theta \mu_\pi(s,a) = \sum_{s,a} d_\pi(s)\mathcal{Q}^\pi(f)(s,a)\nabla_\theta \pi(a|s),$$

where $\mathcal{Q}^\pi$ is defined as an operator such that

$$\mathcal{Q}^\pi(f)(s,a) = \mathbb{E}_{\pi|s_t=s,a_t=a} \sum_{l=0}^{\infty} \gamma^l f(s_{t+l}, a_{t+l}).$$

Decomposing the loss and divergence in two parts (8), we have

$$\nabla_\theta \left\langle g_t, \mu_\pi \right\rangle = \left\langle d_\pi \mathcal{Q}^\pi(g_t), \nabla_\theta \pi(a|s) \right\rangle, \tag{9}$$

which is the usual policy gradient, and

$$\nabla_\theta D_F(\mu_\pi, \mu_t) = \langle \nabla F(\mu_\pi) - \nabla F(\mu_t), \nabla_\theta \mu_\pi \rangle = \left\langle d_\pi \mathcal{Q}^\pi \left(\nabla F(\mu_\pi) - \nabla F(\mu_t) \right), \nabla_\theta \pi(a|s) \right\rangle. \tag{10}$$

Similarly, we have the policy gradient for the conditional divergence (3) as

$$\nabla_\theta D_F^{d_t}(\pi, \pi_t) = \left\langle d_t(\nabla F(\pi) - \nabla F(\pi_t)), \nabla_\theta \pi(a|s) \right\rangle,$$

which does not have a discounted sum, since d_t is fixed and independent of $\pi = \pi_\mu$.

3.3 Off-policy Correction

In this section, we discuss the practical method for estimating $\mathcal{Q}^\pi(f)$ under a behavior policy π_t. In distributed reinforcement learning with asynchronous gradient update, the policy π_t which generated the trajectories may deviate from the policy π_θ currently being optimized. Thus off-policy correction is usually needed for the robustness of the algorithm (e.g. V-trace as in IMPALA (Espeholt et al., 2018)). Consider

$$\begin{aligned}\sum_{s,a} d_\pi(s)\mathcal{Q}^\pi(f)(s,a)\nabla_\theta \pi(a|s) &= \mathbb{E}_{(s,a)\sim \pi d_\pi} \mathcal{Q}^\pi(f)(s,a)\nabla_\theta \log \pi(a|s) \\ &= \mathbb{E}_{(s,a)\sim \pi_t d_{\pi_t}} \frac{d_\pi(s)}{d_{\pi_t}(s)} \frac{\pi(a|s)}{\pi_t(a|s)} \mathcal{Q}^\pi(f)(s,a)\nabla_\theta \log \pi(a|s)\end{aligned}$$

for $f = g_t$ or $f = \nabla F(\mu_\pi) - \nabla F(\mu_t)$. We would like to have an accurate estimation of $\mathcal{Q}^\pi(g_t)$ (9) and $\mathcal{Q}^\pi(\nabla F(\mu_\pi) - \nabla F(\mu_t))$ (10), and correct the deviation from d_{π_t} to d_π and π_t to π.

For the estimation of $\mathcal{Q}^\pi(f)$ under a behavior policy π_t, possible methods include Retrace (Munos et al., 2016) providing an estimator of state-action value $\mathcal{Q}^\pi(f)$, and V-trace (Espeholt et al., 2018) providing an estimator of state value $\mathbb{E}_{a\sim\pi}\mathcal{Q}^\pi(f)(s,a)$. In this work, we utilize the V-trace (Section 4.1 (Espeholt et al., 2018)) estimation $v_{s_i} = v_i$ along a trajectory starting at $(s_i, a_i = s, a)$ under π_t.

Details of multi-step Q-value estimation can be found in Appendix A. With the value estimation v_s, the $\mathcal{Q}^\pi(g_t)$ is estimated with

$$\hat{A}_{s,a} = r_i + \gamma v_{i+1} - V_\theta(s_i). \tag{11}$$

We subtract a baseline $V_\theta(s_i)$ to reduce variance in estimation, as $\mathbb{E}_{\pi_t,d_t} \frac{\pi_\theta}{\pi_t} V_\theta(s) \nabla_\theta \log \pi_\theta = 0$. For the estimation of $\mathcal{Q}^\pi(\nabla F(\mu_\pi) - \nabla F(\mu_t))$, we use the n-steps truncated importance sampling as

$$\hat{D}_{s,a} = f(s_i, a_i) + \sum_{j=1}^{n} \gamma^j (\prod_{k=0}^{j-1} c_{i+k}) \rho_{i+j} f(s_{i+j}, a_{i+j}). \tag{12}$$

in which we use the notation $c_j = \min(\bar{c}_D, \frac{\pi_\theta(a_j|s_j)}{\pi_t(a_j|s_j)})$ and $\rho_j = \min(\bar{\rho}_D, \frac{\pi_\theta(a_j|s_j)}{\pi_t(a_j|s_j)})$. The formula also corresponds to V-trace under the condition $V(\cdot) \equiv 0$. For RNN model trained on continuous roll-out samples, we set n equals to the max-length till the end of roll-out.

For the correction of state distribution $d_\pi(s)/d_{\pi_t}(s)$, previous solutions include the use of emphatic algorithms as in (Sutton et al., 2016), or through an estimate of state density ratio as in (Liu et al., 2018). However, in our experience, less than the optimal estimation of density ratio will lead to additional error, causing instability. Therefore in this paper, we propose a different solution by restricting our attention to the correction of π_t to π via importance sampling and omitting the difference of d_π/d_{π_t} in the algorithm. This introduces a bias in the gradient estimation, which we propose a new method to handle in this paper. Specifically, we show that although the omission of the state ratio introduces a bias in the gradient, the bias can be bounded by the regularization term of conditional KL divergence (see Appendix B). Therefore by explicitly adding an KL divergence regularization, we can effectively control the degree of off-policy bias caused by d_π/d_{π_t} in that small regularization value implies a small bias. This approach naturally combines mirror descent with KL divergence regularization, leading to a more stable algorithm that is robust to off-policy data, as we will demonstrate by empirical experiments.

The final loss consists of the policy loss $L_\pi(\theta)$ and the value loss $L_v(\theta)$. To be specific, the gradient of policy loss is defined as

$$\nabla_\theta L_\pi(\theta) = \mathbb{E}_{\pi_t,d_t} \frac{\pi}{\pi_t} (\hat{D}_{s,a} - \eta \hat{A}_{s,a}) \nabla_\theta \log \pi. \tag{13}$$

We can also use proximal methods like PPO (Schulman et al., 2017b) in conjunction with divergence augmentation. A practical implementation is elaborated later in Formula (19). In addition to the policy loss, we also update V_θ with value gradient defined as

$$\nabla L_v(\theta) = \mathbb{E}_{\pi_t,d_t} \frac{\pi}{\pi_t} (V_\theta(s) - v_s) \nabla_\theta V_\theta(s), \tag{14}$$

where $v_s = v_{s_i}$ is the multi-step value estimation with V-trace. The parameter θ is then updated with a mixture of policy loss and value loss

$$\theta \leftarrow \theta - \alpha_t (\nabla_\theta L_\pi(\theta) + b \nabla_\theta L_v(\theta)), \tag{15}$$

in which α_t is the current learning rate, and b is the loss scaling coefficient. The algorithm is summarized in Algorithm 1.

4 Related Works

The policy performance in Equation (2) and the well-known policy difference lemma (Kakade and Langford, 2002) serve a fundamental role in policy-based reinforcement learning (e.g TRPO, PPO (Schulman et al., 2015, 2017b)). The gradient with respect to the policy performance and policy difference provides a natural direction for policy optimization. And to restrict the changes in each policy improvement step, as well as encouraging exploration at the early stage, the constraint-based policy optimization methods try to limit the changes in the policy by constraining the divergence between behavior policy and current policy. The use of entropy maximization in reinforcement learning can be dated back to the work of Williams and Peng (1991). And methods with relative entropy regularization include Peters et al. (2010); Schulman et al. (2015). The relationship between these methods and the mirror descent method has been discussed in Neu et al. (2017). With

Algorithm 1 Divergence-Augmented Policy Optimization (DAPO)

Input: $D_F(\mu', \mu)$, total iteration T, batch size M, learning rate α_t.
Initialize : randomly initiate θ_0
for $t = 0$ **to** T **do**
  (in parallel) Use $\pi_t = \pi_{\theta_t}$ to generate trajectories.
  for $m = 1$ **to** M **do**
    Sample $(s_i, a_i) \in \mathcal{S} \times \mathcal{A}$ w.p. $d_t \pi_t$.
    Estimate state value v_{s_i} (e.g. by V-trace).
    Calculate Q-value estimation $\hat{A}_{s,a}$ (11) and divergence estimation $\hat{D}_{s,a}$ (12).
      $\hat{A}_{s,a} = r_i + \gamma v_{i+1} - V_\theta(s_i)$,
      $\hat{D}_{s,a} = f(s_i, a_i) + \sum_{j=1}^{n} \gamma^j (\prod_{k=0}^{j-1} c_{i+k}) \rho_{i+j} f(s_{i+j}, a_{i+j})$.
    Update θ with respect of policy loss (13, optionally 19) and value loss (14)
      $\theta \leftarrow \theta - \alpha_t (\nabla_\theta L_\pi(\theta) + b \nabla_\theta L_v(\theta))$.
  end for
  Set $\theta_{t+1} = \theta$.
end for

notations in this work, consider the natural choice of F as the *negative Shannon entropy* defined as $F(x) = \sum_i x_i \log(x_i)$, the $D_F(\cdot, \cdot)$ becomes the KL-divergence $D_{\text{KL}}(\cdot, \cdot)$. By the equivalence of sub-gradient projection (6) and mirror descent (7), the mirror descent policy optimization with KL-divergence can be written as

$$\mu_{t+1} = \arg\min_{\mu \in \Delta_\Pi} D_{\text{KL}}(\mu, \tilde{\mu}_{t+1}) = \arg\min_{\mu \in \Delta_\Pi} D_{\text{KL}}(\mu, \mu_t) + \eta \langle g_t, \mu \rangle. \tag{16}$$

Under slightly different settings, this learning objective is the regularized version of the constrained optimization problem considered in Relative Entropy Policy Search (REPS) (Peters et al., 2010); And for $\ell_t(\mu)$ depending on t, the Equation (16) can also recover the O-REPS method considered in Zimin and Neu (2013). On the other hand, as the KL-divergence (and Bregman divergence) is asymmetric, we can also replace the $D_F(x, y)$ in either formulation (6, 7) with reverse KL $D_{\text{KL}}(y, x)$, which will result in different iterative algorithms (as the reverse KL is no longer a Bregman divergence, the equivalence of Formula (6) and (7) no longer holds). Consider replacing $D_F(\mu, \tilde{\mu}_{t+1})$ with $D_{\text{KL}}(\tilde{\mu}_{t+1}, \mu)$ in sub-gradient projection (6), we have the "mirror map" method with reverse KL as

$$\mu_{t+1} = \arg\min_{\mu \in \Delta_\Pi} D_{\text{KL}}(\tilde{\mu}_{t+1}, \mu), \tag{17}$$

which is essentially the MPO algorithm (Abdolmaleki et al., 2018) under a probabilistic inference perspective, and MARWIL algorithm (Wang et al., 2018) when learning from off-policy data. Similarly, consider the replacement of $D_F(\mu, \mu_t)$ with $D_{\text{KL}}(\mu_t, \mu)$ in mirror descent (7), we have the "mirror descent" method with reverse KL as

$$\mu_{t+1} = \arg\min_{\mu \in \Delta_\Pi} D_{\text{KL}}(\mu_t, \mu) + \eta \langle g_t, \mu \rangle, \tag{18}$$

which can approximately recover the TRPO optimization objective (Schulman et al., 2015) (if the relative entropy between two state-action distributions $D_{\text{KL}}(\mu_t, \mu)$ in (18) is replaced by the conditional entropy $D_{\text{KL}}^{d_t}(\pi_t, \pi)$, also see Section 5.1 of Neu et al. (2017)).

Besides, we note that there are other choices of constraint for policy optimization as well. For example, in (Lee et al., 2018; Chow et al., 2018; Lee et al., 2019), a Tsallis entropy is used to promote sparsity in the policy distribution. And in (Belousov and Peters, 2017), the authors generalize KL, Hellinger distance, and reversed KL to the class of f-divergence. In preliminary results, we found divergence based on 0-potential (Audibert et al., 2011; Bubeck and Cesa-Bianchi, 2012) is also promising for policy optimization. We left this for future research.

For multi-step KL divergence regularized policy optimization, we note that the formulation also corresponds to the KL-divergence-augmented return considered previously in several works (Fox et al. (2015), Section 3 of Schulman et al. (2017a)), although in Schulman et al. (2017a) the authors use a fixed behavior policy instead of π_t as in ours. More often, the Shannon-entropy-augmented return can be dated back to earlier works (Kappen, 2005; Todorov, 2007; Ziebart et al., 2008; Nachum et al., 2017), and is a central topic in "soft" reinforcement learning (Haarnoja et al., 2017, 2018).

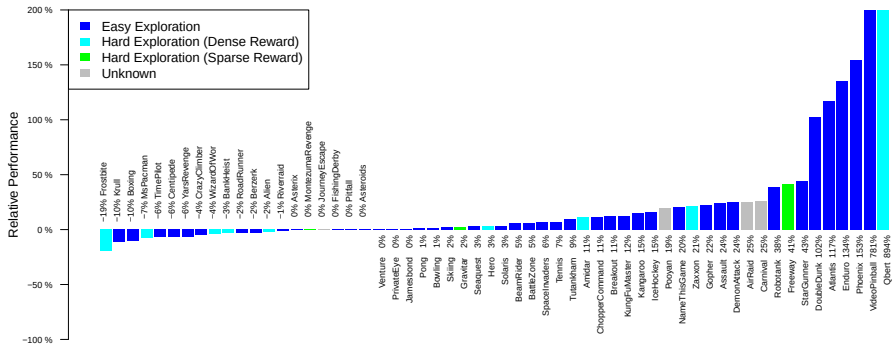


Figure 1: Relative score improvement of PPO+DA compared with PPO on 58 Atari environments. The relative performance is calculated as a $\frac{\text{proposed}-\text{baseline}}{\max(\text{human},\text{baseline})-\text{random}}$ (Wang et al., 2016b). The Atari games are categorized according to Figure 4 of (Oh et al., 2018).

The mirror descent method is originally introduced by the seminal work of Nemirovsky and Yudin (1983) as a convex optimization method. Also, the online stochastic mirror descent method has alternative views, e.g. Follow the Regularized Leader (McMahan, 2011), and Proximal Point Algorithm (Rockafellar, 1976). For more discussions on mirror descent and online learning, we refer interested readers to the work of Cesa-Bianchi and Lugosi (2006) and Bubeck and Cesa-Bianchi (2012).

5 Experiments

In the experiments, we test the exploratory effect of divergence augmentation comparing with entropy augmentation, and the empirical difference between multi-step and 1-step divergence. For the experiments, we mainly consider the DAPO algorithm (1) associated with the conditional KL divergence (see R_C and D_C in (Neu et al., 2017)). For $F(\mu) = \sum_{s,a} \mu(s,a) \log \frac{\mu(s,a)}{\sum_b \mu(s,b)}$, we have the gradient in (10) as

$$\nabla F(\mu_\pi) - \nabla F(\mu_t) = \log \frac{\pi}{\pi_t}.$$

The multi-step divergence augmentation term as in (12) is then calculated as

$$\hat{D}^{\text{KL}}_{s,a} = \log \frac{\pi(a_i|s_i)}{\pi_t(a_i|s_i)} + \sum_{j=1}^{n} \gamma^j (\prod_{k=1}^{j-1} c_{i+k}) \rho_{i+j} \log \frac{\pi(a_{i+j}|s_{i+j})}{\pi_t(a_{i+j}|s_{i+j})}.$$

As a baseline, we also implement the PPO algorithm with a V-trace (Espeholt et al., 2018) estimation of advantage function A^π for target policy[3]. Specifically, we consider the policy loss as:

$$L^{\text{PPO}}_\pi(\theta) = \mathbb{E}_{\pi_t, d_t} \min(\frac{\pi_\theta}{\pi_t} A_{s,a}, \text{clip}(\frac{\pi_\theta}{\pi_t}, 1-\epsilon, 1+\epsilon) A_{s,a}), \tag{19}$$

where we choose $\epsilon = 0.2$ and the advantage is estimated by $R_{s,a}$. We also tested the DAPO algorithm with PPO, with the advantage estimation $A_{s,a}$ in (19) replaced with $\hat{A}_{s,a} - \frac{1}{\eta}\hat{D}_{s,a}$ defined in (11) and (12). We will refer to this algorithm as PPO+DA in the following sections.

5.1 Algorithm Settings

The algorithm is implemented with TensorFlow (Abadi et al., 2016). For efficient training with deep neural networks, we use the Adam (Kingma and Ba, 2014) method for optimization. The learning rate is linearly scaled from 1e-3 to 0. The parameters are updated according to a mixture of policy loss and value loss, with the loss scaling coefficient $c = 0.5$. In calculating multi-step λ-returns $R_{s,a}$

[3] In the original PPO (Schulman et al., 2017b) they use $\hat{A}$ as the advantage estimation of behavior policy A^{π_t}.

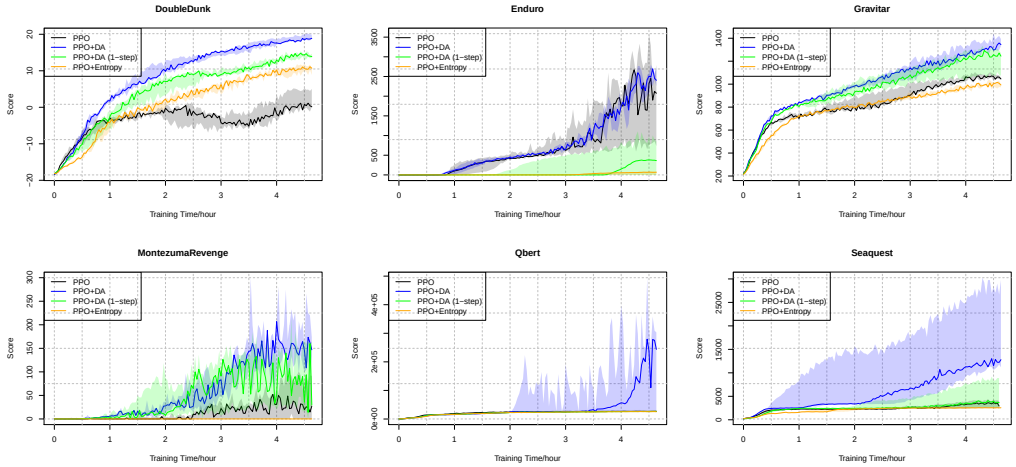


Figure 2: Performance comparison of selected environments of Atari games. The performance of **PPO**, PPO+DA, PPO+DA (1-step), and PPO+Entropy are plotted in different colors. The score for each game is plotted on the y-axis with running time on the x-axis, as the algorithm is paralleled asynchronously in a distributed environment. For each line in the plots, we run the experiment 5 times with the same parameters and environment settings. The median scores are plotted in solid lines, while the regions between 25% and 75% quantiles are shaded with respective colors.

and divergence $D_{s,a}$, we use fixed $\lambda = 0.9$ and $\gamma = 0.99$. The batch size is set to 1024, with roll-out length set to 32, resulting in 1024/32=32 roll-outs in a batch. The policy π_t and value V_t is updated every 100 iterations (M = 100 in Algorithm 1). With our implementation, the training speed is about 25k samples per second, and the data generating speed is about 220 samples per second for each actor, resulting in about 3500 samples per second for a total of 16 actors. Note that the PPO results may not be directly comparable with other works (Schulman et al., 2017b; Espeholt et al., 2018; Xu et al., 2018), mainly due to the different number of actors used. Unless otherwise noted, each experiment is allowed to run 16000 seconds (about 4.5 hours), corresponding a total of 60M samples generated and 400M samples (with replacement) trained. Details of experimental settings can be found in Appendix A.

5.2 Empirical Results

We test the algorithm on 58 Atari environments and calculate its relative performance with PPO (Schulman et al., 2017b). The empirical performance is plotted in Figure 1. We run PPO and PPO+DA with the same environmental settings and computational resources. The relative performance is calculated as $\frac{\text{proposed}-\text{baseline}}{\max(\text{human},\text{baseline})-\text{random}}$ (Wang et al., 2016b). We also categorize the game environments into easy exploration games and hard exploration games (Oh et al., 2018). We see that with a KL-divergence-augmented return, the algorithm PPO+DA performs better than the baseline method, especially for the games that may have local minimums and require deeper exploration. We plot the learning curves of PPO+DA (in blue) comparing with PPO (in **black**) and other baseline methods on 6 typical environments in Figure 2. Detailed learning curves for PPO and PPO+DA for the complete 58 games can be found in Figure 3 in the Appendix.

5.2.1 Divergence augmentation vs Entropy augmentation

We test the effect of divergence augmentation in contrast to the entropy augmentation (plotted in orange in Figure 2). Entropy augmentation can prevent premature convergence and encourage exploration as well as stabilize policy during optimization. However, the additional entropy may hinder the convergence to the optimal action, as it alters the original learning objective. We set $f(s,a)$ as $\log \pi(a|s)$ in Formula (12), and experiment the algorithm with $\frac{1}{\eta} = 0.5, 0.1, 0.01, 0.001$, in which we found that $\frac{1}{\eta} = 0.1$ performs best. From the empirical results, we see that divergence-augmented PPO works better, while the entropy-augmented version may be too conservative on policy changes, resulting in inferior performance on these games.

5.2.2 Multi-step divergence vs 1-step divergence

In Figure 2, we also test the PPO+DA algorithm with its 1-step divergence-augmented counterpart (plotted in green). We rerun the experiments with the parameter $\bar{c}_D$ (Formula (12)) set to 0, which means we only aggregate the divergence on the current state and action $f(s_i, a_i)$, without summing up future discounted divergence $f(s_{i+j}, a_{i+j})$. This method also relates to the conditional divergence defined in Formula (3), and shares more similarities with previous works on regularized and constrained policy optimization methods (Schulman et al., 2015; Achiam et al., 2017). We see that with multi-step divergence augmentation, the algorithm can achieve high scores, especially on games requiring deeper exploration like Enduro and Qbert. We hypothesize that the accumulated divergence on future states can encourage the policy to explore more efficiently.

6 Conclusion

In this paper, we proposed a divergence-augmented policy optimization method to improve the stability of policy gradient methods when it is necessary to reuse off-policy data. We showed that the proposed divergence augmentation technique can be viewed as imposing Bregman divergence constraint on the state-action space, which is related to online mirror descent methods. Experiments on Atari games showed that in the data-scarce scenario, the proposed method works better than other state-of-the-art algorithms such as PPO. Our results showed that the technique of divergence augmentation is effective when data generated by previous policies are reused in policy optimization.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P. A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zhang, X. (2016). Tensorflow: A system for large-scale machine learning. *arXiv preprint arXiv:1605.08695*.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. (2018). Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. *arXiv preprint arXiv:1705.10528*.

Asmussen, S. (2003). Markov chains. *Applied Probability and Queues*, pages 3–38.

Audibert, J.-Y. and Bubeck, S. (2009). Minimax policies for adversarial and stochastic bandits. In *In Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*, pages 217–226.

Audibert, J.-Y., Bubeck, S., and Lugosi, G. (2011). Minimax policies for combinatorial prediction games. In Kakade, S. M. and von Luxburg, U., editors, *Proceedings of the 24th Annual Conference on Learning Theory (COLT)*, volume 19 of *Proceedings of Machine Learning Research*, pages 107–132, Budapest, Hungary. PMLR.

Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175.

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

Belousov, B. and Peters, J. (2017). f-Divergence constrained policy improvement. *ArXiv e-prints*.

Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

Bubeck, S. and Cesa-Bianchi, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122.

Cesa-Bianchi, N., Gentile, C., Lugosi, G., and Neu, G. (2017). Boltzemann exploration done right. In *Advances in Neural Information Processing Systems*, pages 6284–6293.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

Chow, Y., Nachum, O., and Ghavamzadeh, M. (2018). Path consistency learning in tsallis entropy regularized mdps. In *International Conference on Machine Learning*, pages 978–987.

Csiszar, I. and Körner, J. (2011). *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. (2018). IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*.

Fox, R., Pakman, A., and Tishby, N. (2015). Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. (2017). Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

Hasselt, H. v., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2094–2100. AAAI Press.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, pages 267–274.

Kakade, S. M. (2002). A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538.

Kappen, H. J. (2005). Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005(11):P11011.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, K., Choi, S., and Oh, S. (2018). Maximum causal tsallis entropy imitation learning. In *Advances in Neural Information Processing Systems*, pages 4403–4413.

Lee, K., Kim, S., Lim, S., Choi, S., and Oh, S. (2019). Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning.

Liu, Q., Li, L., Tang, Z., and Zhou, D. (2018). Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5361–5371.

McMahan, B. (2011). Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 525–533, Fort Lauderdale, FL, USA. PMLR.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. (2016). Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2775–2785.

Nemirovsky, A. S. and Yudin, D. B. (1983). *Problem complexity and method efficiency in optimization*. Wiley.

Neu, G., Jonsson, A., and Gómez, V. (2017). A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*.

Oh, J., Guo, Y., Singh, S., and Lee, H. (2018). Self-imitation learning. In *International Conference on Machine Learning*, pages 3875–3884.

Peters, J., Mülling, K., and Altun, Y. (2010). Relative entropy policy search. In *AAAI*, pages 1607–1612.

Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898.

Schulman, J., Chen, X., and Abbeel, P. (2017a). Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897.

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

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Sutton, R. S. and Barto, A. G. (2017). *Introduction to reinforcement learning, 2nd edition, in progress*. MIT press.

Sutton, R. S., Mahmood, A. R., and White, M. (2016). An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 17(1):2603–2631.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Todorov, E. (2007). Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pages 1369–1376.

Wang, Q., Xiong, J., Han, L., Sun, P., Liu, H., and Zhang, T. (2018). Exponentially weighted imitation learning for batched historical data. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 6291–6300. Curran Associates, Inc.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2016a). Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016b). Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003.

Williams, R. J. and Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.

Xu, Z., van Hasselt, H. P., and Silver, D. (2018). Meta-gradient reinforcement learning. In *Advances in neural information processing systems*, pages 2396–2407.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA.

Zimin, A. and Neu, G. (2013). Online learning in episodic markovian decision processes by relative entropy policy search. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc.

A Details of the algorithm

A.1 Environment Settings

We evaluate the algorithm on the Atari 2600 video games from Arcade Learning Environment (ALE) (Bellemare et al., 2013), which is widely used as a standard benchmark for deep reinforcement learning, especially for distributed training (Horgan et al., 2018; Espeholt et al., 2018). There are at least two evaluation protocols, One is the "human starts" protocol (Hasselt et al., 2016), in which each episode for evaluation is started at a state sampled from human play. The other is the "no-ops start" protocol (Mnih et al., 2015), in which the starting state of an episode for evaluation is generated by playing a random length of no-op actions in the environment. We adopt the later evaluation protocol both for training and testing in this work. The environment is based on the OpenAI Gym (Brockman et al., 2016). We mostly follow the environment settings as in (Mnih et al., 2015). The environment is randomly initialized by playing a random number (no more than 30) of no-op actions. During playing, each action is repeated for 4 contiguous frames, and every 4th frame is taken a pixel-wise max over the previous frame, and then returned as the screen observation. The size of the raw screen is 210×160 pixels with 128 colors. The colored image is firstly converted to gray-scale and then resized to 84×84 pixels represented by integers from 0 to 255, followed by a scaling to floats between 0 to 1. We also use the "episodic life" trick in the training phase: For games with a life counter, the loss of life is marked as an end for the current episode. The rewards are clipped with a $sgn()$ function, such that positive rewards are represented by 1, negative rewards as -1, and 0 otherwise. For some games (e.g. Atlantis) we observe that there is a maximum limit of 100000 steps for each episode, corresponding to 400000 raw frames. In addition to the settings above, we also reset the environment if no reward is received in 1000 steps, to prevent the environment from accidental stuck.

A.2 Network Structure

For comparable results, we use a similar network structure as in (Mnih et al., 2015). The first layer consists of 32 convolution filters of 8×8 with stride 4 and applies a ReLU non-linearity. And the second layer convolves the image with 64 filters of 4×4 with stride 2 followed by a ReLU rectifier. The third layer has 64 filters of 3×3 with stride 1 followed by a rectifier. The final hidden layer is fully-connected and consists of 512 ReLU units. The output layer is two-headed, representing $\pi_\theta(\cdot|s)$ and $V_\theta(s)$ respectively. For the vanilla model, the input is stacked with 4 frames; while for RNN model, we put an additional LSTM (Hochreiter and Schmidhuber, 1997) layer with 256 cells after the fully-connected layer, which is similar to the previous works (Espeholt et al., 2018).

A.3 Training Framework

For large-scale training with the interested algorithms, we adopt an "Actor-Learner" style (Horgan et al., 2018; Espeholt et al., 2018) distributed framework. In our distributed framework, actors are responsible for generating massive trajectories with current policy; while learners are responsible for updating policy with the data generated by the actors. To be specific, in its main loop, an actor runs a local environment with actions from current local policy and caches the generated data at local memory. The running policy is updated periodically to the latest policy at the learner; while the generated data is sent to the learner asynchronously. At the learner side, the learner keeps at most 20 latest episodes generated by each actor respectively, in a FIFO manner. Each batch of samples for training are randomly sampled from these trajectories with replacement. We deploy the distributed training framework on a small cluster. The learner runs on a GPU machine and occupies an M40 card, while actors run in 16 parallel processes on 2.5GHz Xeon Gold 6133 CPUs.

A.4 Multi-step Return

In this work, we utilize the V-trace (Section 4.1 (Espeholt et al., 2018)) estimation $v_{s_i} = v_i$ along a trajectory starting at $(s_i, a_i = s, a)$ under π_t defined recursively as

$$v_j = \begin{cases} V_\theta(s_j) + \delta_j V_\theta + \gamma c_j (v_{j+1} - V_\theta(s_{j+1})), & i \le j < n \\ r_j + \gamma \lambda_V v_{j+1} + \gamma(1-\lambda_V) V_t(s_{j+1}), & n \le j < T \end{cases} \tag{20}$$

Table 1: Hyper-parameters

Name	Value
Batch size	1024
Replay memory size	16384 (2^{14})
λ	0.9
Rollout length	32
Burn-in samples	1024
Learning rate	0.001 to 0
$\bar{c}_D$ (Formula 12)	0.5
$\bar{\rho}_D$ (Formula 12)	1.0
$\bar{c}_V$ (Formula 20)	1.0
$\bar{\rho}_V$ (Formula 20)	1.0
ϵ (Formula 19)	0.2
$1/\eta$	0.5
b (Formula 15)	0.5
Optimizer	Adam

where $\delta_j V_\theta = \rho_j(r_j + \gamma V_\theta(s_{j+1}) - V_\theta(s_j))$, $\rho_j = \min(\bar{\rho}_V, \frac{\pi_\theta(a_j|s_j)}{\pi_t(a_j|s_j)})$, $c_j = \min(\bar{c}_V, \frac{\pi_\theta(a_j|s_j)}{\pi_t(a_j|s_j)})$, and $0 \leq \lambda_V \leq 1$. The state value estimation function at iteration t is denoted as $V_t(\cdot) = V_{\theta_t}(\cdot)$. The definition of (20) can be seen as following V-trace algorithm along the roll-out (for which we have $\pi_\theta(a_j|s_j)$ and $V_\theta(s_j)$) for $i \leq j < n$, and switch to TD(λ) until a terminal time T (which is estimated offline as we only have $V_t(s_j)$ instead of $\pi_\theta(a_j|s_j)$ and $V_\theta(s_j)$ for $n \leq j < T$). It is noted that TD(λ) also corresponds to V-trace in on-policy settings (Remark 2, (Espeholt et al., 2018)).

A.5 Hyper-parameters

The default hyper-parameters used in our experiments are given in Table 1.

B Theoretical Analysis

In this section, we provide some theoretical analysis of the parametrized algorithm considered in this work. We show that the bias introduced by omitting the state ratio can be bounded by the divergence up to a constant factor.

B.1 Error bound of the biased gradient

Without ambiguity, we define $\pi = \pi_\theta$, $\pi_t = \pi_{\theta_t}$ for brevity, the policy value as

$$V(\theta) = \langle r, \mu_{\pi_\theta} \rangle = \mathbb{E}_{s,a \sim d_\pi \pi} r(s, a),$$

and the conditional KL-divergence for parametrized policies as

$$D(\theta, \theta_t) = \mathbb{E}_{s,a \sim d_\pi \pi} \log \frac{\pi(a|s)}{\pi_t(a|s)}.$$

The algorithm iteratively maximizes the following equation with SGD steps

$$\theta_{t+1} \approx \arg\max_\theta f(\theta, \theta_t) \equiv V(\theta) - \lambda D(\theta, \theta_t). \tag{1}$$

Denote $A^\pi(s, a)$ as the advantage function of reward r following target policy π, and $\mathbf{A}^\pi_{\pi_t}(s, a)$ as the advantage function of pseudo reward $(\log \pi - \log \pi_t)$ following target policy π, the gradient of (1) is

$$\nabla_\theta f(\theta, \theta_t) = \mathbb{E}_{(s,a) \sim d_{\pi_t} \pi_t} \frac{d_\pi(s)}{d_{\pi_t}(s)} \frac{\pi(a|s)}{\pi_t(a|s)} (A^\pi(s, a) - \lambda \mathbf{A}^\pi_{\pi_t}(s, a)) \nabla_\theta \log \pi(a|s).$$

In the actual implementation, we omit the state ratio of d_π / d_{π_t}, resulting in a biased gradient

$$g(\theta, \theta_t) = \mathbb{E}_{(s,a) \sim d_{\pi_t} \pi_t} \frac{\pi(a|s)}{\pi_t(a|s)} (A^\pi(s, a) - \lambda \mathbf{A}^\pi_{\pi_t}(s, a)) \nabla_\theta \log \pi(a|s).$$

In the following proposition, for target policy $\pi \equiv \pi_\theta$ and reference policy $\tilde{\pi} \equiv \pi_{\tilde{\theta}}$, we show that the error $\delta(\theta, \tilde{\theta}) \equiv \left\| \nabla f(\theta, \tilde{\theta}) - g(\theta, \tilde{\theta}) \right\|$ introduced by omitting the state ratio can be bounded by the conditional KL divergence. To be rigorous, we make the following assumptions:

Assumption 1 (Universal boundedness)**.** *For all $\theta, \tilde{\theta} \in \Theta$, the set for policy parametrization, there exist non-negative constants ζ_1, ζ_2 such that,*

$$\max_s \mathbb{E}_{a\sim\pi} \left\| \nabla_\theta \ln \pi(a|s) \right\| \leq \zeta_1,$$

$$\max \left\{ \max_{s,a} |A^\pi(s,a)|, \max_{s,a} |A^\pi(s,a) - \mathbf{A}^\pi_{\tilde{\pi}}(s,a)| \right\} \leq \zeta_2.$$

Then we have the following proposition

Proposition 1. *Under Assumption 1, the norm of the gradient bias can be bounded by the conditional KL-divergence:*

$$\delta(\theta, \tilde{\theta})^2 \leq cD(\theta, \tilde{\theta}).$$

Proof. The proof provided here is based on the perturbation theory. We firstly define the symbols and notations we used in the proof. Consider the difference on each state denoted as

$$\Delta_{\theta,\tilde{\theta}}(s) = \mathbb{E}_{a\sim\pi} \left\| [A^\pi(s,a) - \lambda \mathbf{A}^\pi_{\tilde{\pi}}(s,a)] \nabla_\theta \ln \pi(a|s) \right\|. \tag{2}$$

By triangular inequality and Hölder's inequality, we have

$$\delta(\theta, \tilde{\theta}) = \left\| \nabla_\theta f(\theta, \tilde{\theta}) - g(\theta, \tilde{\theta}) \right\| \leq \langle |d_\pi - d_{\tilde{\pi}}|, \Delta_{\theta,\tilde{\theta}} \rangle \leq \|d_\pi - d_{\tilde{\pi}}\|_1 \left\| \Delta_{\theta,\tilde{\theta}} \right\|_\infty.$$

Let $P_\pi \in \mathbb{R}^{|S|\times|S|}$ be the transition matrix associated with policy π

$$P_\pi(s'|s) = \sum_a P(s'|s,a)\pi(a|s).$$

The discounted state distribution can be written as

$$d_\pi = (1-\gamma) \sum_{t=0}^{\infty} (\gamma P_\pi)^t d_0 = (1-\gamma)(I - \gamma P_\pi)^{-1} d_0,$$

where d_0 is the initial state distribution. For policies π and $\tilde{\pi}$, consider the matrices $G \equiv (I - \gamma P_\pi)^{-1}$ and $\tilde{G} \equiv (I - \gamma P_{\tilde{\pi}})^{-1}$, we have

$$G^{-1} - \tilde{G}^{-1} = (I - \gamma P_\pi) - (I - \gamma P_{\tilde{\pi}}) = \gamma(P_{\tilde{\pi}} - P_\pi).$$

Multiplying by $\tilde{G}$ and G on the left and right side respectively, we have

$$\tilde{G} - G = \gamma \tilde{G}(P_{\tilde{\pi}} - P_\pi)G.$$

The difference of state distribution can be bounded as

$$\begin{aligned}
\|d_\pi - d_{\tilde{\pi}}\|_1 &= \left\| (1-\gamma)(G - \tilde{G}) d_0 \right\|_1 \\
&= \left\| \gamma(1-\gamma)\tilde{G}(P_\pi - P_{\tilde{\pi}}) G d_0 \right\|_1 \\
&= \left\| \gamma \tilde{G}(P_\pi - P_{\tilde{\pi}}) d_\pi \right\|_1 \\
&\leq \gamma \left\| \tilde{G} \right\|_1 \|(P_\pi - P_{\tilde{\pi}}) d_\pi\|_1 \\
&= \gamma \left\| \sum_{t=0}^{\infty} (\gamma P_{\tilde{\pi}})^t \right\|_1 \|(P_\pi - P_{\tilde{\pi}}) d_\pi\|_1 \\
&\leq \gamma \sum_{t=0}^{\infty} \gamma^t \left\| P^t_{\tilde{\pi}} \right\|_1 \|(P_\pi - P_{\tilde{\pi}}) d_\pi\|_1 .
\end{aligned}$$

Since the transition matrix, $P_{\tilde{\pi}}$ is a left stochastic matrix (Asmussen, 2003),

$$\begin{aligned}\|d_\pi - d_{\tilde{\pi}}\|_1 &\leq \gamma \sum_{t=0}^{\infty} \gamma^t \|(P_\pi - P_{\tilde{\pi}}) d_\pi\|_1 \\ &= \gamma(1-\gamma)^{-1} \|(P_\pi - P_{\tilde{\pi}}) d_\pi\|_1 ,\end{aligned}$$

we have that

$$\begin{aligned}\|(P_\pi - P_{\tilde{\pi}}) d_\pi\|_1 &= \sum_s \left| \sum_{s',a} \left(P(s'|s,a) \left(\pi(a|s) - \tilde{\pi}(a|s)\right)\right) d_\pi(s) \right| \\ &\leq \sum_{s,s'} \left| \sum_a \left(P(s'|s,a) \left(\pi(a|s) - \tilde{\pi}(a|s)\right)\right) \right| d_\pi(s) \\ &\leq \sum_{s,s',a} P(s'|s,a) \left|\left(\pi(a|s) - \tilde{\pi}(a|s)\right)\right| d_\pi(s) \\ &= \sum_{s,a} \left|\left(\pi(a|s) - \tilde{\pi}(a|s)\right)\right| d_\pi(s) \sum_{s'} P(s'|s,a) \\ &= \mathbb{E}_{s\sim d_\pi} \|\pi(\cdot|s) - \tilde{\pi}(\cdot|s)\|_1 \\ &= 2\mathbb{E}_{s\sim d_\pi} D_{\mathrm{TV}} \left(\pi(\cdot|s), \tilde{\pi}(\cdot|s)\right) \\ &\leq \mathbb{E}_{s\sim d_\pi} \sqrt{2 D_{\mathrm{KL}} \left(\pi(\cdot|s), \tilde{\pi}(\cdot|s)\right)} \\ &\leq \sqrt{2\mathbb{E}_{s\sim d_\pi} D_{\mathrm{KL}} \left(\pi(\cdot|s), \tilde{\pi}(\cdot|s)\right)}.\end{aligned}$$

of which the first two inequalities follow from the triangular inequality, the relationship between D_{TV} and D_{KL} is deduced by Pinsker's inequality (Csiszar and Körner, 2011), and the last inequality is by Jensen's inequality with concavity.

From the definition of $D(\theta, \tilde{\theta})$ and (2), we could get

$$\begin{aligned}\delta(\theta, \tilde{\theta}) &\leq \|d_\pi - d_{\tilde{\pi}}\|_1 \left\| \Delta_{\theta,\tilde{\theta}} \right\|_\infty \\ &\leq \frac{\gamma}{1-\gamma} \|(P_\pi - P_{\tilde{\pi}}) d_\pi\|_1 \left\| \Delta_{\theta,\tilde{\theta}} \right\|_\infty \\ &\leq \frac{\gamma}{1-\gamma} \max_s \Delta_{\theta,\tilde{\theta}}(s) \sqrt{2 D(\theta, \tilde{\theta})}.\end{aligned}$$

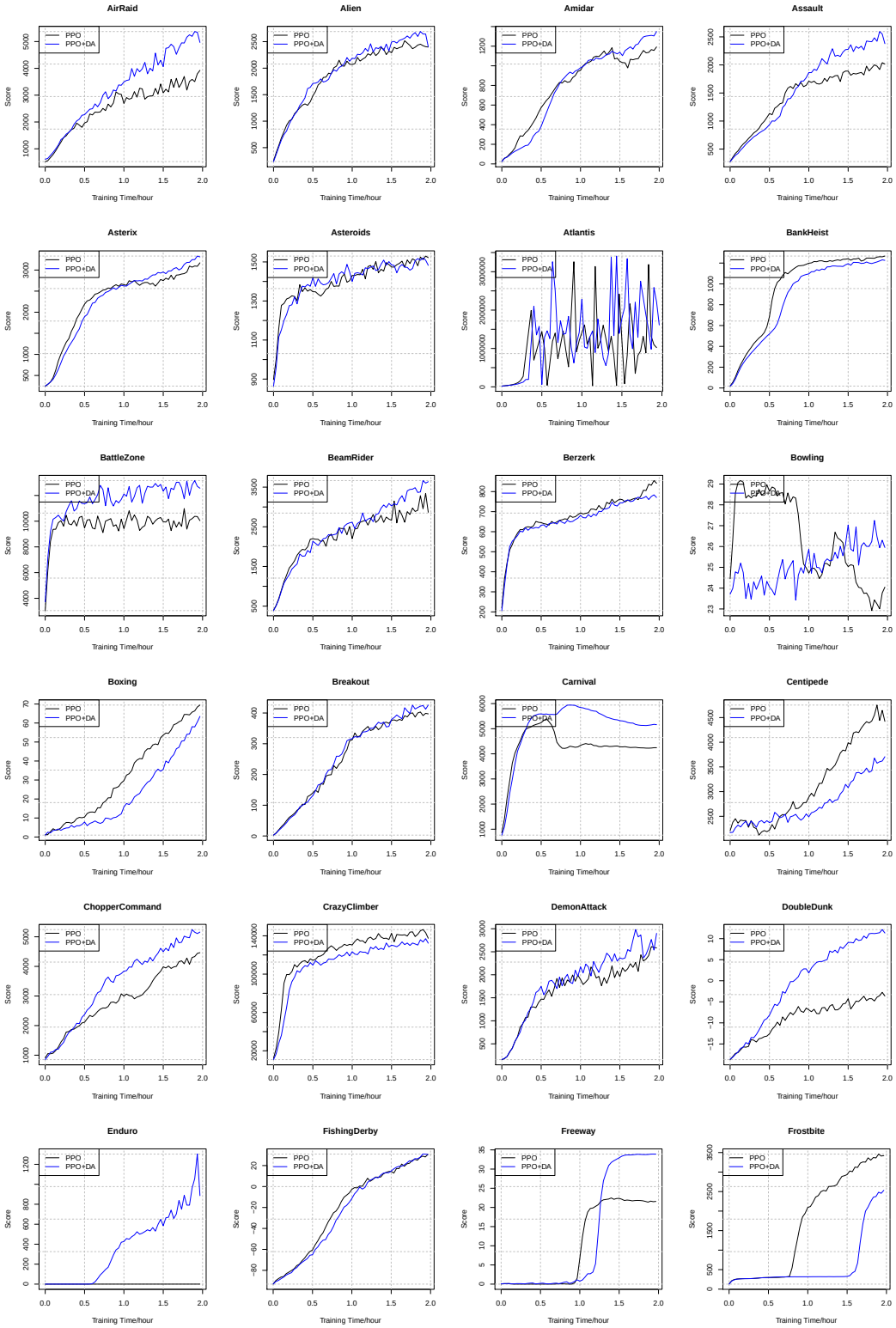
The final result follows from squaring both sides

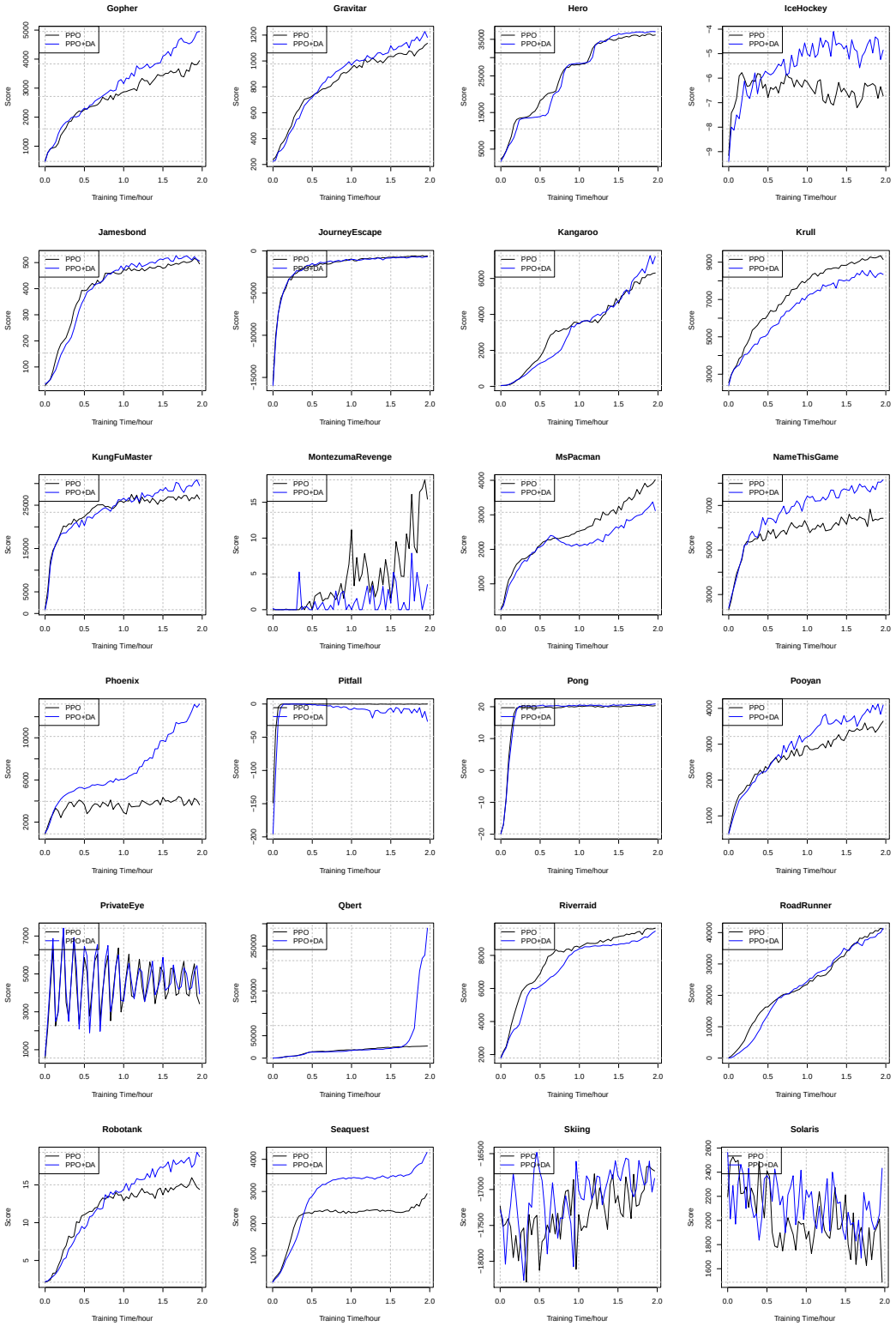
$$\begin{aligned}\delta(\theta, \tilde{\theta})^2 &\leq 2 \left(\frac{\gamma}{1-\gamma} \max_s \Delta_{\theta,\tilde{\theta}}(s) \right)^2 D(\theta, \tilde{\theta}) \\ &\leq 2 \left(\frac{\gamma}{1-\gamma} \zeta_1 \zeta_2 \right)^2 D(\theta, \tilde{\theta}) \\ &= c D(\theta, \tilde{\theta}).\end{aligned}$$

□

C Additional Empirical Results

For the algorithm performance as summarized in Figure 1, we provide the comparison results for each game in details. We also provide experimental results with 64 actors for interested readers.


AirRaid
Alien
Amidar
Assault
Asterix
Asteroids
Atlantis
BankHeist
BattleZone
BeamRider
Berzerk
Bowling
Boxing
Breakout
Carnival
Centipede
ChopperCommand
CrazyClimber
DemonAttack
DoubleDunk
Enduro
FishingDerby
Freeway
Frostbite
PPO
PPO+DA
Score
Training Time/hour


Gopher
Gravitar
Hero
IceHockey
Jamesbond
JourneyEscape
Kangaroo
Krull
KungFuMaster
MontezumaRevenge
MsPacman
NameThisGame
Phoenix
Pitfall
Pong
Pooyan
PrivateEye
Qbert
Riverraid
RoadRunner
Robotank
Seaquest
Skiing
Solaris
PPO
PPO+DA
Score
Training Time/hour

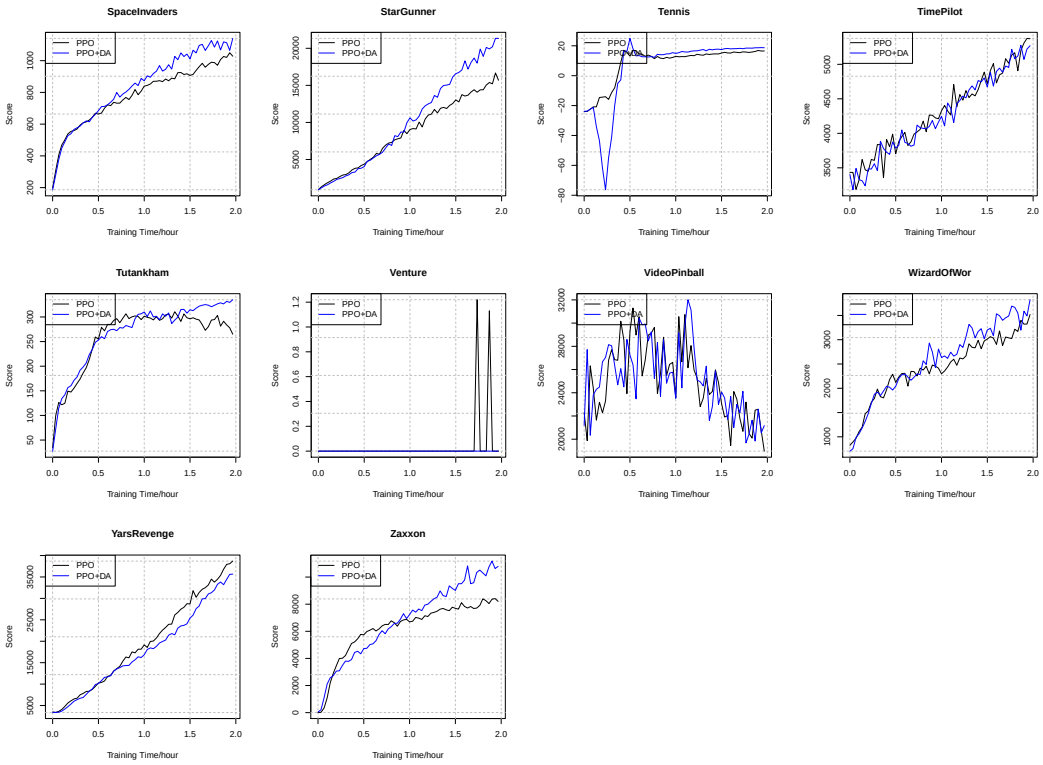
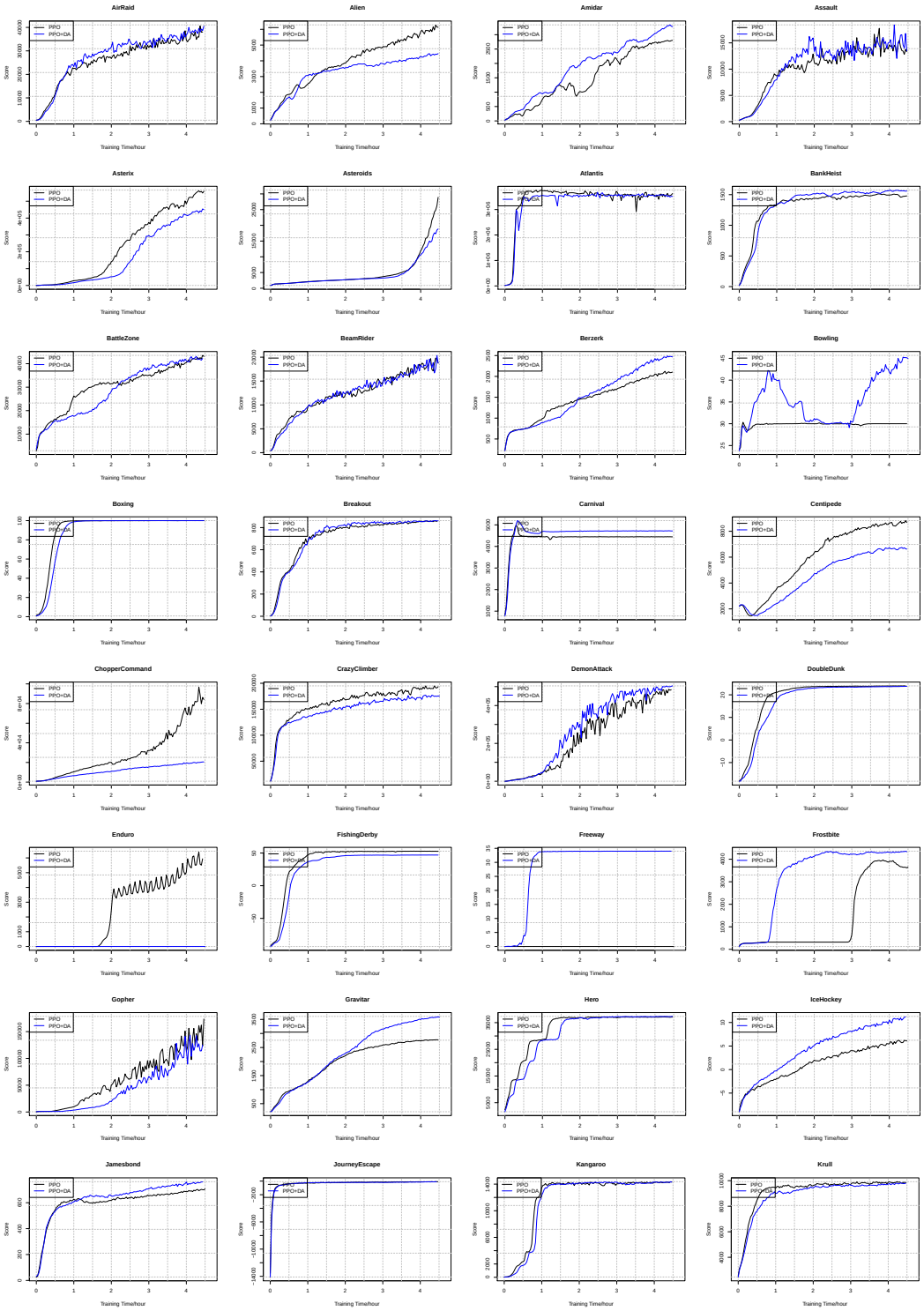


Figure 3: Performance comparison of PPO+DA with PPO on 58 Atari games. Each experiment is allowed to run for 2 hours as a limited time.


AirRaid
Alien
Amidar
Assault
Asterix
Asteroids
Atlantis
BankHeist
BattleZone
BeamRider
Berzerk
Bowling
Boxing
Breakout
Carnival
Centipede
ChopperCommand
CrazyClimber
DemonAttack
DoubleDunk
Enduro
FishingDerby
Freeway
Frostbite
Gopher
Gravitar
Hero
IceHockey
Jamesbond
JourneyEscape
Kangaroo
Krull
PPO
PPO+DA
Score
Training Time/hour

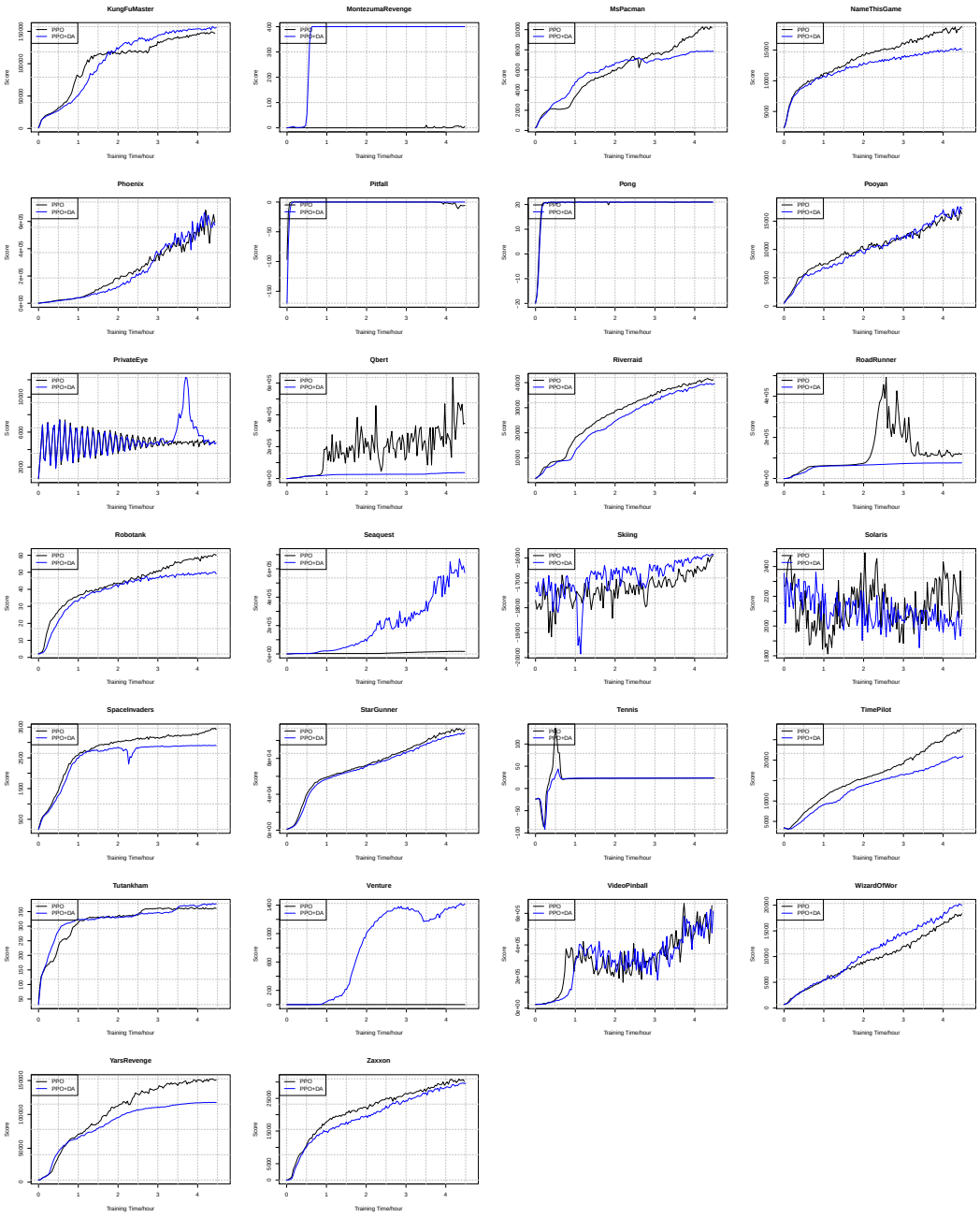


Figure 4: Performance comparison of PPO+DA with PPO on 58 Atari games, with the number of used actors increased to 64 and running time increased to 4 hours.