Trust Region Policy Optimization of POMDPs

Kamyar Azizzadenesheli
University of California, Irvine
kazizzad@uci.edu

Manish Kumar Bera
IIT Kanpur
bera.manish.kumar@gmail.com

Animashree Anandkumar
California Institute of Technology
anima@caltech.edu

Abstract

We propose Generalized Trust Region Policy Optimization (GTRPO), a Reinforcement Learning algorithm for TRPO of Partially Observable Markov Decision Processes (POMDP). While the principle of policy gradient methods does not require any model assumption, previous studies of more sophisticated policy gradient methods are mainly limited to MDPs. Many real-world decision-making tasks, however, are inherently non-Markovian, i.e., only an incomplete representation of the environment is observable. Moreover, most of the advanced policy gradient methods are designed for infinite horizon MDPs. Our proposed algorithm, GTRPO, is a policy gradient method for continuous episodic POMDPs. We prove that its policy updates monotonically improve the expected cumulative return. We empirically study GTRPO on many RoboSchool environments, an extension to the MuJoCo environments, and provide insights into its empirical behavior.

1 Introduction

One of the central challenges in reinforcement learning (RL) is to design an efficient algorithm for high-dimensional environments. Recently, model-free deep RL has shown promise in being able to tackle high-dimensional continuous environments. The primary model-free approaches on extreme ends of the deep RL spectrum are value-based and policy-gradient-based methods. Value-based approaches mainly learn a function to approximate the value of actions at any state. Then they exploit the approximated function to reason about the actions. On the other hand, policy gradient-based approaches, directly learn the policy and remove the value learning overhead.

Value-based model-free approaches have been successful in a wide variety of simulated domains, but they are limited to MDPs (Mnih et al., 2015). Similarly, even though the principle of policy gradient does not require any model assumption (Aleksandrov, 1968; Rubinstein, 1969; Baxter and Bartlett, 2001; Williams, 1992), previous advances of more sophisticated policy gradient methods are mainly limited to MDPs (Sutton et al., 2000; Schulman et al., 2015, 2017; Lillicrap et al., 2015) However, real-world problems rarely follow a MDP and the entire environment is rarely observable. Moreover, Sutton et al. (1998) argues that when a function approximation is deployed to represent the states, due to loss of information in the representation function the problem becomes POMDP in general. In addition, previous analyses in MDPs have mostly been dedicated to infinite horizon settings (Sutton et al., 2000; Schulman et al., 2015, 2017; Lillicrap et al., 2015). However, empirical examinations of these methods are mostly in episodic environments.

If the underlying environment is an MDP, then limiting to memoryless policies is appropriate since the optimal policy in an MDP is usually deterministic and memoryless. On the other hand, if the environment is a POMDP, then the optimal policy in the class of memoryless policies is in general stochastic. Policies for POMDPs can also depend on the entire history. However, maintaining history dependent policies is infeasible (PSPACE-Complete) (Montúfar et al., 2015; Azizzadenesheli et al., 2016a; Vlassis et al., 2012), and hence, many works dealing with POMDPs limit to the class of stochastic memoryless policies Azizzadenesheli et al. (2016b); Montúfar et al. (2015). Extension of the value-based method to POMDPs requires computation in belief space which is expensive and requires maintaining entire history. Extending the value-based methods to stochastic memoryless and limited memory policies is not possible if optimality is concerned. But

with policy gradient methods, we can design efficient algorithm under the class of stochastic memoryless policies.

Despite the MDP assumption in the mainstream of recent policy gradient methods, empirical studies have demonstrated superior performance when the class of stochastic policies is considered (Schulman et al., 2017, 2015). The stochastic policies also contribute to exploration. Interestingly, in many recent works, the stochasticity has been kept toward the end of the training phase, but these works do not explicitly assume a POMDP (Schulman et al., 2017, 2015).

In the policy gradient methods, in on-policy setting we collect data under the current policy at hand and exploit the acquired data to search for a new policy. This procedure iteratively improves the policy and maximizes the expected return. Under the infinite horizon MDP modeling assumption, Kakade and Langford (2002); Schulman et al. (2015) study the trust-region, a class of policy gradients methods which perform the policy search around the vicinity of the current policy, e.g., TRPO. They construct a surrogate objective using advantage functions and propose a policy gradient on this surrogate objective. They prove that the expected return of the updated policy monotonically increases. This is so-called Monotonic Improvement Lemma.

In low sample setting, the precise estimation of trust regions is not tractable. TRPO (Schulman et al., 2015), as a trust-region method of MDPs, explicitly induces the trust region constraints on the parameter space which might be hard to maintain in the low samples setting. To mitigate this, (Schulman et al., 2017) offer Proximal Policy Optimization (PPO), a simple extension to TRPO, which approximately retains the trust-region constraints directly on the policy space. It also significantly reduces the computation cost of TRPO, therefore it is a reasonable choice for empirical study.

Contributions: In this work, We develop a new surrogate objective and advantage function for POMDP environments. We show that policy gradient around the current policy using the new objective and advantage function results in policies whose improvements in the expected returns are bounded below. Therefore, we conserve the Monotonic Improvement Lemma. To achieve this guarantee, we show the advantage function needs to depend on three consecutive observations. Surprisingly, this matches the statement in (Azizzadenesheli et al., 2016b) which shows three consecutive observations are necessary to learn the POMDP dynamics and guarantee a regret upper bound in a model-based RL setting. In the analysis of TRPO, the construction of the trust region is independent of the episode length which results in bias estimation of the region in the episodic setting. We show that it is necessary to incorporate the length of each episode and to construct the trust region as a function of the episode lengths.

Generally, discount factors play an essential role in the construction of the trust regions. Discount factors represent how important each segment of a trajectory is therefore how critical is the role each segment in the construction of the trust region. While non of the prior works considers discount factors in the study of trust region, we further extend the analysis in this work and introduce a new notion of divergence which deploys the discount factor to construct a meaningful trust region. We also show to extend the techniques developed in this work to the prior works, e.g.,TRPO and PPO in episodic MDPs. In GTRPO, we use the same techniques as in PPO to reduce computation complexity. We apply it to a variety of RoboSchool (Schulman et al., 2017) environments, which are the extension to the MuJoCo environments (Todorov et al., 2012). We empirically study GTRPO performance on these simulated environments and report its behavior under different simulation design choices. Throughout the experiments, we observe a similar behavior of the MDP based approach PPO and POMDP based approach GTRPO. This might be due to the simplicity of the environment as well as the close similarity of current state of environments are close to MDP

2 Preliminaries

An episodic POMDP M is a tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{Y}, P_0, T, R, O, \gamma, x_T \rangle$ with latent state space $\mathcal{X}$, observation space $\mathcal{Y}$, action space $\mathcal{A}$, discount factor of $0 \leq \gamma \leq 1$ and stochastic reward distribution of $R(x, a)$ with mean $\overline{R}(x, a) = \mathbb{E}[R(x, a)]$, $\forall x \in \mathcal{X}, a \in \mathcal{A}$. x_T is the terminal state which is *accessible* from any other state, i.e. starting from any other state there is a nonzero probability mass of reaching the x_T in finite time steps. The episode terminates when the process reaches the x_T. The initial latent states are drawn from distribution P_1, then the dynamics follows stochastically as $\mathbb{P}(x'|x, a) = T(x'|x, a)$, $\forall x, x' \in \mathcal{X}, a \in \mathcal{A}$. The observation process is generated as $\mathbb{P}(y|x) = O(y|x)$, $\forall x \in \mathcal{X}, y \in \mathcal{Y}$ and a memory less policy is deployed which maps a current observation to a distribution over actions. The graphical model associated to the POMDP is illustrated in Fig. 1.

We consider a set of parameterized policies π_θ with $\theta \in \Theta$. For each (y, a) pair, let $\pi_\theta(a|y)$ denotes the conditional probability distribution of choosing action a under the policy π_θ when an observation y is observed. Furthermore, we define a random trajectory τ as a finite length $|\tau|$ sequence of events

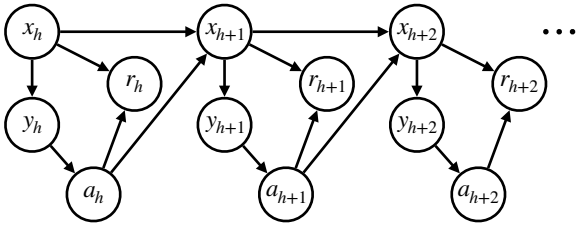


Figure 1: Graphical temporal model of a POMDP under a memory-less policy

$\{(x_1, y_1, a_1, r_1), (x_2, y_2, a_2, r_2), \dots (x_{|\tau|}, y_{|\tau|}, a_{|\tau|}, r_{|\tau|})\}$ where the termination happens at the step after $x_{|\tau|}$, i.e. $x_{|\tau|+1} = x_T$. Let $f(\tau;\theta)$, $\forall \tau \in \Upsilon$ denote the probability distribution of trajectories under policy π_θ and Υ is the set of all possible trajectories. Furthermore, $R(\tau)$ denotes the cumulative γ-discounted rewards of the trajectory $\tau \in \Upsilon$ and $f_\theta(\tau)$ denotes the probability density of the trajectory τ under policy π_θ. The agent goal is to maximize the unnormalized expected cumulative return $\eta(\theta) = \mathbb{E}_{\tau|\theta}[R(\tau)]$;

$$\theta^* = \arg\min_{\theta\in\Theta} \eta(\theta) := \int_{\tau\in\Upsilon} f(\tau;\theta) R(\tau) d\tau \qquad (1)$$

with $\pi^* = \pi(\theta^*)$ the optimal policy.

3 Policy Gradient

In this section, we study the policy gradients methods for POMDPs. Generally, the optimization problem of interest in Eq. 1 is a non-convex problem. Therefore, hill climbing methods might not converge to the global optima. While finding the optimal solution to this problem is intractable in general, we study the gradient ascent based approaches. Gradient ascent for Eq. 1 results in the policy gradient method. It is a well-known Lemma that the gradient of the expected cumulative return does not require the explicit knowledge of the dynamics but just the cumulative reward distribution (Williams, 1992; Baxter and Bartlett, 2001). This Lemma has mainly been proven through the construction of score function (see section A.1). In this section, we re-derive the same Lemma but through importance sampling since it is more related to the latter parts of this paper.

Importance sampling is a general technique for estimating the properties of a particular distribution, while only having samples generated from another distribution. One can estimate $\eta(\theta')$, $\theta' \in \Theta$, while the expectation is over the distribution induced by π_θ;

$$\begin{aligned}\eta(\theta') = \mathbb{E}_{\tau|\theta'}[R(\tau)] &= \int_{\tau\in\Upsilon} f(\tau;\theta) \left(\frac{f(\tau;\theta')}{f(\tau;\theta)} R(\tau)\right) d\tau \\ &= \mathbb{E}_{\tau|\theta}\left[\frac{f(\tau;\theta')}{f(\tau;\theta)} R(\tau)\right] \qquad (2)\end{aligned}$$

as long as for each τ that $f(\tau;\theta') > 0$ also $f(\tau;\theta) > 0$. The gradient of $\eta(\theta')$ with respect to θ' is

$$\begin{aligned}\nabla_{\theta'}\eta(\theta') &= \mathbb{E}_{\tau|\theta}\left[\frac{\nabla_{\theta'} f(\tau;\theta')}{f(\tau;\theta)} R(\tau)\right] \\ &= \mathbb{E}_{\tau|\theta}[\frac{f(\tau;\theta')}{f(\tau;\theta)} \nabla_{\theta'} \log(f(\tau;\theta')) R(\tau)]\end{aligned}$$

The gradient at $\theta' = \theta$ is;

$$\nabla_{\theta'}\eta(\theta') \mid_{\theta'=\theta} = \mathbb{E}_{\tau|\theta}\left[\nabla_\theta \log(f(\tau;\theta)) R(\tau)\right] \qquad (3)$$

Since for each trajectory τ, the $\log(f(\tau;\theta)) :=;$

$$\begin{aligned}\log\Big(P_1(x_1) O(y_1|x_1) R(r_1|x_1,a_1) \Pi_{h=2}^{|\tau|} T(x_h|x_{h-1},a_{h-1}) \\ O(y_h|x_h) R(r_h|x_h,a_h)\Big) + \log\Big(\Pi_{h=1}^{|\tau|}\pi_\theta(a_h|y_h)\Big)\end{aligned}$$

and the first part is independent θ we have;

$$\nabla_\theta \log(f(\tau;\theta)) = \nabla_\theta \log\Big(\Pi_{h=1}^{|\tau|}\pi_\theta(a_h|y_h)\Big)$$

This derivation suggest that given trajectories under a policy π_θ we can compute the gradient of the expected return with respect to the parameters of π_θ without the knowledge of the dynamics. In practice, however we are not able to compute the exact expectation. Instead we can deploy Monte Carlo sampling technique to estimate the gradient. Given m trajectories $\{\tau^1, \dots, \tau^m\}$ with elements $(x_h^t, y_h^t, a_h^t, r_h^t), \forall h \in \{1, \dots, |\tau^t|\}$ and $\forall t \in \{1, \dots, m\}$ generated under a policy π_θ, we can estimate the gradient in Eq. 3 at point θ;

$$\nabla_\theta \widehat{\eta}(\theta) = \frac{1}{m}\sum_{t=1}^{m} \nabla_\theta \log\Big(\Pi_{h=1}^{|\tau^t|}\pi_\theta(a_h^t|y_h^t)\Big) R(\tau^t) \qquad (4)$$

3.1 Natural Policy Gradient

Generally, the notion gradient depends on the metric space it lives. Given a pre-specified Riemannian metric, a gradient direction is defined. When the metric is Euclidean, the notion of gradient reduces to the standard gradient (Lee, 2006). This general notion of gradient adjusts the standard gradient direction based on the local curvature induced by the Riemannian manifold of interest. Valuable knowledge of the curvature assists to find an ascent direction which might conclude to big ascend in the objective function. This approach is also interpreted as a trust region method where we are interested in assuring that the ascent steps do not change the objective beyond a safe region where the local curvature is still valid. In general, a valuable manifold might not be given, and we need to adopt one. Fortunately, when the objective function is an expectation over a parameterized distribution, Amari (2016) recommend employing a Riemannian metric, induced

by Fisher information. This choice of metric results in a well knows notion of gradient, so-called *natural gradient*. For the objective function in 1, the Fisher information matrix is defined as follows;

$$F(\theta) := \int_{\tau\in\Upsilon} f(\tau;\theta)\left[\nabla_\theta \log\left(f(\tau;\theta)\right)\nabla_\theta \log\left(f(\tau;\theta)\right)^\top\right] d\tau \tag{5}$$

Natural gradients are firstly deployed by Kakade (2002) for RL in MDPs. Consequently, the direction of the gradient with respect to F is defined as $F(\theta)^{-1}\nabla_\theta(\eta(\theta))$. One can compute the inverse of this matrix to come up with the direction of the natural gradient. Since neither storing the Fisher matrix is always possible nor computing the inverse is practical, direct utilization of $F(\theta)^{-1}\nabla_\theta(\eta(\theta))$ is not feasible. As also used in TRPO, we suggest to first deploy $\mathcal{D}_{KL}$ divergence substitution technique and then conjugate gradient method to tackle the computation and storage bottlenecks.

Lemma 1. *Under some regularity conditions;*

$$\nabla^2_{\theta'}\mathcal{D}_{KL}(\theta,\theta')|_{\theta'=\theta} = F(\theta) \tag{6}$$

with $\mathcal{D}_{KL}(\theta,\theta') := -\int_{\tau\in\Upsilon} f(\tau;\theta)\log\left(f(\tau;\theta')/f(\tau;\theta)\right)d\tau$

Proof of Lemma 1 in Subsection A.2. In practice, it is not feasible to compute the expectation of neither the Fisher information matrix nor the $\mathcal{D}_{KL}$ divergence, but their empirical estimates. Given m trajectories

$$\nabla^2_{\theta'}\widehat{\mathcal{D}}_{KL}(\theta,\theta')|_{\theta'=\theta} = -\frac{1}{m}\nabla^2_{\theta'}\sum_{t=1}^{m}\Big[\log\Big(\Pi_{h=1}^{|\tau^t|}\pi_{\theta'}(a_h^t|y_h^t)\Big) - \log\Big(\Pi_{h=1}^{|\tau^t|}\pi_{\theta}(a_h^t|y_h^t)\Big)\Big]|_{\theta'=\theta} = -\frac{1}{m}\nabla^2_{\theta'}\sum_{t=1}^{m}\sum_{h=1}^{|\tau^t|}\log\left(\frac{\pi_{\theta'}(a_h^t|y_h^t)}{\pi_\theta(a_h^t|y_h^t)}\right)$$

This derivation of $\mathcal{D}_{KL}$ is common between MDPs and POMDPs. The analysis in most of the state-of-the-art policy gradient methods, e.g. TRPO, PPO, are dedicated to infinite horizon MDPs, while almost all the experimental studies are in the episodic settings. Therefore the estimator used in these methods;

$$\nabla^2_{\theta'}\widehat{\mathcal{D}}^{TRPO}_{KL}(\theta,\theta')|_{\theta'=\theta} = -\frac{1}{\sum_t^m|\tau^t|}\nabla^2_{\theta'}\sum_{t=1}^{m}\sum_{h=1}^{|\tau^t|}\log\left(\frac{\pi_{\theta'}(a_h^t|y_h^t)}{\pi_\theta(a_h^t|y_h^t)}\right)$$

is a bias estimation of the $\mathcal{D}_{KL}$ in episodic settings.

Remark 1. *[$\mathcal{D}_{KL}$ vs ${\mathcal{D}_{KL}}^{TRPO}$] The use of $\mathcal{D}_{KL}$ instead of ${\mathcal{D}_{KL}}^{TRPO}$ is motivated by theory also intuitively recommended. A small change in the policy at the beginning of short episodes does not make a drastic shift in the distribution of that trajectory but might cause radical shifts when the horizon length is long. Therefore, for longer horizons, the trust region needs to shrink. Consider two trajectories, one long and one short. The $\mathcal{D}_{KL}\leq\delta$ induces a region which allows higher changes in the policy for short trajectory while limiting changes in long trajectory. While ${\mathcal{D}_{KL}}^{TRPO}\leq\delta$ induces the region which does not convey the length of each trajectory and look at the samples as they happened in stationary distribution of an infinite horizon* MDP*. Consider a simpe game. where at the beginning of the learning, when the policy is not good, the agent dies at early stages of the episodes, and the game terminates. In this case, the trust region under $\mathcal{D}_{KL}$ is vast and allows for the more substantial change in the policy space, while again ${\mathcal{D}_{KL}}^{TRPO}$ does not consider the length of the episode. On the other hand, toward the end of learning, when the agent plays a good policy, the length of the horizon grows, and small changes in the policy cause drastic changes in the trajectory distribution. Therefore the trust region shrinks again, and tiny changes in the policy space are allowed, which is again captured by $\mathcal{D}_{KL}$ but not ${\mathcal{D}_{KL}}^{TRPO}$.*

Compatible Function Approximation As it is mentioned before, one way of computing the direction of the natural gradient is to estimate the $\widehat{\mathcal{D}}_{KL}$ and use conjugate gradient methods to find $F^{-1}\nabla_\theta(\eta)$. There is also another interesting way to estimate $F^{-1}\nabla_\theta(\eta)$, which is based on compatible function approximation methods. Kakade (2002) study this approach in the context of MDPs. In the following, we develop this approach for POMDPs. Consider a feature map $\phi(\tau)$ in some ambient space defined on Γ. We approximate the return $R(\tau)$ by a linear function ω on the feature representation $\phi(\tau)$, i.e.,

$$\min_\omega \epsilon(\omega)\ s.t.;\ \epsilon(\omega) : \int_{\tau\in\Upsilon} f(\tau,\theta)[\phi(\tau)^\top\omega - R(\tau)]^2 d\tau$$

To find the optimal ω we take the gradient of $\epsilon(\omega)$ and set it to zero;

$$0 = \nabla_\omega\epsilon(\omega)|_{\omega=\omega^*} = \int_{\tau\in\Upsilon} 2f(\tau,\theta)\phi(\tau)[\phi(\tau)^\top\omega^* - R(\tau)]d\tau$$

For the optimality,

$$\int_{\tau\in\Upsilon} f(\tau,\theta)\phi(\tau)\phi(\tau)^\top\omega^* d\tau = \int_{\tau\in\Upsilon} f(\tau,\theta)\phi(\tau;\theta)R(\tau)d\tau$$

If we consider the $\phi(\tau) = \nabla_\theta\log\Big(\Pi_{h=1}^{|\tau|}\pi_\theta(a_h|y_h)\Big)$, the LHS of this equation is $F(\theta)\omega^*$. Therefore

$$F(\theta)\omega = \nabla_\theta\eta(\theta) \Longrightarrow \omega^* = F(\theta)^{-1}\nabla\rho$$

In practice, either of the discussed approaches of computing the natural gradient is applicable, and one needs to choose one of them depending on the problem and

application at hand. Due to the close relationship between $\mathcal{D}_{KL}$ and Fisher information matrix Lemma.1 and also the fact that the Fisher matrix is equal to second order Taylor expansion of $\mathcal{D}_{KL}$, instead of considering the area $\| (\theta - \theta')^\top F (\theta - \theta') \|_2 \leq \delta$, or $\| (\theta - \theta')^\top \nabla^2_{\theta'} \mathcal{D}_{KL}(\theta, \theta')|_{\theta'=\theta} (\theta - \theta') \|_2 \leq \delta$, we can approximately consider $\mathcal{D}_{KL}(\theta, \theta') \leq \delta/2$. The relationship between these three approaches toward trust-regions is used throughout this paper.

4 TRPO for POMDPs

In this section we develop the MDP analysis in Kakade and Langford (2002); Schulman et al. (2015) to POMDPs, propose GTRPO, and derive a guarantee on its monotonic improvement property. We prove the monotonic improvement property using $\mathcal{D}_{KL}$. We also develop a new discount factor depended divergence and provide the same guarantee under the new divergence.

The $\mathcal{D}_{KL}$ divergence and Fisher information matrix in Eq. 6, Eq. 5 do not convey the effect of the discount factor. Consider a setting with a small discount factor γ. In this setting, we do not mind drastic distributional changes in the latter part of episodes. Therefore, we desire to have a even wider trust region and allow bigger changes for later parts of trajectories. This is a valid intuition and in the following, we re-derive the $\mathcal{D}_{KL}$ divergence by also incorporating γ. Let τ_1^h denote the elements in τ up to time step h; we rewrite $\eta(\theta)$ as follows;

$$\eta(\theta) = \int_{\tau \in \Upsilon} f(\tau;\theta) R(\tau) d\tau = \int_{\tau \in \Upsilon} \sum_{h=1}^{|\tau|} f(\tau_1^h;\theta) \gamma^h r_h(\tau) d\tau$$

Following the Amari (2016) reasoning for Fisher information of each component of the sum, we derive a γ dependent divergence;

$$\mathcal{D}_\gamma(\pi_\theta, \pi_{\theta'}) = \sum_{h=1}^{\tau_{\max}} \gamma^h \mathcal{D}_{KL}\left(\tau_1^h \sim f(\cdot;\pi_{\theta'}), \tau_1^h \sim f(\cdot;\pi_\theta)\right) \tag{7}$$

For some upper bound on trajectory length $\tau_{\max}$. This divergence less penalizes the distribution mismatch in the later part of trajectories. Similarly, taking into account the relationship between KL divergence and Fisher information we have discount factor γ dependent definition of the Fisher information;

$$F_\gamma(\theta) := \int_{\tau \in \Upsilon} \sum_{h=1}^{\tau_{\max}} \gamma^h f(\tau_1^h;\theta) \left[\nabla_\theta \log\left(f(\tau_1^h;\theta)\right) \nabla_\theta \log\left(f(\tau_1^h;\theta)\right)^\top\right] d\tau$$

In the following we develop GTRPO monotonic improvement guarantee under both $\mathcal{D}_\gamma$ and $\mathcal{D}_{KL}$.

4.1 Advantage function on the hidden states

In the following let π_θ denote policy under which we collect data, so-called the *current policy*, and $\pi_{\theta'}$ the policy which we evaluate its performance, so-called the *new policy*. Generally, any policy on the on observation space is transferable to a policy on the latent states as follows; $\pi(a|x) = \int_{y \in \mathcal{Y}} \pi(a|y) O(y|x) dy$ for each pair of (x, a). Consider the case where the agent also observes the latent state, i.e. POMDP $\rightarrow$MDP. Since the dynamics on the latent states is MDP, we define the advantage function on the latent states: at time step h of episode;

$$\widetilde{A}_\pi(a, x, h) = \mathbb{E}_{x' \sim T(x'|x,a,h)}\left[r(x, a, h) + \gamma \widetilde{V}_\pi(x', h) - \widetilde{V}_\pi(x, h)\right]$$

Where $\widetilde{V}_\pi$ denote the value function of underlying MDP of latent states when a policy π is deployed. For this choice of advantage function we have Therefore,

$$\begin{aligned}
&\mathbb{E}_{\tau \sim f(\tau, \pi_{\theta'})}\left[\sum_h^{|\tau|} \gamma^h \widetilde{A}_{\pi_\theta}(x_h, a_h, h)\right] \\
&= \mathbb{E}_{\tau \sim f(\tau, \pi_{\theta'})}\left[\sum_h^{|\tau|} \gamma^h \left[r(x_h, a_h, h) + \gamma \widetilde{V}_{\pi_\theta}(x_{h+1}, h) - \widetilde{V}_{\pi_\theta}(x_h, h)\right]\right] \\
&= \mathbb{E}_{\tau \sim f(\tau, \pi_{\theta'})}\left[\sum_h^{|\tau|} \gamma^h r_h\right] - \mathbb{E}_{x_0 \sim P_1(x)}\left[\widetilde{V}_{\pi_\theta}(x_0)\right] = \eta(\pi_{\theta'}) - \eta(\pi_\theta)
\end{aligned}$$

If we have the advantage function of the current policy π_θ and sampled trajectories from $\pi_{\theta'}$, we could compute the improvement in the expected return $\eta(\pi_{\theta'}) - \eta(\pi_\theta)$ therefore maximize it. In this case we also could potently just maximize the expected return for $\pi_{\theta'}$ without incorporating any knowledge from π_θ. Instead, in practice, we do not have sampled trajectories from the new policy $\pi_{\theta'}$, rather we have sampled trajectories from the current policy π_θ. Therefore, we might be interested in maximizing the following surrogate objective function since we can compute it;

$$\widetilde{L}_{\pi_\theta}(\pi_{\theta'}) := \eta(\pi_\theta) + \mathbb{E}_{\tau \sim \pi_\theta, a \sim \pi_{\theta'}(a'_h|x_h,h)}\left[\sum_h^{|\tau|} \gamma^h \widetilde{A}_{\pi_\theta}(x_h, a'_h, h)\right]$$

For infinite horizon MDPs when O is an identity matrix, i.e. at each time step $x = y$, Kakade and Langford (2002); Schulman et al. (2015) show that optimizing $\widetilde{L}_{\pi_\theta}(\pi_{\theta'})$ over θ' can provide an improvement in the expected discounted return. They derive a lower bound on this improvement if the $\mathcal{D}_{KL}$ divergence of $\pi_{\theta'}$ and

π_θ for all x's is bounded. In the following, we extend these analyses to the general class of environments, i.e. POMDPs and show such guarantees are conserved.

Generally, in POMDPs, when classes of memory-less policies are regarded, neither Q nor V functions are well-defined as they are for MDP by the Bellman equation. In the following, we define two quantities similar to the Q and V in MDPs while for simplicity use the same Q and V notation for them. The conditional value and Q-value functions of POMDPs

$$V_\pi(y_h, h, y_{h-1}, a_{h-1}) := \mathbb{E}_\pi\left[\sum_h^H \gamma^h r_h | y_h = y, y_{h-1} = y_{h-1}, a^{h-1} = a_{h-1}\right]$$

$$Q_\pi(y_{h+1}, a, y_h, h) := \tag{8}$$

$$\mathbb{E}_\pi\left[\sum_h^H \gamma^h r_h | y_h = y, y_{h+1} = y_{h+1}, a^h = a\right] \tag{9}$$

For $h = 0$ we relax the conditioning on y_{h-1} for V_π and simply denote it as $V_\pi(y, 0)$. Deploying these two quantities, we define the advantage function as follows;

$$\begin{aligned} &A_\pi(y_{h+1}, y_h, a, h, y_{h-1}, a_{h-1}) \\ &= Q_\pi(y_{h+1}, a_h, y_h, h) - V_\pi(y_h, h, y_{h-1}, a_{h-1}) \\ &= r_\pi(y_{h+1}, y_h, a_h, h) + \gamma V_\pi(y_{h+1}, h+1, y_h, a_h) \\ &\quad - V_\pi(y_h, h, y_{h-1}, a_{h-1}) \end{aligned}$$

Here $r_\pi(y_{h+1}, y, a, h)$ denotes the reward random variable conditioned on current observation, action as well as one step successor observation. It worth noting that since the reward process is not Markovian, this conditioning does not make it independent of past or future. Furthermore, we defined the following surrogate objective function;

$$L_{\pi_\theta}(\pi_{\theta'}) = \eta(\pi_\theta) + \mathbb{E}_{\tau\sim\pi_\theta, a\sim\pi_{\theta'}(a|y)} \sum_h^{|\tau|} \gamma^h A_{\pi_\theta}(y_{h+1}, y_h, a_h, h, y_{h-1}, a_{h-1}) \tag{10}$$

Similar to MDPs, one can compute and maximize this surrogate objective function in Eq. 10 by just having sampled trajectories and advantage function of the current policy π_θ.

Lemma 2. *The improvement in expected return, $\eta(\pi_{\theta'}) - \eta(\pi_\theta)$, is as follows;*

$$\begin{aligned} &\mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_h^{|\tau|} \gamma^h A_{\pi_\theta}(y_{h+1}, y_h, a_h, h, y_{h-1}, a_{h-1}) \\ &\quad = \eta(\pi_{\theta'}) - \eta(\pi_\theta) \end{aligned}$$

Proof of Lemma 2 in Subsection A.3.

In practice, one can estimate the advantage function $A_{\pi_\theta}(y_{h+1}, y, a, h, y_{h-1}, a_{h-1})$ by approximating $Q_{\pi_\theta}(y_{h+1}, a, y_h, h)$ and $V_{\pi_\theta}(y_h, h, y_{h-1}, a_{h-1})$ using on-policy data of π_θ. It is worth noting that $L_{\pi_\theta}(\pi_{\theta'})$ has also following nice property from the derivation of policy gradient theorem

$$L_{\pi_\theta}(\pi_\theta) = \eta(\pi_\theta), \text{ and } \nabla_{\theta'} L_{\pi_\theta}(\pi_{\theta'})|_{\pi_\theta=\pi_\theta} = \nabla_\theta \eta(\pi_\theta)$$

In the following we show that maximizing $L_{\pi_\theta}(\pi_{\theta'})$ over θ' results in a lower bound on the improvement $\eta(\pi_{\theta'}) - \eta(\pi_\theta)$ when π_θ and $\pi_{\theta'}$ are close under $\mathcal{D}_{KL}$ or $\mathcal{D}_\gamma$ divergence. Lets define the averaged advantage function

$$\begin{aligned} &\overline{A}_{\pi_\theta,\pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) = \\ &\quad \mathbb{E}_{a\sim\pi_{\theta'}}\left[A_{\pi_\theta}(y_{h+1}, y_h, h, a, y_{h-1}, a_{h-1})\right] \end{aligned}$$

also the maximum span of the averaged advantage function and its discounted sum as follows;

$$\begin{aligned} \epsilon' &= \max_{\tau\in\Upsilon} \overline{A}_{\pi_\theta,\pi_\theta}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) \\ \epsilon &= \max_{\tau\in\Upsilon} \sum_h^{|\tau|} \gamma^h \overline{A}_{\pi_\theta,\pi_\theta}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) \end{aligned}$$

Theorem 1 (Monotonic Improvement Guarantee)**.** *For two π_θ and $\pi_{\theta'}$, construct $L_{\pi_\theta}(\pi_{\theta'})$, then*

$$\begin{aligned} \eta(\pi_{\theta'}) &\geq L_{\pi_\theta}(\pi_{\theta'}) - \epsilon TV\left(\tau \sim f(\cdot; \pi_{\theta'}), \tau \sim f(\tau; \pi_\theta)\right) \\ &\geq L_{\pi_\theta}(\pi_{\theta'}) - \epsilon\sqrt{\frac{1}{2}\mathcal{D}_{KL}\left(\pi_{\theta'}, \pi_\theta\right)} \end{aligned}$$

and also

$$\eta(\pi_{\theta'}) \geq L_{\pi_\theta}(\pi_{\theta'}) - \epsilon'\sqrt{\mathcal{D}_\gamma\left(\pi_\theta, \pi_\theta\right)}$$

Proof. Following the result in the Lemma 2 we have

$$\begin{aligned} &\eta(\pi_{\theta'}) = \eta(\pi_\theta) \\ &\quad + \mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_h^{|\tau|} \gamma^h A_{\pi_\theta}(y_{h+1}, y_h, h, a_h, y_{h-1}, a_{h-1}) \end{aligned}$$

therefore,

$$\begin{aligned} &\eta(\pi_{\theta'}) - L_\pi(\pi_{\theta'}) \\ &= \mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_h^{|\tau|} \gamma^h A_{\pi_\theta}(y_{h+1}, y_h, h, a_h, y_{h-1}, a_{h-1}) \\ &- \mathbb{E}_{\tau\sim\pi_\theta, a'_h\sim\pi_{\theta'}(a|y)} \sum_h^{|\tau|} \gamma^h A_{\pi_\theta}(y_{h+1}, y_h, h, a'_h, y_{h-1}, a_{h-1}) \end{aligned}$$

following the definition of $\widetilde{A}_{\pi_\theta,\pi_{\theta'}}$

$$\begin{aligned} \eta(\pi_{\theta'}) - L_{\pi_\theta}(\pi_{\theta'}) = \int_\tau &\left(f(\tau; \pi_{\theta'}) - f(\tau; \pi_\theta)\right) \\ &\sum_h^{|\tau|} \gamma^h \overline{A}_{\pi_\theta,\pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) d\tau \end{aligned}$$

Deploying the maximum span of averaged advantage function ϵ and the Pinsker's inequality we have

$$\begin{aligned}|\eta(\pi_{\theta'}) - L_{\pi_\theta}(\pi_{\theta'})| \\ &\leq \epsilon TV\left(\tau \sim f(\cdot;\pi_{\theta'}), \tau \sim f(\tau;\pi_\theta)\right) \\ &\leq \epsilon\sqrt{\frac{1}{2}\mathcal{D}_{KL}\left(\tau \sim f(\cdot;\pi_{\theta'}), \tau \sim f(\tau;\pi_\theta)\right)}\end{aligned}$$

Which results in the first part of the theorem. On the other hand

$$\eta(\pi_{\theta'}) - L_{\pi_\theta}(\pi_{\theta'}) = \int_\tau \sum_{h=1}^{|\tau|} \left(f(\tau_1^h;\pi_{\theta'}) - f(\tau_1^h;\pi_\theta)\right) \gamma^h \overline{A}_{\pi_\theta,\pi_{\theta'}}(y_{h+1}, y_h, h, y_{h-1}, a_{h-1}) d\tau$$

Deploying the definition of ϵ' and the Pinsker's inequality again we have

$$\begin{aligned}|\eta(\pi_{\theta'}) - L_{\pi_\theta}(\pi_{\theta'})| \\ &\leq \epsilon' \sum_{h=1}^{\tau_{\max}} \gamma^h TV\left(\tau_1^h \sim f(\cdot;\pi_{\theta'}), \tau_1^h \sim f(\cdot;\pi_\theta)\right) \\ &\leq \epsilon' \sum_{h=1}^{\tau_{\max}} \gamma^h \sqrt{\frac{1}{2}\mathcal{D}_{KL}\left(\tau_1^h \sim f(\cdot;\pi_{\theta'}), \tau_1^h \sim f(\cdot;\pi)\right)} \\ &\leq \epsilon' \sqrt{\mathcal{D}_\gamma(\pi_\theta, \pi_{\theta'})}\end{aligned}$$

and the second part of the theorem goes through. □

Algorithm 1 GTRPO

1: Initial π_0, and ϵ'
2: Choice of divergence: $\mathcal{D}_{KL}$ or $\mathcal{D}_\gamma$
3: **for** episode = 1 until convergence **do**
4: Estimate the advantage function $\widehat{A}$
5: Construct the surrogate objective $\widehat{L}_{\pi_{t-1}}(\pi)$
6: Find the next policy

$$\begin{aligned}\pi_t = \arg\max_\pi L_{\pi_{t-1}}(\pi) \\ s.c \qquad \widehat{D}(\pi_{t-1}, \pi) \leq \delta\end{aligned}$$

The Theorem. 1 recommend optimizing $L_{\pi_\theta}(\pi_{\theta'})$ over $\pi_{\theta'}$ around the vicinity defined by $\mathcal{D}_{KL}$ or $\mathcal{D}_\gamma$ divergences Therefore, given the current policy π_θ we are interested in either of the following optimization:

$$\max_{\theta'} L_{\pi_\theta}(\pi_{\theta'}) - C\sqrt{\mathcal{D}_{KL}\left(\pi_\theta, \pi_{\theta'}\right)}$$
$$\max_{\theta'} L_{\pi_\theta}(\pi_{\theta'}) - C'\sqrt{\mathcal{D}_\gamma\left(\pi_\theta, \pi_{\theta'}\right)}$$

Where C and C' are the problem dependent constants which also are the nobs to restrict the trust region. Similar to TRPO, using C and C' as they are might result in tiny changes in policy. Therefore, for practical purposes, we view them as the nobs to restrict the trust region denoted by δ, δ' and turn these optimization problems to constraint optimization problems;

$$\max_{\theta'} L_{\pi_\theta}(\pi_{\theta'}) \quad s.t. \quad \mathcal{D}_{KL}\left(\pi_\theta, \pi_{\theta'}\right) \leq \delta$$
$$\max_{\theta'} L_{\pi_\theta}(\pi_{\theta'}) \quad s.t. \quad \mathcal{D}_\gamma\left(\pi_\theta, \pi_{\theta'}\right)) \leq \delta'$$

Taking into account the relationship between the KL divergence and Fisher information, we can also approximate these two optimization up to their second order Taylor expansion of the constraints;

$$\max_{\theta'} L_{\pi_\theta}(\pi_{\theta'}) \quad s.t. \quad \frac{1}{2}\| \left(\theta' - \theta\right)^\top F \left(\theta' - \theta\right) \|_2 \leq \delta$$
$$\max_{\theta'} L_{\pi_\theta}(\pi_{\theta'}) \quad s.t. \quad \frac{1}{2}\| \left(\theta' - \theta\right)^\top F_\gamma \left(\theta' - \theta\right) \|_2 \leq \delta'$$

These analyses provide insights to design similar algorithm as TRPO and PPO for the general class of problems, i.e., POMDPs.

5 Experiments

Extension to PPO: Usually, in low sample setting, estimating and then constructing the trust region is hard, especially when the region dependents on the inverse of the estimated Fisher matrix or optimizing over the non-convex function of θ' in KL divergence. Therefore, trusting the estimated trust region is questionable. While TRPO construct the trust region in the parameter space, its final goal is to keep the new policy close to the current policy, i.e., small $\mathcal{D}_{KL}\left(\pi_\theta, \pi_{\theta'}\right)$ or $\mathcal{D}_\gamma\left(\pi_\theta, \pi_{\theta'}\right)$. Proximal Policy Optimization (PPO) is instead proposed to impose the structure of the trust region directly onto the policy space. This method approximately translates the constraints developed in TRPO to the policy space. It penalized the gradients of the objective function when the policy starts to operate beyond the region of trust by setting it to zero.

$$\begin{aligned}\mathbb{E}[\min\{&\frac{\pi_{\theta'}(a|x)}{\pi_\theta(a|x)}\widetilde{A}_{\pi_\theta}(a,x) \\ &, clip(\frac{\pi_{\theta'}(a|x)}{\pi_\theta(a|x)}; 1-\delta_L, 1+\delta_U)\widetilde{A}_{\pi_\theta}(a,x)\}]\end{aligned}$$

We dropped the h dependency in the advantage function since this approach is for the infinite horizon. If the advantage function is positive, and the importance weight is above $1+\delta_U$ this objective function saturates. When the advantage function is negative, and the importance weight is below $1-\delta_L$ this objective function saturates again. In either case, when the objective function saturates, the gradient of this objective

function is zero therefore further development in that direction is obstructed. This approach, despite its simplicity, approximates the trust region effectively and substantially reduce the computation cost of TRPO. **Note:** In the original PPO paper $\delta_U = \delta_L$.

Following the TRPO, the clipping trick ensures that the importance weight, derived from estimation of $\mathcal{D}_{KL}$ does not go beyond a certain limit. i.e. each $|\log \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)}| \leq \nu$. This results in

$$1 - \delta_L := \exp(-\nu) \leq \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)} \leq 1 + \delta_U := \exp(\nu) \tag{11}$$

As discussed in the Remark. 1 we propose a principled change in the clipping such that it matches Eq. 6 and conveys information about the length of episodes; $|\log \frac{\pi_\theta(a|y)}{\pi_{\theta'}(a|y)}| \leq \frac{\nu}{|\tau|}$; therefore for $\alpha := \exp(\nu)$

$$1 - \delta_L := \alpha^{-1/|\tau|} \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta_\theta(a|y)}} \leq 1 + \delta_U := \alpha^{1/|\tau|} \tag{12}$$

This change ensures more restricted clipping for longer trajectories, while softer for shorter ones. Moreover, as it is suggested in theorem. 1, and the definition of $\mathcal{D}_\gamma(\pi_\theta, \pi_{\theta'})$ in Eq. 7, we propose a further extension in the clipping to conduct information about the discount factor. For a sample at time step h of an episode we have $|\log \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)}| \leq \frac{\nu}{|\tau|\gamma^h}$. Therefore;

$$1 - \delta_L := \exp(-\frac{\nu}{|\tau|\gamma^h}) \leq \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)} \leq 1 + \delta_U := \exp(\frac{\nu}{|\tau|\gamma^h})$$
$$\rightarrow \alpha^{-1/|\tau|}\alpha^{-1/\gamma^h} \leq \frac{\pi_{\theta'}(a|y)}{\pi_\theta(a|y)} \leq \alpha^{1/|\tau|}\alpha^{1/\gamma^h} \tag{13}$$

As it is interpreted, for deeper parts in the episode, we make the clipping softer and allow for larger changes in policy space. This means, we are more restricted at the beginning of trajectories compared to the end of trajectories. The choice of γ and α are critical here. In practical implementation of RL algorithm, as also theoretically suggested by Jiang et al. (2015); Lipton et al. (2016) we usually choose discount factors smaller than the one for depicted in the problem. Therefore, the discount factor we use in practice is much smaller than the true one specially when we deploy function approximation. Therefore, instead of keeping γ^h in Eq. 13, since the true γ in practice is unknown and can be arbitrary close to 1, we substitute it with a maximum value, i.e.

$$1 - \delta_L := \max\{\alpha^{-1/|\tau|}, 1 - \beta\} \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta_\theta(a|y)}}$$
$$\leq 1 + \delta_U := \max\{\alpha^{1/|\tau|}, 1 + \beta\} \tag{14}$$

RoboSchool, a variant to MuJoCo: In the experimental study, we first started to analyze the behavior of the plain PPO agent but observe that the environment enforces a short termination which results in significantly short trajectories. We relaxed this hard threshold and analyzed PPO Section C Subsection C.2. We deploy the analysis in Eq. 12 and Eq. 14, apply the suggested changes to the plain PPO and examine its performance in the variety of different parameters and environments Subsection C.3 and Subsection C.4. In Section E, we apply the GTRPO on the variety of different environments and analyze its behavior. As it is provided in the Appendix, along with the mentioned experimental studies, we have done an extensive study on a variety of different settings to present a more detailed understanding of policy gradient methods. Throughout the experiments, we observe a similar behavior of the MDP based approach PPO and POMDP based approach GTRPO. This might be due to the simplicity of the environment as well as the close similarity of current state of environments are close to MDP. Along the course of the experimental study, we realized that the environment set-up and the deployed reward shaping require a critical and detailed modification to make the test-bed suitable for further studies. Section D and Subsection C.3.2.

6 Conclusion

In this paper, we propose GTRPO, a trust region policy optimization method for general class of POMDPs. We consider memoryless policies and show that each policy update derived by GTRPO monotonically improves the expected return. We develop a new advantage function for POMDPs which depends on three consecutive observation. The dependency on three consecutive observations also matches the claim in Azizzadenesheli et al. (2016b) which shows learning the model and minimizing the regret requires modeling three consecutive observations. GTRPO deploys this advantage function to perform the policy updates. Additionally, we show how to utilize the analyses in this work and extend the infinite horizon MDP based policy gradient methods, TRPO and PPO, to finite horizon MDPs. Finally, the same way that PPO extends the analyses in TRPO, we extend GTRPO analyses and make it computationally more efficient. We implement this extension and empirical study its behavior along with PPO on Roboschool environments.

References

Aleksandrov, V. M., S. V. I. . S. V. V.
1968. Stochastic optimaization. *Engineering Cybernetics*, 5(11-16):229–256.

Amari, S.-i.
2016. *Information geometry and its applications*. Springer.

Azizzadenesheli, K., A. Lazaric, and A. Anandkumar
2016a. Open problem: Approximate planning of pomdps in the class of memoryless policies. In *Conference on Learning Theory*, Pp. 1639–1642.

Azizzadenesheli, K., A. Lazaric, and A. Anandkumar
2016b. Reinforcement learning of pomdps using spectral methods. *arXiv preprint arXiv:1602.07764*.

Baxter, J. and P. L. Bartlett
2001. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350.

Bernstein, J., Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar
2018. signsgd: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*.

Jiang, N., A. Kulesza, S. Singh, and R. Lewis
2015. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, Pp. 1181–1189.

Kakade, S. and J. Langford
2002. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, Pp. 267–274.

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

Kostrikov, I.
2018. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr`.

Lee, J. M.
2006. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra
2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lipton, Z. C., K. Azizzadenesheli, A. Kumar, L. Li, J. Gao, and L. Deng
2016. Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al.
2015. Human-level control through deep reinforcement learning. *Nature*.

Montúfar, G., K. Ghazi-Zahedi, and N. Ay
2015. Geometry and determinism of optimal stationary control in partially observable markov decision processes. *arXiv preprint arXiv:1503.07206*.

Rubinstein, R. Y.
1969. Some problems in monte carlo optimization. *Ph.D. thesis*.

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz
2015. Trust region policy optimization. In *International Conference on Machine Learning*, Pp. 1889–1897.

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

Sutton, R. S., A. G. Barto, F. Bach, et al.
1998. *Reinforcement learning: An introduction*. MIT press.

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

Todorov, E., T. Erez, and Y. Tassa
2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Pp. 5026–5033. IEEE.

Vlassis, N., M. L. Littman, and D. Barber
2012. On the computational complexity of stochastic controller optimization in pomdps. *ACM Transactions on Computation Theory (TOCT)*, 4(4):12.

Williams, R. J.
1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

A Appendix

A.1 Score function

It is well know that the gradient of the expected cumulative return can be approximated without knowledge of the model dynamics (Williams, 1992; Baxter and Bartlett, 2001). We restate this development of the previous works for POMDPs from the point of view of score function

$$\nabla_\theta \eta(\theta) = \nabla_\theta \int_{\tau\in\Upsilon} f(\tau;\theta)R(\tau)d\tau = \int_{\tau\in\Upsilon} \nabla_\theta f(\tau;\theta)R(\tau)d\tau = \int_{\tau\in\Upsilon} f(\tau;\theta)\nabla_\theta \log(f(\tau;\theta))R(\tau)d\tau$$

for a single trajectory of $\tau = \{(x_1, y_1, a_1, r_1), (x_2, y_2, a_2, r_2), \ldots, (x_{|\tau|}, y_{|\tau|}, a_{|\tau|}, r_{|\tau|})\}$, $R(\tau) = \sum_{h=1}^{|\tau|} r_h|\tau$. while

$$\begin{aligned} f(\tau;\theta) = {} & P_1(x_1)O(y_1|x_1)\pi_\theta(a_1|y_1)R(r_1|x_1,a_1) \\ & \prod_{h=2}^{|\tau|} T(x_h|x_{h-1},a_{h-1})O(y_h|x_h)\pi_\theta(a_h|y_h)R(r_h|x_h,a_h) \end{aligned}$$

Therefore, for the gradient of the log we have;

$$\begin{aligned} & \nabla_\theta \log(f(\tau;\theta)) \\ & \quad = \nabla_\theta \log\left(P_1(x_1)O(y_1|x_1)R(r_1|x_1,a_1)\Pi_{h=2}^{|\tau|}T(x_h|x_{h-1},a_{h-1})O(y_h|x_h)R(r_h|x_h,a_h)\right) \\ & \qquad + \nabla_\theta \log\left(\Pi_{h=1}^{|\tau|}\pi_\theta(a_h|y_h)\right) \end{aligned}$$

since the first part is independent of θ, its derivative is zero. Therefore we have

$$\nabla_\theta \eta(\theta) = \int_{\tau\in\Upsilon} f(\tau;\theta)\nabla_\theta \log\left(\Pi_{h=1}^{|\tau|}\pi_\theta(a_h|y_h)\right)R(\tau)d\tau$$

It is clear through Monte Carlo sampling theorem that given a set of m trajectories $\{\tau^1, \ldots, \tau^m\}$ with elements $(x_h^t, y_h^t, a_h^t, r_h^t), \forall h \in \{1, \ldots, |\tau^t|\}$ and $\forall t \in \{1, \ldots, m\}$, the empirical mean of the gradient is

$$\widehat{\nabla}_\theta(\eta) = \frac{1}{m}\sum_{t=1}^{m} \nabla_\theta \log\left(\Pi_{h=1}^{|\tau^t|}\pi_\theta(a_h^t|y_h^t)\right)R(\tau^t) \tag{15}$$

which does not depend on underlying dynamic except through cumulative reward $R(\tau)$

A.2 Proof of Lemma 1

Proof.

$$\begin{aligned} \nabla^2_{\theta'} KL(\theta,\theta')|_{\theta'=\theta} := & -\nabla^2_{\theta'} \int_{\tau\in\Upsilon} f(\tau;\theta)\left[\log\left(f(\tau;\theta')\right) - \log\left(f(\tau;\theta)\right)\right]d\tau|_{\theta'=\theta} \\ = & -\int_{\tau\in\Upsilon} f(\tau;\theta)\nabla^2_{\theta'}\log\left(f(\tau;\theta')\right)d\tau|_{\theta'=\theta} \\ = & -\int_{\tau\in\Upsilon} f(\tau;\theta)\nabla_{\theta'}\left[\frac{1}{f(\tau;\theta')}\nabla_{\theta'}f(\tau;\theta')\right]d\tau|_{\theta'=\theta} \\ = & \int_{\tau\in\Upsilon} f(\tau;\theta)\left[\frac{1}{f(\tau;\theta')^2}\nabla_{\theta'}f(\tau;\theta')\nabla_{\theta'}f(\tau;\theta')^\top\right]d\tau|_{\theta'=\theta} \\ & -\int_{\tau\in\Upsilon} f(\tau;\theta)\left[\frac{1}{f(\tau;\theta')}\nabla^2_{\theta'}f(\tau;\theta')\right]d\tau|_{\theta'=\theta} \\ = & \int_{\tau\in\Upsilon} f(\tau;\theta)\left[\frac{1}{f(\tau;\theta')^2}\nabla_{\theta'}f(\tau;\theta')\nabla_{\theta'}f(\tau;\theta')^\top\right]d\tau|_{\theta'=\theta} \\ & -\nabla^2_{\theta'}\int_{\tau\in\Upsilon} f(\tau;\theta')d\tau|_{\theta'=\theta} = F(\theta) \end{aligned} \tag{16}$$

□

A.3 Proof of Lemma 2

Proof. With a few substitutions in the first term we have;

$$
\begin{aligned}
&\mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_{h}^{|\tau|} \gamma^h A_{\pi_\theta}(y_{h+1}, y_h, a_h, h, y_{h-1}, a_{h-1}) \\
&= \mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_{h}^{|\tau|} \gamma^h [r_{\pi_\theta}(y_{h+1}, a_h, y_h, h) \\
&\quad + \gamma V_{\pi_\theta}(y_{h+1}, h+1, y_h, a_h) - V_{\pi_\theta}(y_h, h, y_{h-1}, a_{h-1})] \\
&= \mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_{h}^{|\tau|} \gamma^h \left[r_{\pi_\theta}(y_{h+1}, a_h, y_h, h)\right] - \mathbb{E}\left[V_{\pi_\theta}(y_0, 0)\right] \\
&= \mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_{h}^{|\tau|} \gamma^h \left[r_{\pi_\theta}(a_h, y_h, h)\right] - \eta(\pi_\theta) \\
&= \mathbb{E}_{\tau\sim\pi_{\theta'}} \sum_{h}^{|\tau|} \gamma^h \left[r_{\pi_\theta}(y_{h+1}, a_h, y_h, h)\right] - \eta(\pi_\theta) \\
&= \eta(\pi_{\theta'}) - \eta(\pi_\theta)
\end{aligned}
$$

□

B Experimental Study

In the following sections we empirically study the sequence of changes that the theoretical analyses suggest to make on PPO. The code for PPO that we used can be found https://github.com/ikostrikov/pytorch-a2c-ppo-acktr(Kostrikov, 2018). We use https://blog.openai.com/roboschool/ environments for our experiments.

We have primarily experimented 4 propositions:

1. **PPO-length-γ-dependent**: Study of PPO when we incorporate length dependent and discount factor dependent way of constructing the truest region.

2. **GTRPO**: The study of GTRPO

3. **Environment Choice**: The study of the Roboschool environments

4. **PPO through signSGD**: A further study of the trust region construction through sign gradient methods, e.g., signSGD (Bernstein et al., 2018).

Note: In all of the following plots, unless otherwise mentioned, the x-axis represents the number of steps that the model has seen, and the y-axis represents the reward. The graph has been normalized and made smooth for better visualization. The label of the plots are the name of the corresponding **roboschool** environment.

C PPO-length-γ-dependent

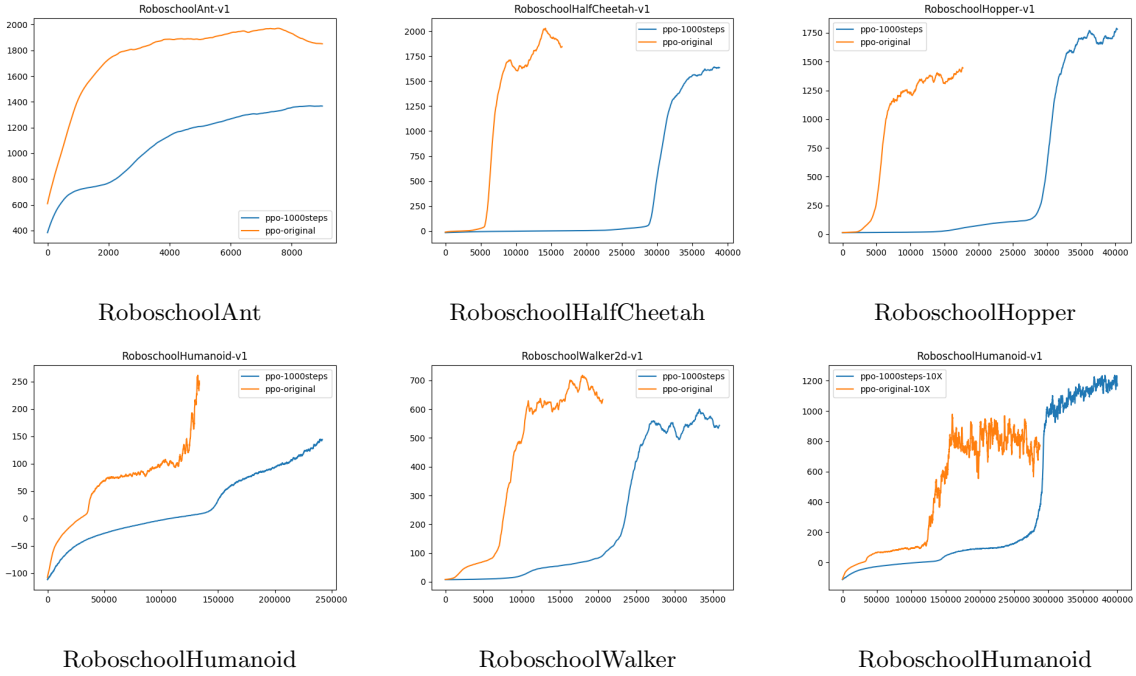
C.1 ppo-len-dep

In PPO, in order to provide a better approximation through Monte Carlo sampling, the policy updates take place after one(or more) full episodes of experiences. The motivation of this variation of PPO lies in the intuition that the trust region deployed for policy update should depend on the number of steps in the episode. When the model is in its initial stages of learning, it is most probable that the length of episodes is very low. Therefore more significant changes in the policy space should be allowed. As the RL agent becomes more experienced, it acquires a better policy, deals better with the environment, and thus, the episode lengths increase. At this point, it is essential that the updates stay small since even small changes might result in a drastic shift in the trajectory distributions. Therefore we modify the update as mentioned in Eq. 12

The original implementation of PPO in https://github.com/ikostrikov has a fixed number (128 to be exact) for the maximum time-steps per episode. Almost all the models would hit this limit very quickly. Setting a significantly low threshold for episode length reduces the capability to study the behavior of RL algorithms. Therefore we increased the number this maximum threshold to 1000. We denote this PPO on this environment as **ppo-1000steps**. The original variant (vanilla PPO) is referred as **ppo-original**.

C.2 ppo-original v/s ppo-1000steps

Before proceeding to the experiments with **ppo-1000steps**, we compare the performances of **ppo-1000steps** and **ppo-original**. Fig. 2 provide the mentioned comparison. We observe that changing episode length threshold does not affect the final performance of either of these two by far. We observe that the convergent value is achieved faster in case of **ppo-original**, this might be caused by the fact that **ppo-original** is allocated more number of updates than **ppo-1000steps** for a given number of total time-steps (x-axis).



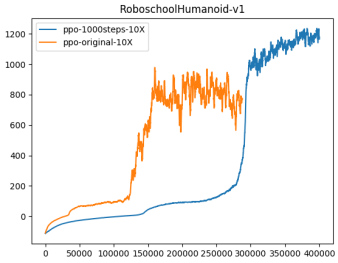
RoboschoolAnt RoboschoolHalfCheetah RoboschoolHopper

RoboschoolHumanoid RoboschoolWalker RoboschoolHumanoid

Figure 2: We run PPO on 5 environments with two design choice. **ppo-original** denote the PPO agent with maximum length of each episode is equal to 128. **ppo-1000steps** denotes the PPO agent when the maximum length of each episode is equal to 1000 .

C.2.1 ppo-original-10X v/s ppo-1000steps-10X

From Fig.2 one can observe that for the environment **humanoid**, **ppo-1000steps** has hard time to converge under the same number of steps. Therefore, we increased the number of episode 10 times and rerun both of them Fig. 3.

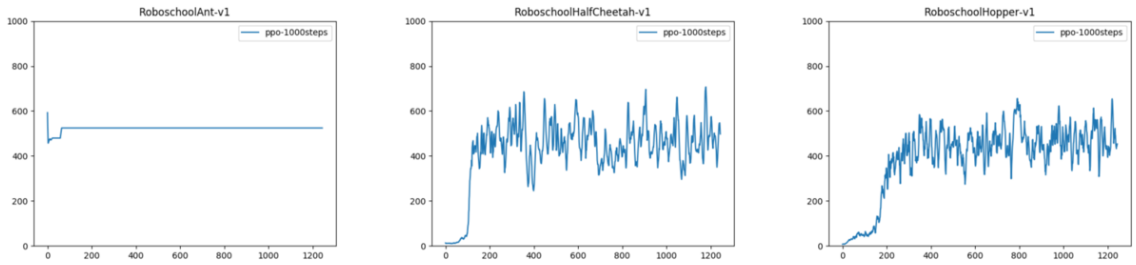


RoboschoolHumanoid

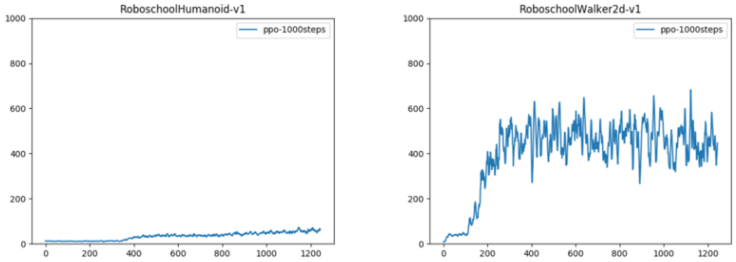
Figure 3: We run PPO on humanoid environment with two design choice but this time for 10 times longer. **ppo-original** denote the PPO agent with maximum length of each episode is equal to 128. **ppo-1000steps** denotes the PPO agent when the maximum length of each episode is equal to 1000 .

C.2.2 episode lengths

In this subsection, we study the episode length distribution induced by **ppo-1000steps** on various environments after we set the maximum threshold of per-episode steps to 1000. In Fig. 4 we plot the episode lengths of **ppo-1000steps** as it learns the policy. The x-axis represents the number of episodes seen by the model, and the y-axis represents the episode length..



RoboschoolAnt RoboschoolHalfCheetah RoboschoolHopper

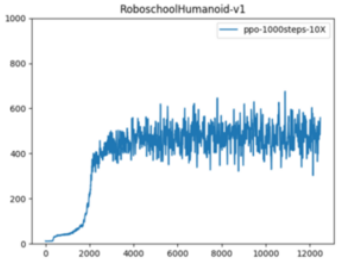


RoboschoolHumanoid RoboschoolWalker

Figure 4: The episode length distribution induced by **ppo-1000steps** on various environments

C.2.3 ppo-1000steps-10X

Similar as before, we run the humanoid also for 10 times longer time steps. Fig. 5



RoboschoolHumanoid

Figure 5: The episode length distribution induced by **ppo-1000steps** on humanoid environment when we run it for 10 times longer

C.3 ppo-1000steps vs ppo-1000steps-len-dep

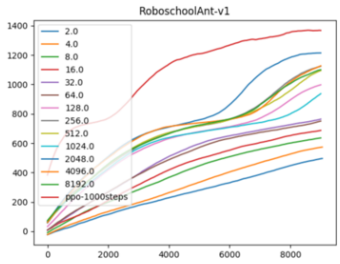
As it is discussed in Section 5, the PPO (Schulman et al., 2017) deploys clipping to make the policy updates maximal while conservative, i.e., for some threshold α_x on states of a MDP

$$\alpha_x \leq \frac{\pi_{\theta'}(a|x)}{\pi_{\theta_\theta(a|x)}} \leq \alpha_x$$

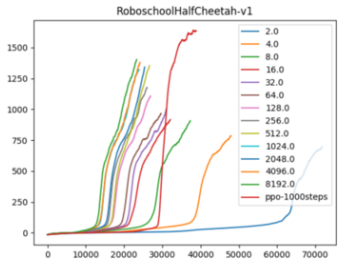
As mentioned in the Eq. 12 for episodic setting, the trust region should depend on the length of trajectories. We re-state the Eq. 12 here

$$1 - \delta_L = \alpha^{-1/|\tau|} \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta_\theta(a|y)}} \leq 1 + \delta_U = \alpha^{1/|\tau|}$$

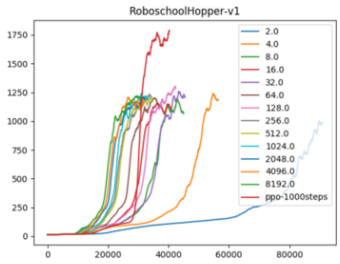
We compare the performances of **ppo-1000steps** and **ppo-1000steps-len-dep** which is same as PPO except the clipping δ_L and δ_U are defined as in Eq. 12 . In Fig. 6, as usual, x-axis represents the number of time stepss seen by the model, and y-axis represents rewards at each time step. The legend denotes the values of the α.



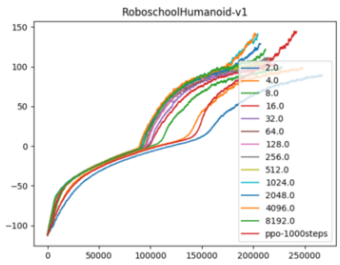
RoboschoolAnt



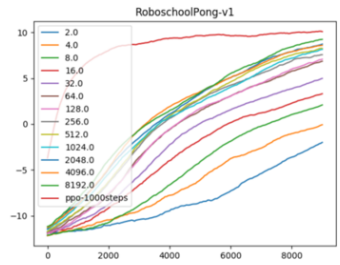
RoboschoolHalfCheetah



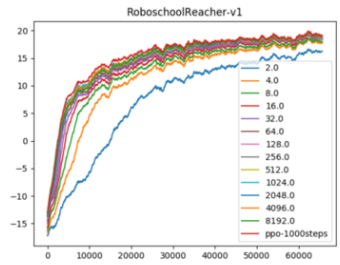
RoboschoolHopper



RoboschoolHumanoid



RoboschoolPong



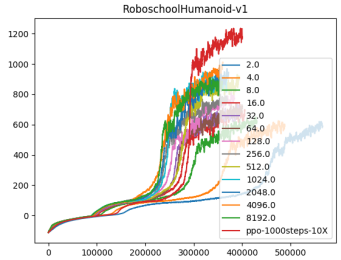
RoboschoolReacher

RoboschoolWalker

Figure 6: We run PPO on variety of environments in Robo-School. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for length dependent trust region construction in Eq. 12, **ppo-1000steps-len-dep**, and variety of different choices of α

C.3.1 Running 10X episodes for humanoid

The **humanoid** environment of **roboschool** takes longer to converge than the other environments. Same as before, we run it for 10X more episodes to observe the behaviour ant convergent values, Fig. 7

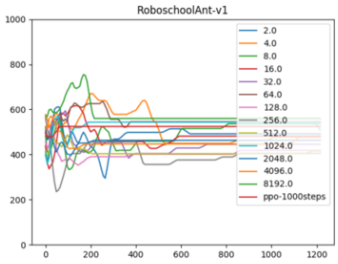


RoboschoolHumanoid

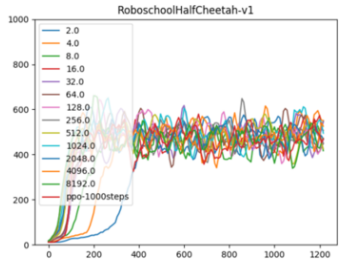
Figure 7: We run PPO on Humanoid environments in Robo-School for 10 times longer than other environments. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for length dependent trust region construction in Eq. 12, **ppo-1000steps-len-dep**, and variety of different choices of α

C.3.2 episode length

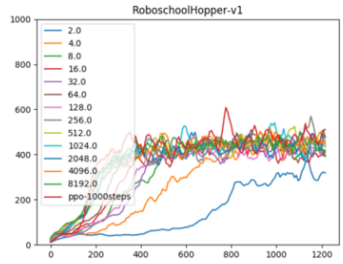
We further study the episode length to see if there are any changes to how the episode length modulates over the training period when we deploy **ppo-1000steps-len-dep** with a variety of α's. The x-axis represents the number of episodes seen by the model, y-axis represents the number of steps in that episode, and the legend represents the value of δ(see above). We observe that there is no significant change in the behaviour of episode length.



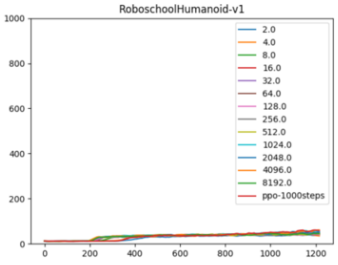
RoboschoolAnt



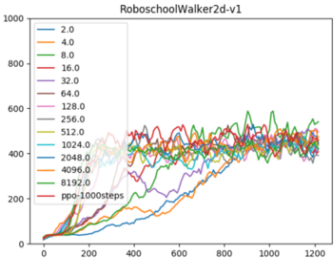
RoboschoolHalfCheetah



RoboschoolHopper



RoboschoolHumanoid



RoboschoolWalker

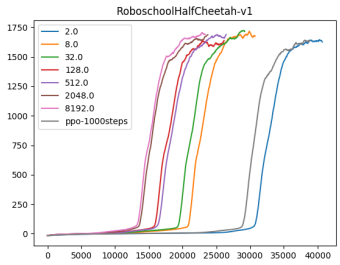
Figure 8: We run PPO on variety of environments in Robo-School. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for length dependent trust region construction in Eq. 12, **ppo-1000steps-len-dep**, and variety of different choices of α. This figure represents the episode length behaviour over the course of training.

C.4 ppo-1000steps vs ppo-1000steps-len-γ-dep

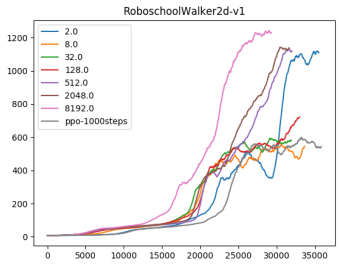
In the previous section, we study the PPO behaviour when we followed the the $\mathcal{D}_{KL}$ divergence definition and Eq. 12, i.e., **ppo-1000steps-len-dep**. In this subsection, we study the trust region suggested by $\mathcal{D}_\gamma$ the discount factor dependent trust region construction, and Eq. 14

$$1-\delta_L := \max\{\alpha^{-1/|\tau|}, 1-\beta\} \leq \frac{\pi_{\theta'}(a|y)}{\pi_{\theta_\theta(a|y)}} \leq 1+\delta_U := \max\{\alpha^{1/|\tau|}, 1+\beta\}$$

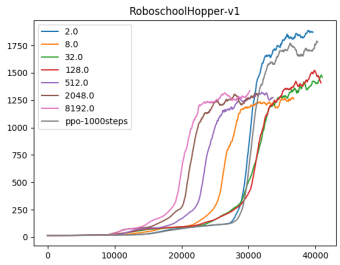
We study the behaviour of **ppo-1000steps-len-γ-dep**. We set $1-\delta_L := min(1-\beta, \left(\frac{1}{\alpha}\right)^{\frac{1}{|\tau|}})$ and $1+\delta_L := max(1+\beta, \alpha^{\frac{1}{|\tau|}})$.



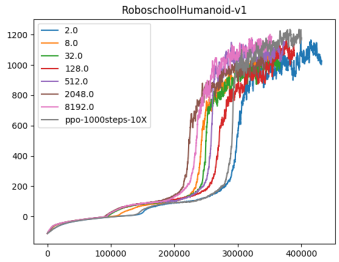
RoboschoolHalfCheetah



RoboschoolWalker



RoboschoolHopper



RoboschoolHumanoid

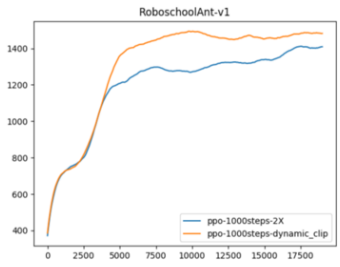
Figure 9: We run PPO on variety of environments in Robo-School. **ppo-1000steps** denote the PPO agent with maximum length of each episode is equal to 1000. The remaining plots are for **ppo-1000steps-len-γ-dep** and variety β choices

C.5 ppo-1000steps-dynamic-clip

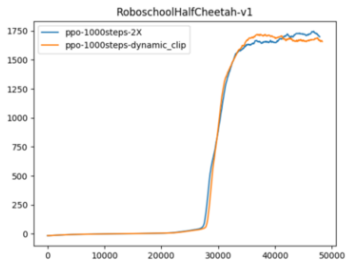
In the original parameters set by Ilya Kostrikov, the total number of frames was set as 10e6 while $\delta_U = \delta_L = 0.1$ and followed by the Eq.11. The clipping parameter of 0.1 is useful for start of training. When a good policy is learnt, making the trust region more conservative and shrinking the clipping parameter to smaller value might be helpful to find be better policy. For this study, we first train plain PPO of **ppo-1000steps** with on clip of 0.1 for 10e6 frames, and then train it with clipping parameter 0.05 for the next 10e6 frames. We express the empirical results for both **ppo-original** with threshold of 128 as the maximum length of episodes, and **ppo-1000steps** with threshold of 1000 as the maximum length of episodes.

C.5.1 max episode lengths = 1000

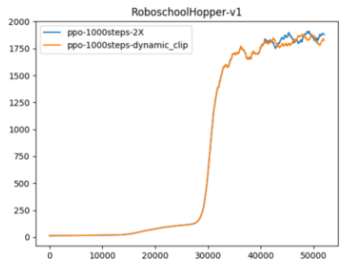
The following plots, Fig. 10 are for experiments run with maximum episode length set as 1000 steps. In the legend **ppo-1000steps-2X** denotes running the vanilla **ppo-1000steps** for 2X number of episodes; **ppo-1000steps-dynamic_clip** denotes model with the above mentioned changes.



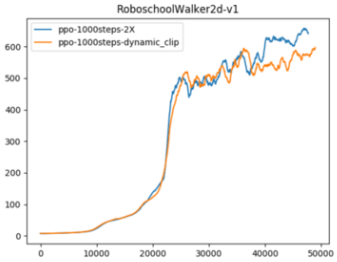
RoboschoolAnt



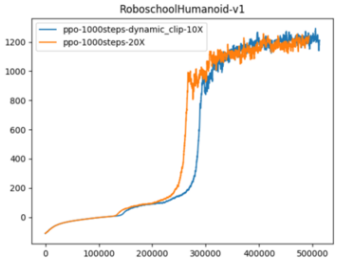
RoboschoolHalfCheetah



RoboschoolHopper



RoboschoolWalker

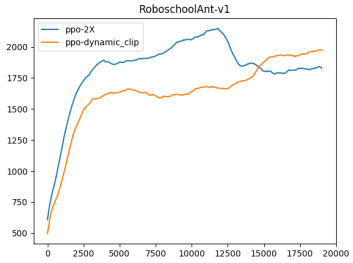


RoboschoolHumanoid

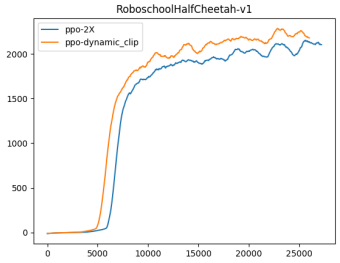
Figure 10: We run **ppo-1000steps** for 2 times longer and denote it as **ppo-1000steps-2X**. We also run **ppo-1000steps-dynamic_clip** which is **ppo-1000steps-2X** except for the first $10e6$ steps the clipping parameter is 0.1 and for the second $10e6$ it is set to 0.05

C.5.2 max episode length = 128

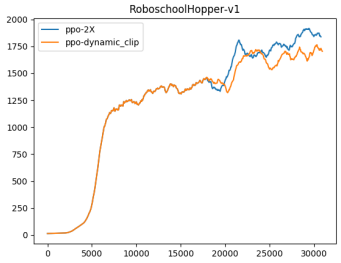
The following plots in Fig. 11 we run the same experiments as Subsection C.5.1 but for threshold on maximum length of episode set to 128 steps.



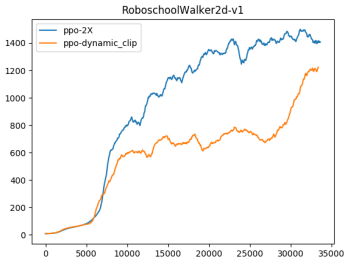
RoboschoolAnt



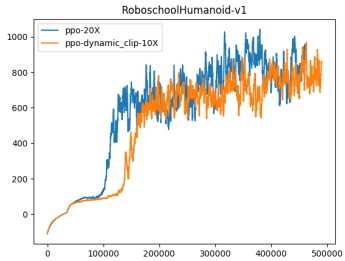
RoboschoolHalfCheetah



RoboschoolHopper



RoboschoolWalker



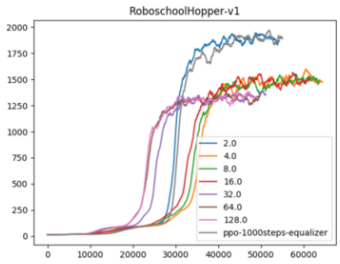
RoboschoolHumanoid

Figure 11: We run **ppo-original** for 2 times longer and denote it as **ppo-2X**. We also run **ppo-dynamic_clip** which is **ppo-2X** except for the first $10e6$ steps the clipping parameter is 0.1 and for the second $10e6$ it is set to 0.05

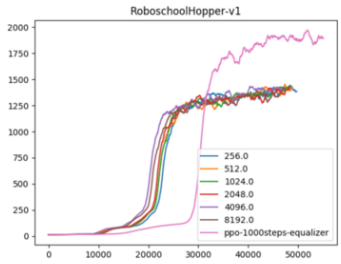
C.6 ppo-1000steps-equalizer vs ppo-len-γ-dep-equalizer running

In this subsection, we study the same empirical setting as the subsection C.4 but instead of running the algorithms for the same count of episodes we run them for the same number of interactions with the environment. We study the behaviour of **ppo-1000steps-len-γ-dep** and **ppo-1000steps-len** when we run both for the same number of time steps and denote them **ppo-1000steps-len-γ-dep-equalizer** and **ppo-1000steps-len-equalizer**. We run **ppo-1000steps-len-γ-dep-equalizer** for a fixed $\beta = 0.1$ and also $\beta = 0.1$ for a variety of α's. We experiment these for the cases when the threshold on the maximum length is 1000 as well as 128.

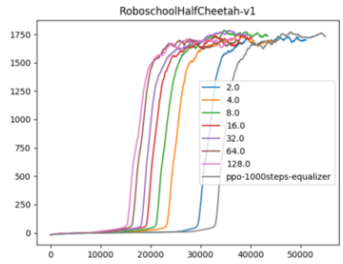
C.6.1 $\alpha = 0.1$ and episode length = 1000



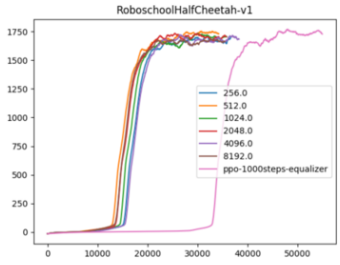
RoboschoolHopper-1



RoboschoolHopper-2



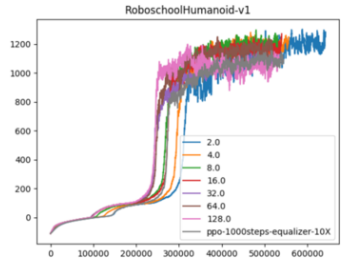
RoboschoolHalfCheetah-1



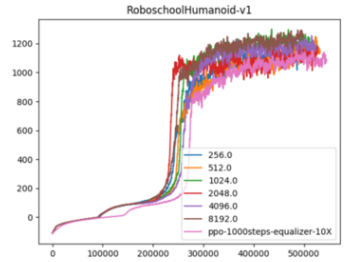
RoboschoolHalfCheetah-2

RoboschoolWalker-1

RoboschoolWalker-2



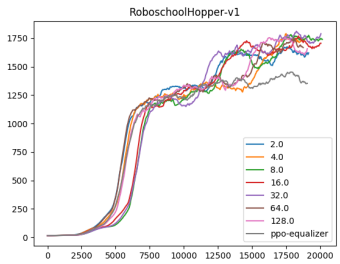
RoboschoolHumanoid-1



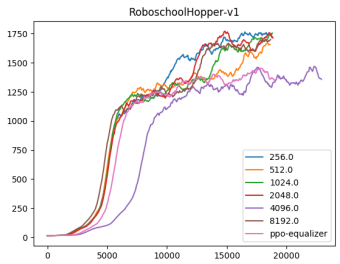
RoboschoolHumanoid-2

Figure 12: We run **ppo-1000steps** and **ppo-1000steps-len-γ-dep** for the same number of interaction when the threshold on the maximum length 1000 and call them **ppo-1000steps-equalizer** and **ppo-1000steps-len-γ-dep-equalizer**. We keep $\beta = 0.1$ and vary α

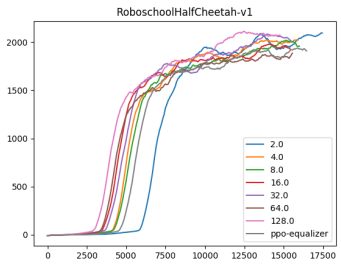
C.6.2 for $\beta = 0.1$ and episode length = 128



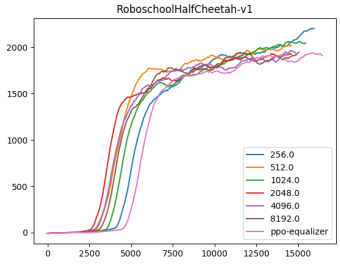
RoboschoolHopper-1



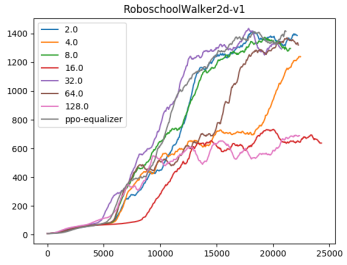
RoboschoolHopper-2



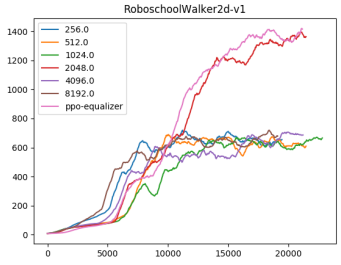
RoboschoolHalfCheetah-1



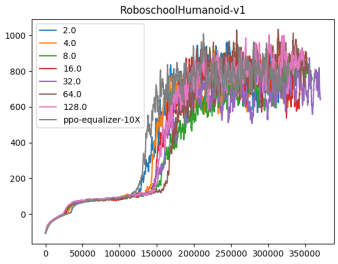
RoboschoolHalfCheetah-2



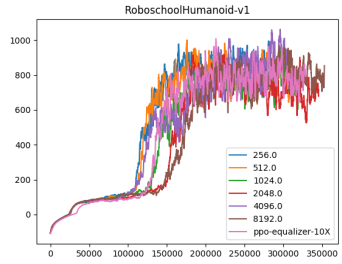
RoboschoolWalker-1



RoboschoolWalker-2



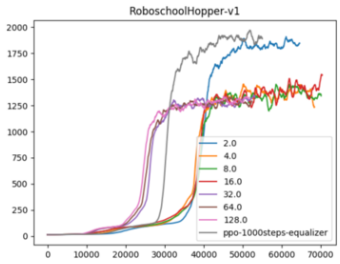
RoboschoolHumanoid-1



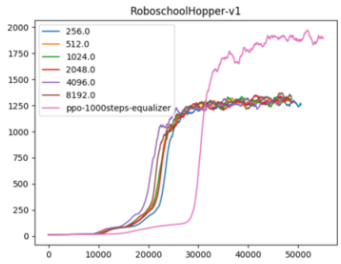
RoboschoolHumanoid-2

Figure 13: We run **ppo-original** and **ppo-original-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.1$ and vary α

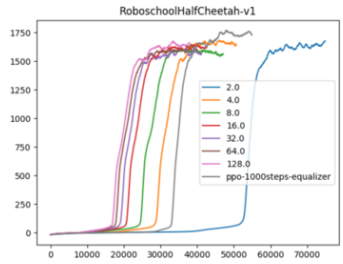
C.6.3 $\beta = 0.05$ and episode length $= 1000$



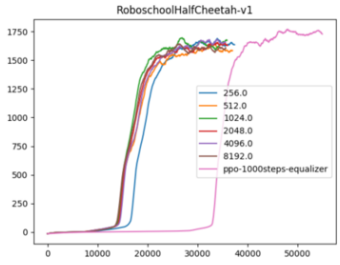
RoboschoolHopper-1



RoboschoolHopper-2



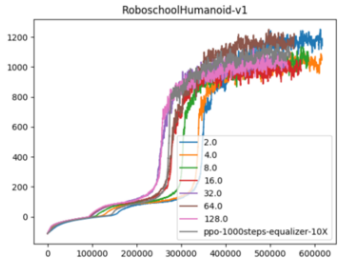
RoboschoolHalfCheetah-1



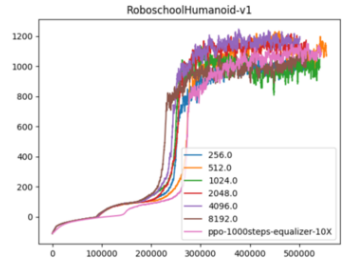
RoboschoolHalfCheetah-2

RoboschoolWalker-1

RoboschoolWalker-2



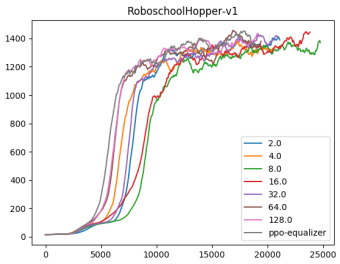
RoboschoolHumanoid-1



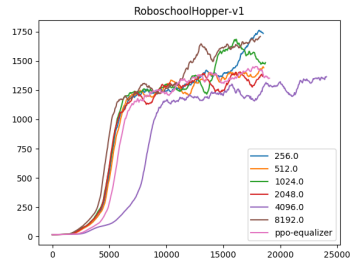
RoboschoolHumanoid-2

Figure 14: We run **ppo-original** and **ppo-original-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.01$ and vary α

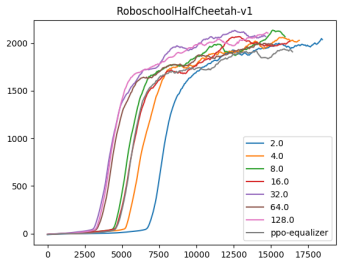
C.6.4 $\beta = 0.05$ and episode length = 128



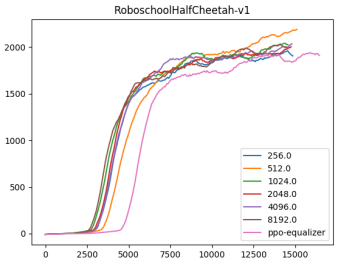
RoboschoolHopper-1



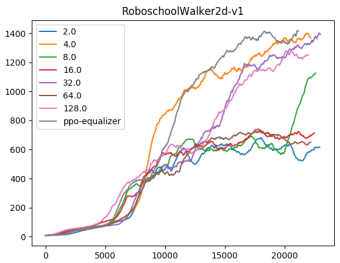
RoboschoolHopper-2



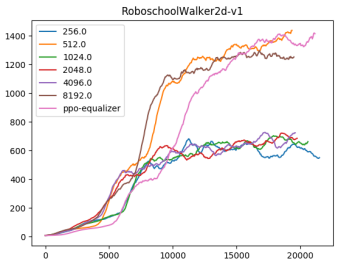
RoboschoolHalfCheetah-1



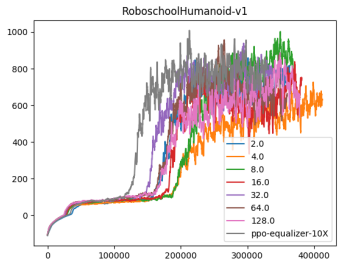
RoboschoolHalfCheetah-2



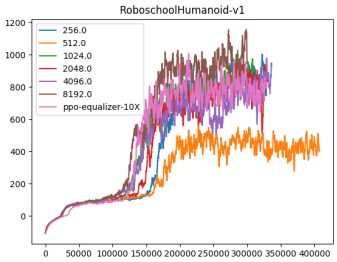
RoboschoolWalker-1



RoboschoolWalker-2



RoboschoolHumanoid-1

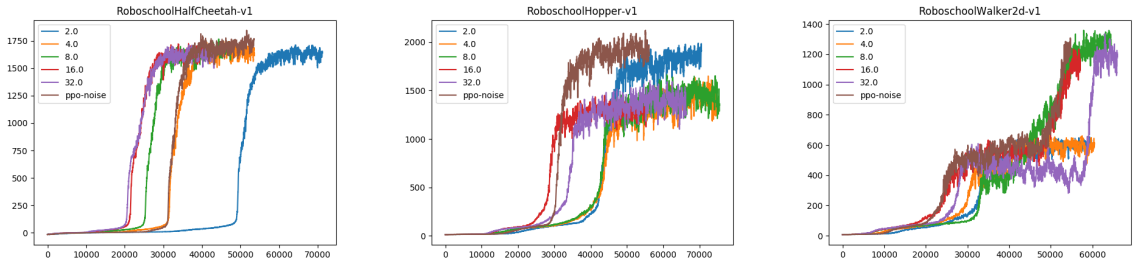


RoboschoolHumanoid-2

Figure 15: We run **ppo-original** and **ppo-original-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.05$ and vary α

C.7 ppo-1000steps-equalizer vs ppo-len-γ-dep-equalizer

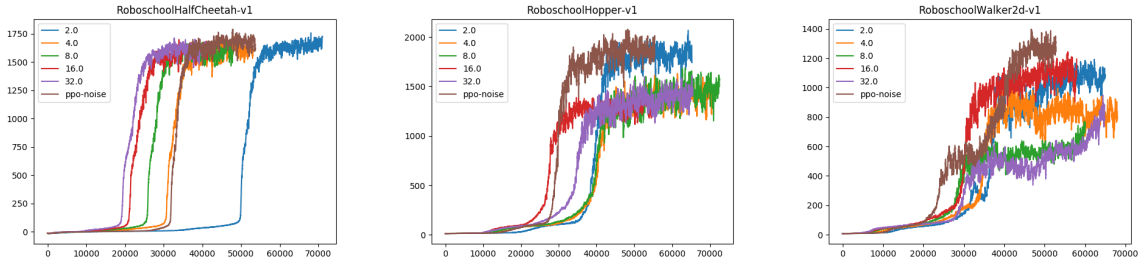
C.7.1 for $\beta = 0.05$ and $stddev = 0.001$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 16: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.05$ and vary α. But we also add gaussian noise to the observation, with stadard deviation *stddev*.

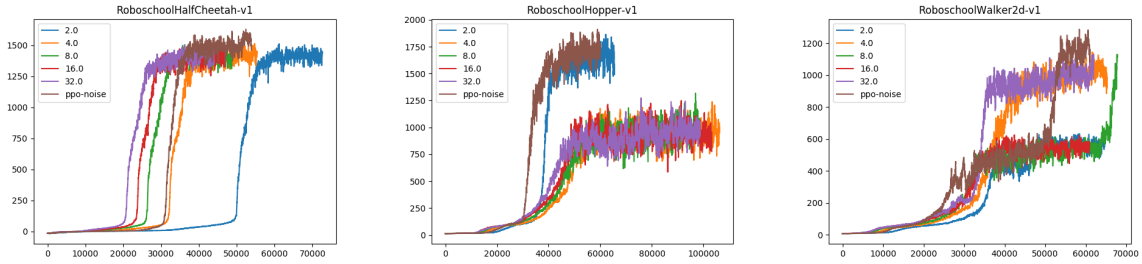
C.7.2 for $\beta = 0.05$ and $stddev = 0.01$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 17: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.05$ and vary α. But we also add gaussian noise to the observation, with stadard deviation *stddev*.

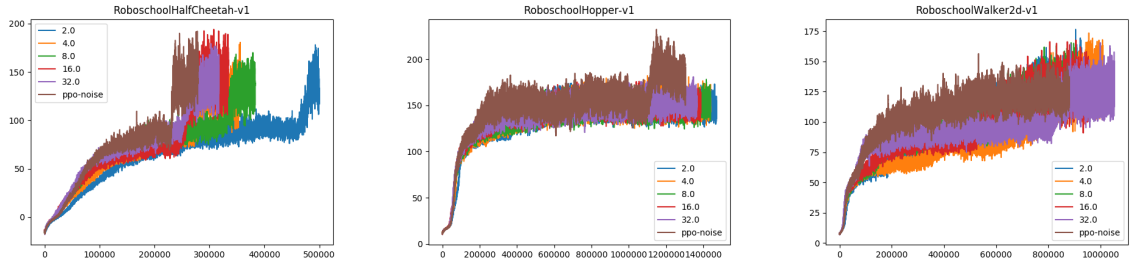
C.7.3 for $\beta = 0.05$ and $stddev = 0.1$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 18: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.05$ and vary α. But we also add gaussian noise to the observation, with stadard deviation *stddev*.

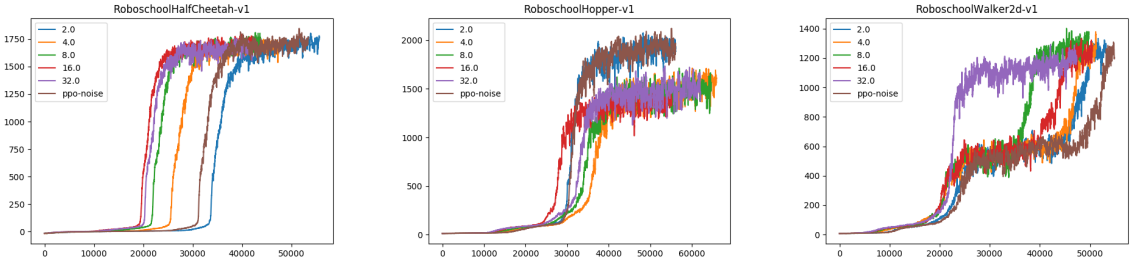
C.7.4 for $\beta = 0.05$ and $stddev = 1.0$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 19: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.05$ and vary α. But we also add gaussian noise to the observation, with stadard deviation $stddev$.

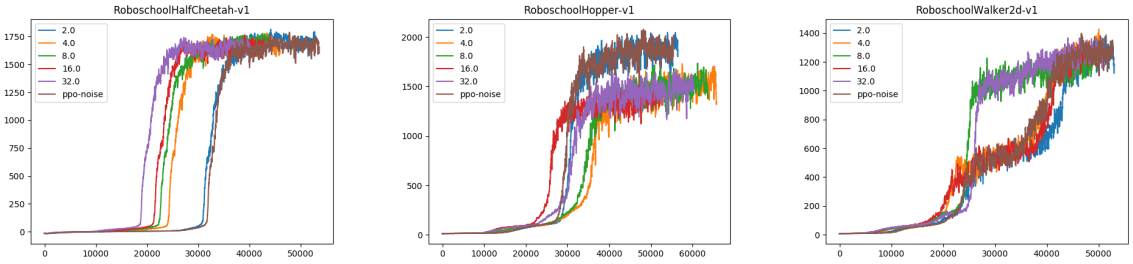
C.7.5 for $\beta = 0.1$ and $stddev = 0.001$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 20: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.1$ and vary α. But we also add gaussian noise to the observation, with stadard deviation $stddev$.

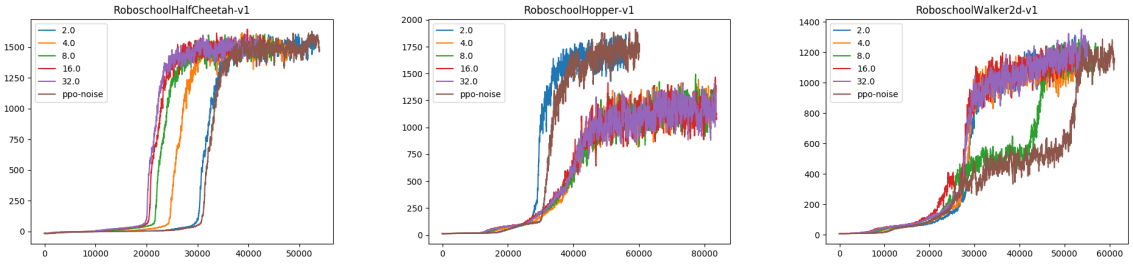
C.7.6 for $\beta = 0.1$ and $stddev = 0.01$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 21: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.1$ and vary α. But we also add gaussian noise to the observation, with stadard deviation $stddev$.

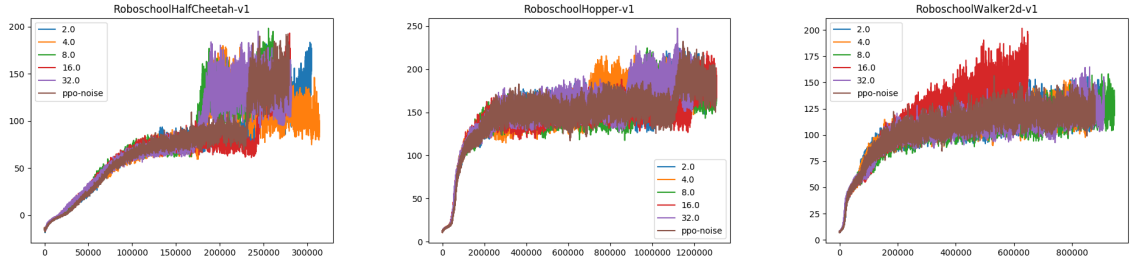
C.7.7 for $\beta = 0.05$ and $stddev = 0.1$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 22: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and, **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.1$ and vary α. But we also add gaussian noise to the observation, with stadard deviation *stddev*.

C.7.8 for $\beta = 0.05$ and $stddev = 1.0$



RoboschoolHalfCheetah RoboschoolHopper RoboschoolWalker

Figure 23: We run **ppo-original** and **ppo-len-γ-dep** for the same number of interaction when the threshold on the maximum length is 128 and call them **ppo-original-equalizer** and **ppo-original-γ-dep-equalizer**. We keep $\beta = 0.1$ and vary α. But we also add gaussian noise to the observation, with stadard deviation *stddev*.

D Study of RoboSchool environments underlying parameters

In the subsection C.3.2 we observe that while the models learns a policy, the episode length increases rapidly and surprisingly saturates, instead of increasing further. This slightly hampers the effect of analysis in length dependent trust region induced by $\mathcal{D}_\gamma$.

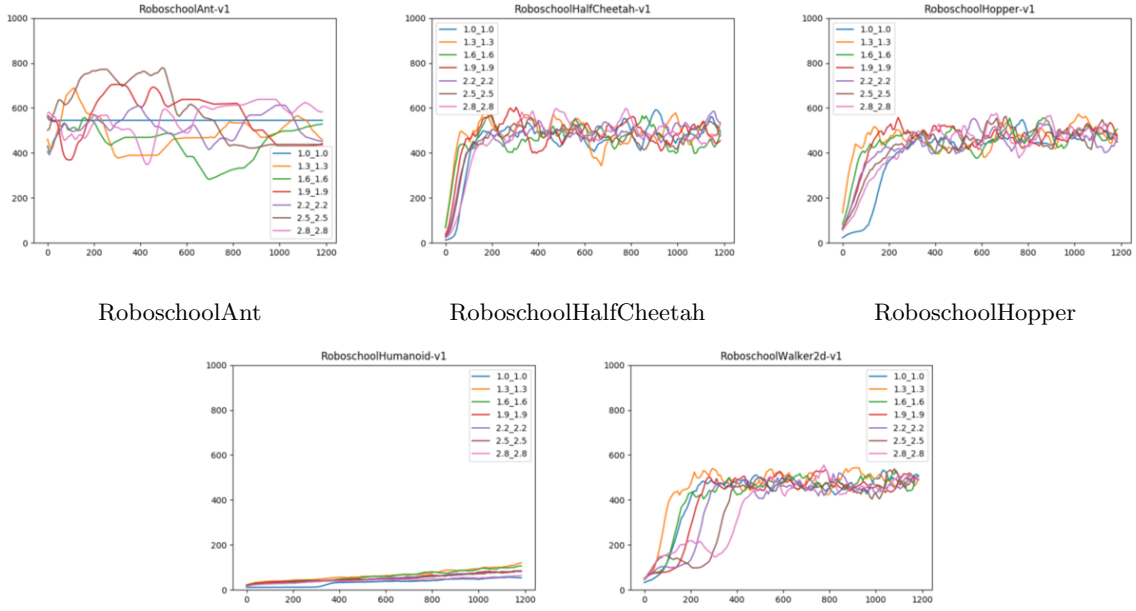
In the roboschool module, one can find $https://github.com/openai/roboschool/blob/master/roboschool/gym\_forward\_walk$ that the reward has 5 components:

1. alive
2. progress,
3. electricity-cost,
4. joints-at-limit-cost,
5. feet-collision-cost

The alive bonus has not been appropriately modulated to do not encourage the agent to die even though the agent can stay alive and collect more rewards. We tried to alter the *alive bonus* to make the staying alive a

significant component for the agnet. We multiply the alive bonus by a coefficient and study the agent behaviour. The alive bonus may be positive or negative, depending on specific parameters of the environment(for example, in the **humanoid** environment, if the center of mass of the robot falls below a certain limit, then it starts receiving negative alive bonuses, while alive bonuses are positive). The alive bonus for the environments is generally one unit. We multiplied the alive bonus with varying factors to observe the behavior of agent and see whether it learn to do not intentionally terminate the round (informally do not intentionally kill itself) as the model trains.

In the following plots, the x-axis represents the number of episodes seen by the model, and the y-axis represents the episode length. The numbers in the legend represent the factor by which the positive and negative alive bonus are scaled respectively. For example, the index 2.2_2.2 means a factor of 2.2 (the first number) multiplies the positive alive bonus, and a factor of 2.2 (the second number) multiplies negative alive bonus. We empirically study the effect of this bonuses, but as one can see from the plots, the changes in alive bonus did not make significant change in the episode length. Fig. 24



RoboschoolAnt RoboschoolHalfCheetah RoboschoolHopper

RoboschoolHumanoid RoboschoolWalker

Figure 24: PPO behavior when we change the alive bonus. It seems that the agent does not learn to do not intentionally terminate the episodes. We believe the reward shaping deployed in PPO requires substantial study and critical modification to make them suitable for further studies.

E GTRPO

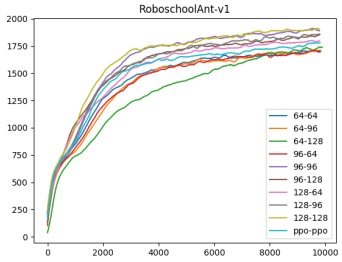
In this section we study the GTRPO behaviour on Robo-School environment . As it is mentioned in the Eq. 8 we need to train a network to estimate V and Q functions. We restate their definitions in the following;

$$
\begin{aligned}
&V_\pi(y_h, h, y_{h-1}, a_{h-1}) := \\
&\quad \mathbb{E}_\pi \left[\sum_h^H \gamma^h r_h | y_h = y, y_{h-1} = y_{h-1}, a^{h-1} = a_{h-1} \right] \\
&Q_\pi(y_{h+1}, a, y_h, h) := \\
&\quad \mathbb{E}_\pi \left[\sum_h^H \gamma^h r_h | y_h = y, y_{h+1} = y_{h+1}, a^h = a \right]
\end{aligned}
$$

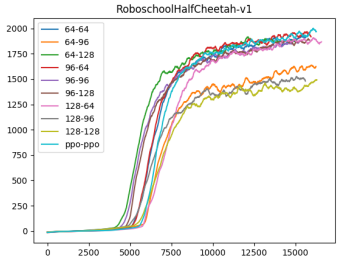
In practice we drop the h dependence. We train the V function using a simple neural network while we use samples of $R(\tau)$ as the data. It is worth noting that we do not use Bellman residual methods to learn the V since there is not Bellman imposed structure when memoryless policies are acquired. For each tuple of $y \rightarrow, a, \rightarrow y'$, where arrows represent the ordering of the events, we train the on-policy $V(y', y, a)$ to match the cumulative reward happens after observing y'. For the Q we deploy directly the sampled returns.

E.1 Network design choice for V

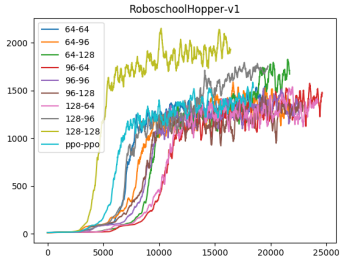
We deploy a simple neural network with 2 fully connected layers to train for V. We try to tune for the number of nodes in each layers Fig. 25. In the legend of the plots, each tuple represents the number of nodes in first and second layer respectively.



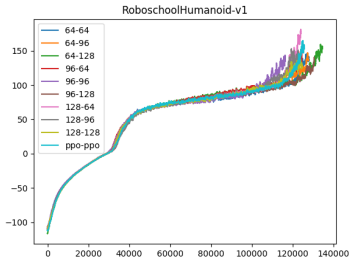
RoboschoolAnt



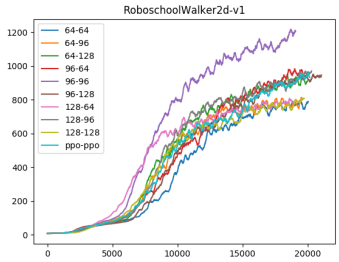
RoboschoolHalfCheetah



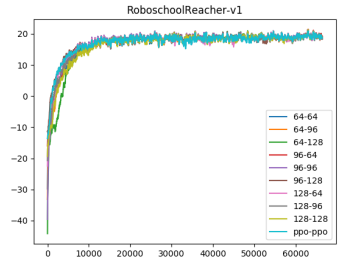
RoboschoolHopper



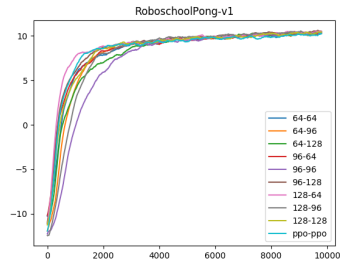
RoboschoolHumanoid



RoboschoolWalker



RoboschoolReacher

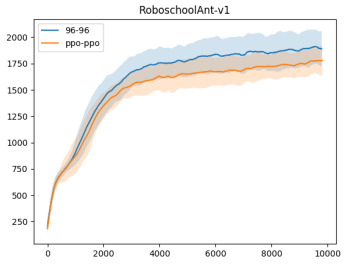


RoboschoolPong

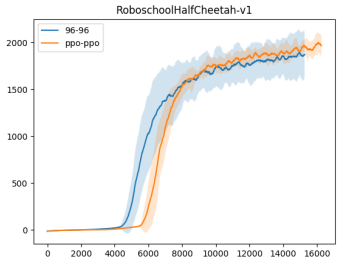
Figure 25: Performance of GTRPO when we use different design choice to train the V.

E.2 Plot with variance

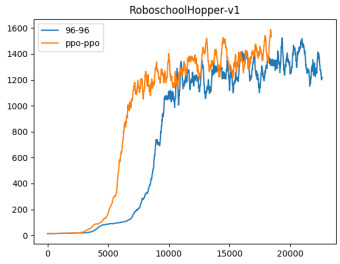
We observed that the neural net with nodes 96-96 performed consistently in comparison to its contemporaries Fig. 26. Following are the comparison plots along with variance.



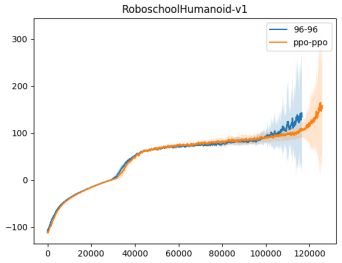
RoboschoolAnt



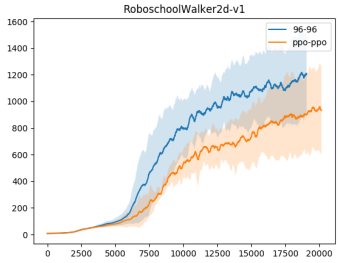
RoboschoolHalfCheetah



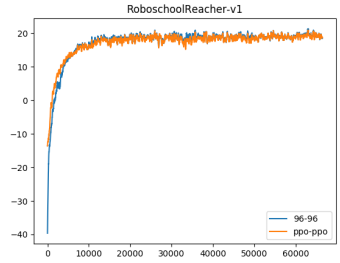
RoboschoolHopper



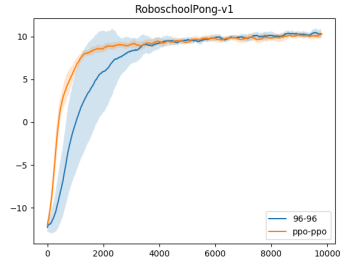
RoboschoolHumanoid



RoboschoolWalker



RoboschoolReacher



RoboschoolPong

Figure 26: Performance of GTRPO when V is approximated with two layer neural network of size 96-96.

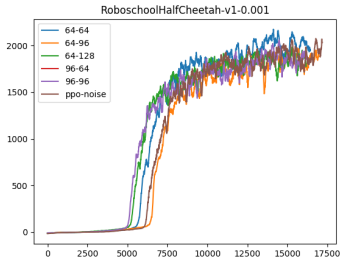
E.3 Noise

For further study, we also introduced Gaussian noise into the observation of RoboSchool environments and reduce the 'observability' of the states. We report the perfomrance for variaty of noise levels.

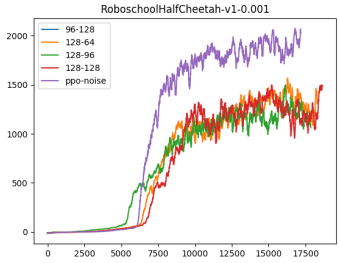
E.3.1 stddev = 0.001

RoboschoolAnt-1

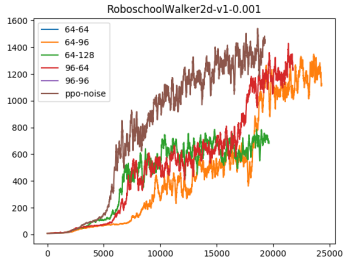
RoboschoolAnt-2



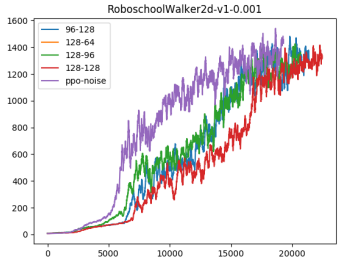
RoboschoolHalfCheetah-1



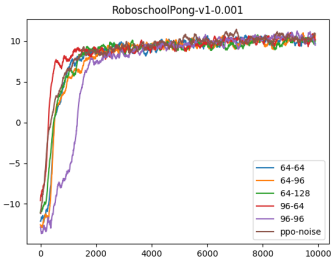
RoboschoolHalfCheetah-2



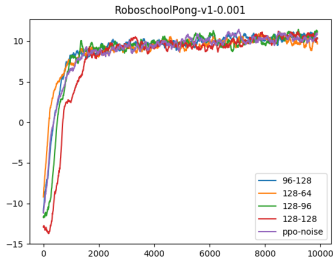
RoboschoolWalker



RoboschoolWalker-2



RoboschoolPong-1



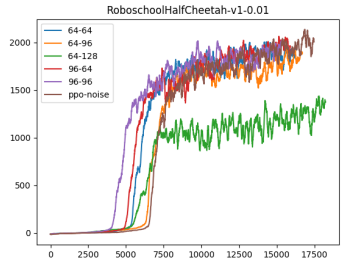
RoboschoolPong-2

Figure 27: Behavior of GTRPO under noised observation with standard deviation of 0.001

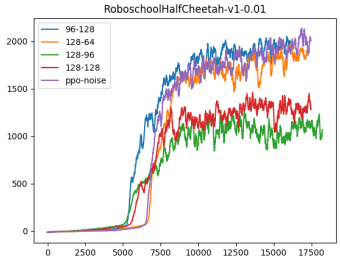
E.3.2 stddev = 0.01

RoboschoolAnt-1

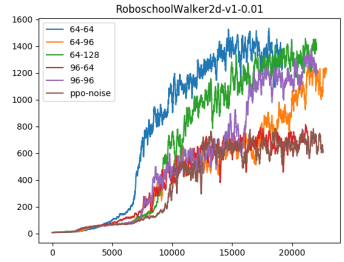
RoboschoolAnt-2



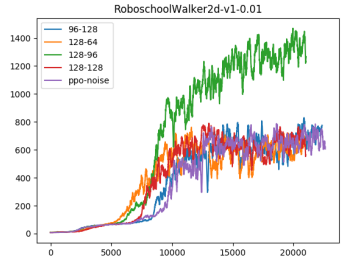
RoboschoolHalfCheetah-1



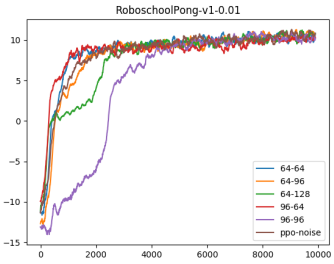
RoboschoolHalfCheetah-2



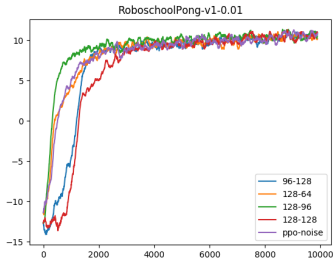
RoboschoolWalker-1



RoboschoolWalker-2



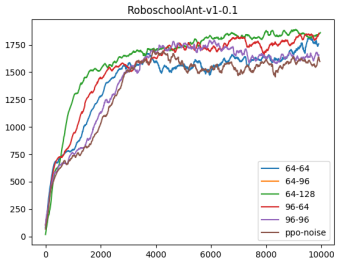
RoboschoolPong-1



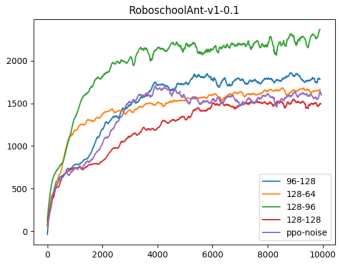
RoboschoolPong-2

Figure 28: Behavior of GTRPO under noised observation with standard deviation of 0.01

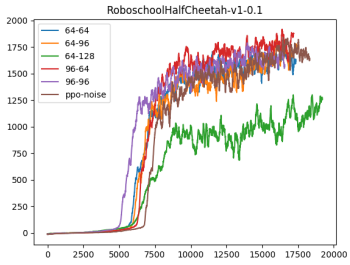
E.3.3 stddev = 0.1



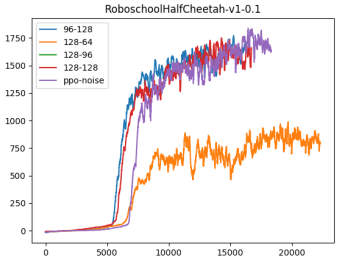
RoboschoolAnt-1



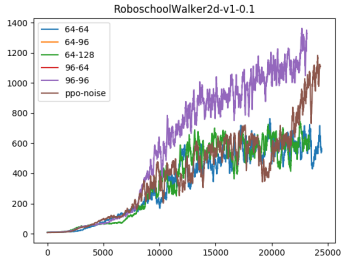
RoboschoolAnt-2



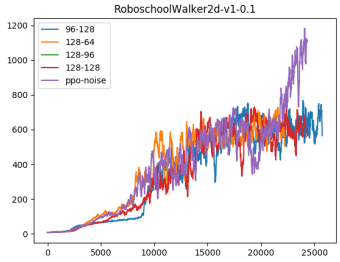
RoboschoolHalfCheetah-1



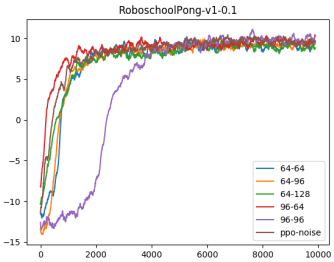
RoboschoolHalfCheetah-2



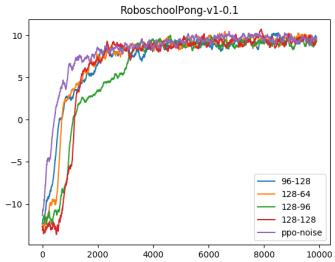
RoboschoolWalker-1



RoboschoolWalker-2



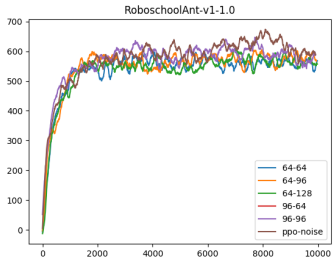
RoboschoolPong-1



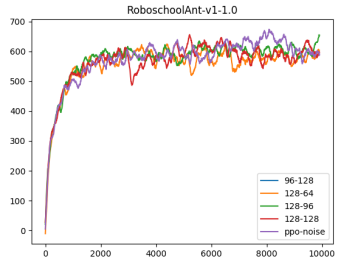
RoboschoolPong-2

Figure 29: Behavior of GTRPO under noised observation with standard deviation of 0.1

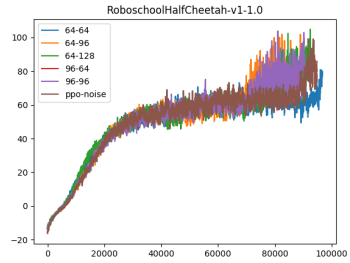
E.4 stddev = 1.0



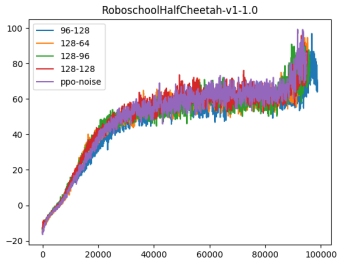
RoboschoolAnt-1



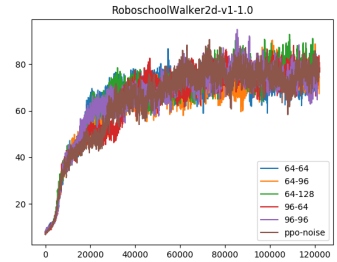
RoboschoolAnt-2



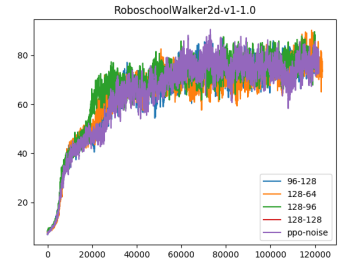
RoboschoolHalfCheetah-1



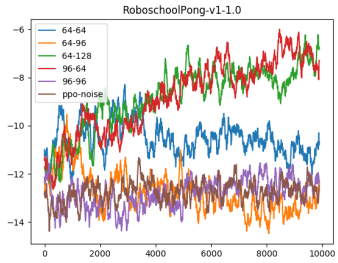
RoboschoolHalfCheetah-1



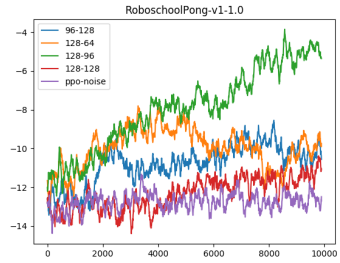
RoboschoolWalker-1



RoboschoolWalker-2



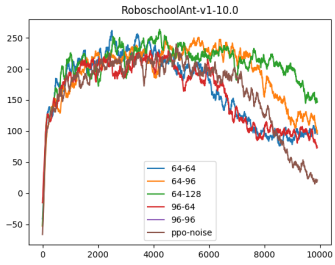
RoboschoolPong-1



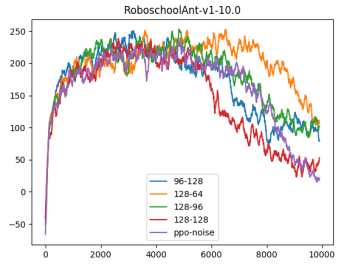
RoboschoolPong-2

Figure 30: Behavior of GTRPO under noised observation with standard deviation of 1.0

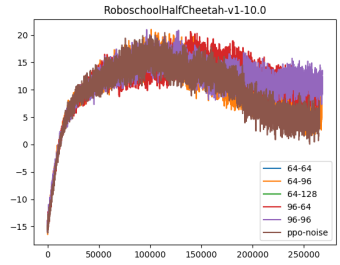
E.4.1 stddev = 10.0



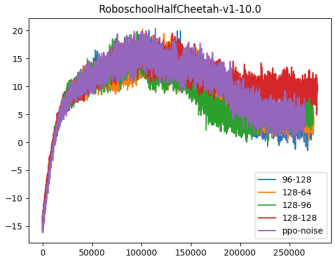
RoboschoolAnt-1



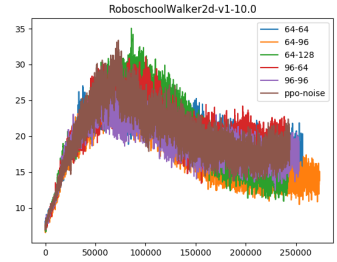
RoboschoolAnt-2



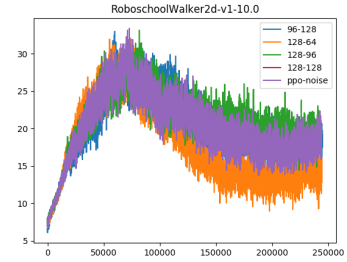
RoboschoolHalfCheetah-1



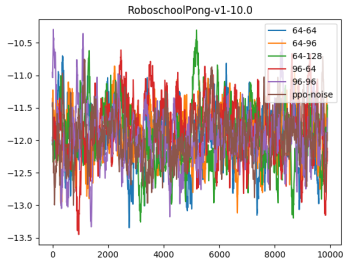
RoboschoolHalfCheetah-2



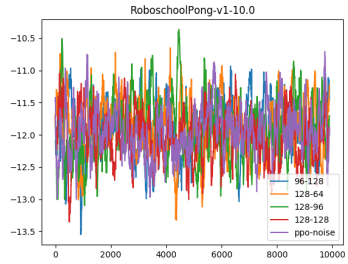
RoboschoolWalker-1



RoboschoolWalker-2



RoboschoolPong-1



RoboschoolPong-2

Figure 31: Behavior of GTRPO under noised observation with standard deviation of 10.0

F PPO through signSGD

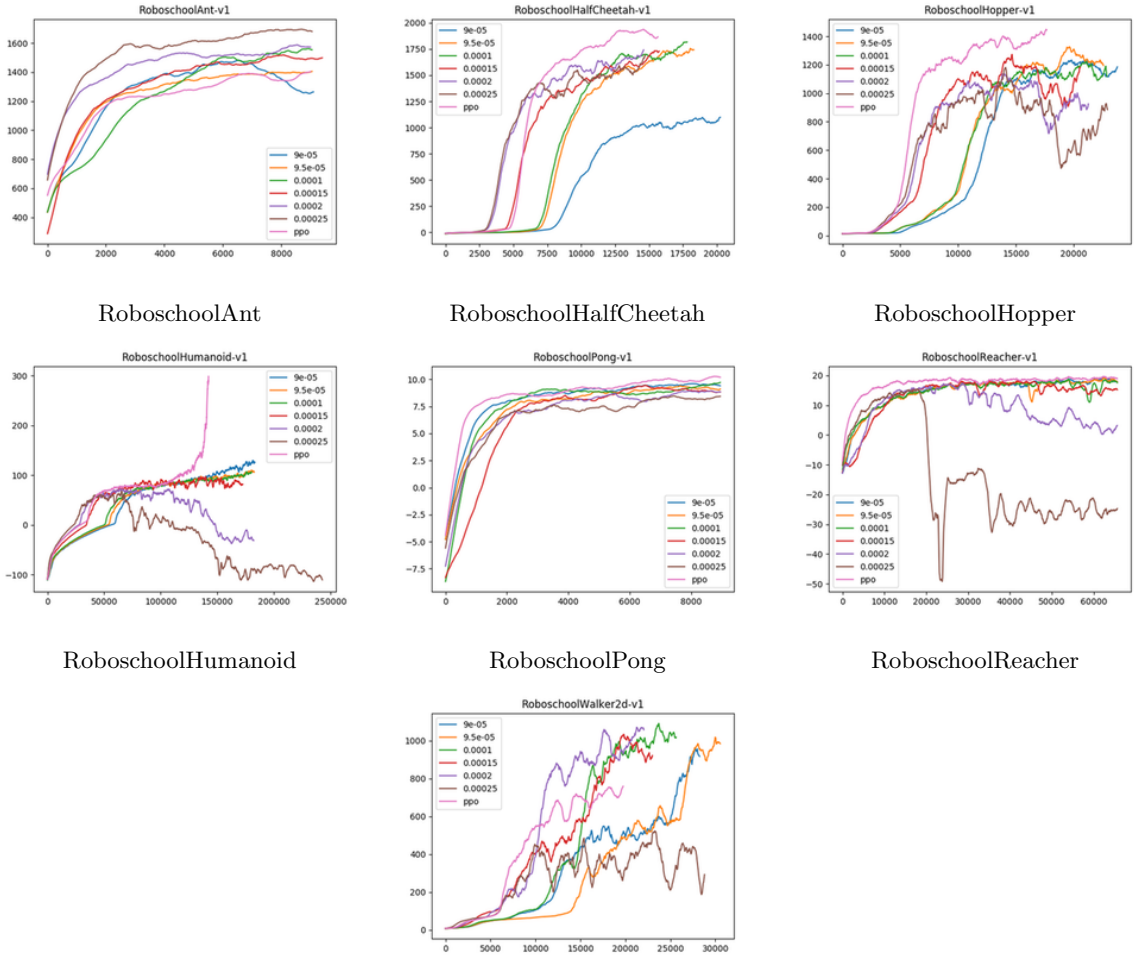
We made a further empirically study of a different way of imposing the trust region. signSGD, as a optimization method computes the gradient vector but move in the direction of sign of gradient. This approach implicitly prevents big changes in the parameter space and moves the parameters in all the directions with the same magnitude. It also forces to move with the same magnitude toward the directions that the trust region suggests low changes in them.

signSGD:

- $g_k \leftarrow$ stochastic gradient
- $x_{k+1} \leftarrow x_k - \delta \ \text{sign}(g_k)$

We apply PPO on the Robo-school environment and deploy the signSGD optimizer for the policy gradient.

The legend in the following plots denote the *learning rate* that we try for signSGD.



RoboschoolAnt RoboschoolHalfCheetah RoboschoolHopper

RoboschoolHumanoid RoboschoolPong RoboschoolReacher

RoboschoolWalker

Figure 32: Behavior of PPO when signSGD is deployed as the optimizer