Revisiting Design Choices in Proximal Policy Optimization

Chloe Ching-Yun Hsu Celestine Mendler-Dünner Moritz Hardt
{chloehsu,mendler,hardt}@ berkeley.edu
University of California, Berkeley

Abstract

Proximal Policy Optimization (PPO) is a popular deep policy gradient algorithm. In standard implementations, PPO regularizes policy updates with clipped probability ratios, and parameterizes policies with either continuous Gaussian distributions or discrete Softmax distributions. These design choices are widely accepted, and motivated by empirical performance comparisons on MuJoCo and Atari benchmarks.

We revisit these practices outside the regime of current benchmarks, and expose three failure modes of standard PPO. We explain why standard design choices are problematic in these cases, and show that alternative choices of surrogate objectives and policy parameterizations can prevent the failure modes. We hope that our work serves as a reminder that many algorithmic design choices in reinforcement learning are tied to specific simulation environments. We should not implicitly accept these choices as a standard part of a more general algorithm.

1 Introduction

The PPO algorithm [22] is a policy gradient method that is used in diverse high profile reinforcement learning (RL) applications to train policies, including playing DOTA [7], manipulating a Rubik's cube [2], and designing chip placement [15].

The key feature of the PPO algorithm is a *surrogate objective* for computing policy updates. The surrogate objective regularizes large policy updates, in the spirit of a trust region method, so that each policy update step stays within a close neighborhood around the previous-iteration policy. This increases the validity of the surrogate objective which is based on data collected from the previous-iteration policy. The algorithm then incrementally refines the policy using multiple steps of stochastic gradient ascent before collecting new data.

In its full generality PPO refers to a family of algorithms, where the exact choice of the surrogate objective is left as a flexible design choice to the user. As a consequence, several versions of this algorithm have been proposed in the literature. All of them build on the idea of regularizing the distance between the initial and the updated policy, but they differ in their implementation.

A natural approach to incorporate regularization is to use the Kullback-Leibler (KL) divergence between successive policy iterations as a penalty in the surrogate objective. Such an approach can be theoretically motivated by approximating the Trust Region Policy Optimization [21] (TRPO) method that is known to monotonically improve policies.

Rather surprisingly, a different, more ad-hoc surrogate objective has emerged as the most common design choice among practitioners. This alternative surrogate objective is the so-called *clipped objective*. Instead of regularizing the update with a penalty, it greedily ignores any change to the parameter update after the probability ratio between the initial and the updated policy exceeds a predefined threshold. This heuristic is easy to implement and has demonstrated good empirical performance on a number of MuJoCo benchmarks [22]. Today, most implementations follow this design choice and it is considered a standard part of PPO – to the extent that it is often not even mentioned in experimental setups.

The choice of surrogate model is only one example for a standard design choice used to learn policies with PPO. Another example is the distribution family used for policy parameterization. Standard PPO implementations use diagonal Gaussian distributions on continuous action spaces, and categorical Softmax distributions on discrete action spaces. Alternative families of distributions have been proposed in the literature in the context of other RL algorithms, and there is no reason to rule them out a priori for PPO.

In this paper we revisit these standard design choices for the PPO algorithm. Our main concern is that they have been made within the limited regime of current benchmarks, in particular, MuJoCo and Atari. It is not a priori clear how robust these methods are when used in different environments. Our goal is to contribute to a more principled understanding of PPO. We point out potential issues with current design choices, investigate alternative design choices, and propose avenues for improving the PPO method.

1.1 Our contributions

We design simple test cases to isolate three failure modes where standard PPO (using the clipped objective and Gaussian/Softmax policy parameterization) does not achieve the desired convergence behavior: On continuous action spaces, standard PPO is unstable when rewards vanish outside bounded support, and it is sensitive to initialization when there are locally optimal actions close to initialization. On discrete action spaces with sparse high rewards, standard PPO often gets stuck at suboptimal actions. We analyze the reason for these failure modes and explain why they are not exposed by standard benchmarks.

We then revisit alternative design choices and evaluate their performance in these situations. On discrete action spaces, we revisit the more principled KL-regularized surrogate objectives, and show that they make PPO more robust to our failure mode example. In fact, KL-regularized PPO even comes with convergence guarantees for one of the settings, and concerningly this favorable property is not preserved by the clipping heuristic.

On continuous action spaces, we find that policy parameterization with Beta distributions is a favorable combination with PPO, because it is more robust to outliers and can be initialized uniformly. Beta policies avoid the identified failure modes, while also significantly improving PPO performance on MuJoCo environments. For example, PPO with beta policy achieves 2x cumulative rewards on the OpenAI Gym [8] Humanoid-v2 task.

In summary, our study suggests that Beta policy parameterization and KL-regularized objectives should be reconsidered for PPO, especially when moving outside the regime of current benchmarks. While we do not claim that our proposed alternatives improve PPO in all settings, we show that the community might have ruled out these variants too soon based on early MuJoCo experiments. We hope this paper serves as a reminder that many algorithmic design choices in RL are decided based on specific simulation environments. Without a deeper understanding, we need to reassess these conclusions in new environments.

2 Background and related work

We start by providing some background on the PPO algorithm and then introduce the different design choices that we investigate and compare in this manuscript.

2.1 Proximal Policy Optimization

PPO is a policy gradient algorithm that learns a parameterized policy π_θ. It iteratively updates the parameters θ of the policy by solving a local optimization problem:

$$\theta_{\text{new}} \leftarrow \arg\max_{\theta} \mathcal{L}(\theta; \hat{A}^{\pi_{\text{old}}}). \quad (1)$$

$$
\begin{aligned}
\mathcal{L}^{\text{CLIP}}(\theta) &:= \mathop{\mathbb{E}}_{a,s\sim\pi_{\text{old}}}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}\hat{A}^{\pi_{\text{old}}}(a,s),\ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)},1-\epsilon,1+\epsilon\right)\hat{A}^{\pi_{\text{old}}}(a,s)\right)\right] && \text{(2a)}\\
\mathcal{L}^{\text{KL,forward}}(\theta) &:= \mathop{\mathbb{E}}_{a,s\sim\pi_{\text{old}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}\hat{A}^{\pi_{\text{old}}}(a,s)\right] - \beta D_{\text{KL}}(\pi_{\text{old}}\|\pi_\theta) && \text{(2b)}\\
\mathcal{L}^{\text{KL,reverse}}(\theta) &:= \mathop{\mathbb{E}}_{a,s\sim\pi_{\text{old}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}\hat{A}^{\pi_{\text{old}}}(a,s)\right] - \beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{old}}) && \text{(2c)}
\end{aligned}
$$

Figure 1: Design choices for PPO surrogate objective.

$$
\begin{aligned}
&\text{Gaussian} && \pi_\theta(a|s) := \mathcal{N}\left(\mu_\theta(s), \sigma_\theta^2(s)\right) && && \text{(3a)}\\
&\text{Beta} && \pi_\theta(a|s) := f\left(\frac{a-l}{r-l}, \alpha_\theta(s), \beta_\theta(s)\right) && \text{with} \quad f(x,\alpha,\beta) := \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x^{\beta-1}) && \text{(3b)}\\
&\text{Softmax} && \pi_\theta(a|s) := \frac{1}{c_s}e^{\phi_\theta(s,a)} && \text{with} \quad c_s = \sum_{a'\in\mathcal{A}} e^{\phi_\theta(s,a')} && \text{(3c)}
\end{aligned}
$$

Figure 2: Design choices for PPO policy parameterization.

The surrogate objective $\mathcal{L}$ uses the advantage estimates $\hat{A}^{\pi_{\text{old}}}(s,a)\ \forall s,a$ to assess how much better a particular action a is on state s compared to a randomly sampled action from the previous-iteration policy $\pi_{\text{old}}(\cdot|s)$. These estimates are obtained by sampling trajectories from π_{old} prior to each optimization step. Based on these estimates, the surrogate objective $\mathcal{L}$ is then optimized to find a new parameter vector θ_{new} for the policy π_θ. The optimization (1) is not performed exactly in PPO, but using multiple epochs of stochastic gradient ascent.

2.1.1 Policy regularization

The *surrogate objective* $\mathcal{L}$ in (1) is the key feature of PPO, as it regularizes excessively large policy updates and allows the algorithm to efficiently reuse available data. There are different variants to implement this regularization, we focus on the three primary variants summarized in Figure 1.

The first objective $\mathcal{L}^{\text{CLIP}}$ in (2a) corresponds to the clipping heuristic proposed in [22]. It is simple to implement and uses a clipping threshold $\epsilon > 0$ to control the size of each policy update. This objective is most popular amongst practitioners because of its good benchmark performance.

The second objective $\mathcal{L}^{\text{KL,forward}}$ in (2b) uses a soft constraint on the forward KL distance between the initial and the updated policy. The regularization strength is controlled by the regularization parameter β. This objective was also proposed in [22] and is closely related to TRPO [21], a well understood trust region algorithm related to PPO.

The third objective $\mathcal{L}^{\text{KL,reverse}}$ in (2c) uses the reversed KL-distance for regularization. This objective has not been studied empirically but has been the main focus of theoretical studies around PPO [11, 14, 16]. The reason is its close connection to mirror descent, a well understood optimization method with provable convergence guarantees.

To make the connection between KL-regularized PPO and mirror descent formal, Liu et al. [14] use the representation power of overparameterized neural networks to approximate the infinite-dimensional mirror descent updates, while other works focus on MDPs with finite state and action spaces. We review and extend these theoretical studies connecting PPO to mirror descent in Appendix A. We simplify their exposition and clarify that the assumptions in the convergence analyses of [14, 16] are satisfied only for certain families of policy parameterizations. These refined theoretical insights help us draw new intuition on the convergence properties of the PPO variants, and how the choice of policy parameterization affects PPO behavior.

2.1.2 Policy parameterization

Standard PPO implementations use diagonal Gaussian distributions for parameterizing the policy π_θ on continuous action spaces, and Softmax distributions on discrete action spaces. In this paper we consider Beta distributions as an alternative parameterization for continuous action spaces. The respective policies $\pi_\theta(a|s)$ are stated in Figure 2.

Beta policy parameterizations have previously been proposed in [9] for the TRPO algorithm. The authors chose Beta policy parameterizations because they can explicitly incorporate action space boundaries $a \in [l, r]$ and eliminate the biased boundary effects caused by truncated Gaussians. In this work we demonstrate that this is not the only benefit of using Beta parameterization, it also leads to more reliable convergence behavior of the PPO algorithm in our test cases and can outperform standard PPO even in settings where boundary effects are not relevant.

2.2 Empirical studies

Previous works have already expressed concerns about the robustness of standard PPO. Henderson et al. [12] highlight the concern that RL algorithm comparisons depend on the environments, and show that PPO and other deep RL algorithms are sensitive to random initialization and reward scaling. Our work complements previous work with more in-depth analysis of standard PPO's lack of robustness in relation to the two design choices: policy parameterization and surrogate objective.

Recent ablations studies [3, 10] investigate how some hyperparameters and design choices affect the performance of PPO and other on-policy RL algorithms on MuJoCo benchmarks. While these studies provide empirical results on how to optimize PPO performance on MuJoCo benchmarks, our work takes a first step in examining failure modes of standard PPO outside of current benchmarks. Our work also differs from existing empirical work on the studied design choices. Motivated by the discovered failure modes of standard PPO, we choose to focus on policy parameterizations and surrogate objectives. While the alternative design choices we study have been previously proposed, their impact on PPO performance has been largely unknown and unquestioned. For example, in a recent large-scale study [3] of more than 50 design choices in on-policy RL algorithms, the choices of clipping as regularization and Gaussian policy are not included in the studied design choices.

3 Failure modes of standard PPO

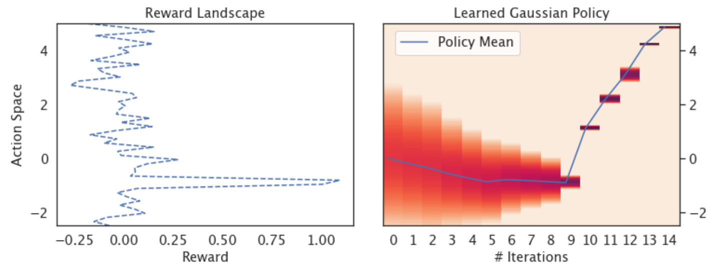
We start by outlining three failure modes of the common combination of clipped surrogate objective with continuous Gaussian and discrete Softmax policy parameterization. We refer to these design choices as *standard PPO*.

3.1 Reward signal with bounded support

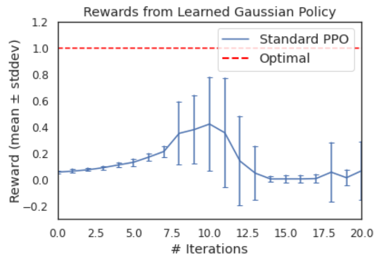
The first failure mode illustrates a scenario where the clipped objective fails to recover from a bad Gaussian policy update step. The clipping mechanism effectively prevents the policy from moving further away once it is outside the trust region, but it does not bound the size of an individual policy update step. This behavior is particularly problematic if a single reward signal can cause the policy to end up in regions with low reward signal. To illustrate this issue, consider the following toy example where the support of the reward signal is bounded:

Failure mode 1. *Consider the simple 1-dimensional example on a continuous action space illustrated in Figure 3a. The reward landscape, visualized in the left plot, has a single peak of high reward around $a = -0.9$ and is purely random noise outside the interval $[-1.0, -0.8]$. For this simple example we illustrate the training of a Gaussian policy with the clipped objective in the right plot. We see that in the initial phase of training the policy π_θ quickly moves towards the high-reward peak and concentrates by rapidly decreasing the standard deviation. But then, at iteration 10, it starts to diverge and drifts away into low reward regions. The experienced reward across iterations is shown in Figure 3b.*

The undesirable behavior of Failure mode 1 can reliably be reproduced in similar 1-dimensional settings, where reward signals are bounded to a subregion of the action space. The abruptly



(a) Reward landscape over action space (left) and heatmap of learned Gaussian policy density across iterations for one training run (right).



(b) Reward of learned Gaussian policies, with mean and std over 20 runs.

Figure 3: *Failure mode 1.* Policy training on the 1-d example outlined in Section 3.1 using standard PPO with Gaussian policy parameterization. Initially the policy moves towards the high reward peak, but after 10 iterations the policy drifts away and the reward drops.

vanishing reward signal outside the interval $[-1.0, -0.8]$ is the main culprit. As we will explain it is particularly problematic for the combination of the clipped objective with the Gaussian policy. Later, in Section 4, we demonstrate that PPO with the KL-regularized surrogate objective or with alternative policy parameterization can better handle this failure mode example.

Why is the combination of Gaussian policy parameterization and clipping problematic for this example? To better understand why this undesirable behavior occurs, let us inspect the gradient of the clipped objective (2a). To simplify notation we denote the probability ratio by $r(a|s) := \pi_\theta(a|s)/\pi_{\text{old}}(a|s)$ and the advantage estimates of action a at state s as $\hat{A}_{a,s} := \hat{A}^{\pi_{\text{old}}}(a,s)$.

$$\nabla_\theta \mathcal{L}^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{a,s\sim\pi_{\text{old}}} \left[1\left\{ |r(a|s) - 1| < \epsilon \text{ or } \operatorname{sgn}(r(a|s) - 1) \neq \operatorname{sgn}(\hat{A}_{a,s}) \right\} r(a|s) \nabla_\theta \log \pi_\theta(a|s) \hat{A}_{a,s} \right]. \tag{4}$$

The gradient update (4) is non-zero as long as the probability ration $r(a|s)$ deviates from 1 by less than ϵ. As soon as this threshold is exceeded, the clipping mechanism effectively prevents the policy from moving futher away in subsequent iterations. However, it does not regularize individual policy update steps to stay inside the trust region. Thus, if update steps are large, a single reward signal can cause the policy to move far away from π_{old}. In Failure mode 1 it is particularly problematic if the policy suddenly jump outside the support of the reward signal. Sampling from such a policy in the next round may contain little signal to recover.

Of course, the strength of the above phenomena critically depend on the choice of policy parameterization. In the following we will explain why a standard Gaussian policy parameterization $\theta = (\mu_\theta, \sigma_\theta^2)$ such as in (3a) can be problematic.

Therefore, recall the gradient update in (4). The advantage value $\hat{A}_{a,s}$ is weighted by the score function $\nabla_\theta \log \pi_\theta(a|s)$ which measures how quickly the probability density at a given action changes as the distribution parameters change. For Gaussian policies the score function is given by

$$\nabla_\theta \log \pi_\theta(a|s) = \frac{1}{\sigma_\theta^2(s)} \left[(a - \mu_\theta(s)) \quad \frac{(a - \mu_\theta(s))^2 - \sigma_\theta(s)^2}{\sigma_\theta(s)} \right].$$

The expression is large for actions further away from the Gaussian policy mean μ and grows inversely proportional to the variance of the Gaussian policy. The score function is visualized in Figure 10a in Section 4.1.2. As the policy variance decreases over the course of training, the policy becomes more and more sensitive to actions in the tails. Hence, in later iterations, a single reward signal relatively far from the policy mean can lead to a large policy update and cause the policy to suddenly jump to low reward regions. Without meaningful reward signals, the mean and the standard deviation of the Gaussian policy fluctuate randomly due to noise. Once this policy is used to sample new data, the small standard deviation makes it extremely unlikely for PPO to re-discover the high-reward interval because it under-explores the reward landscape.

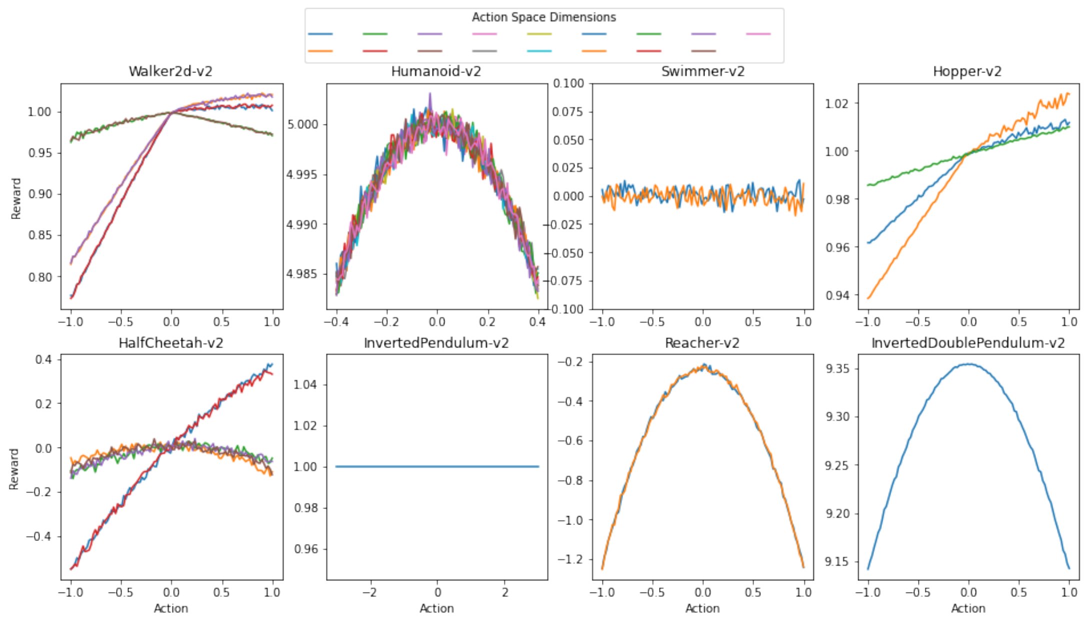


Figure 4: Reward landscapes on initial states of common OpenAI Gym MuJoCo benchmarks.

Why haven't we observed this failure mode on MuJoCo benchmarks? This failure mode is hidden for several reasons. First, OpenAI Gym provides designed action space boundaires for each task to facilitate learning and clipps sampled actions outside of the boundaries. Second, most reward landscapes are smooth (Figure 4), unlike our failure mode example. Third, in many PPO implementations, the variance in the Gaussian policy is a state-independent variable. While state-independent variance avoids the variance to shrink quickly, it also limits the expressive power of the policy, forcing the policy to sample suboptimal actions even on states with higher confidence.

Although failure mode 1 is only a toy example, reward signals with narrow bounded support are of practical relevance. Such situations can arise due to multiple reasons, such as unknown action boundaries and sparse rewards. If we do not know a priori where meaningful reward signals lie in the action space prior to policy training, we have to set wider action space boundaries that might include vanishing reward signals. Even with knowledge of action space boundaries, sparse rewards can also result in limited reward signals, such as in robotics, where the robot only receives a high reward for completing a task such as inserting a cable or stacking two blocks [20, 25].

3.2 High-dimensional discrete action spaces

In the second failure mode we illustrate an example where clipping converges to suboptimal actions for distrecte action spaces with the standard softmax policy parameterization. This failure mode is particularly worrying because it happens within the convergence regime of the more principled KL-regularized PPO which clipping aims to approximate. We consider the following setup:

Failure mode 2. *Let half of the actions have zero reward, half of the actions have suboptimal reward around 0.5, and there is only one optimal action with average reward 1. This example is illustrated in Figure 5. It shows how the clipped objective is likely to converge to suboptimal actions with discrete Softmax policy. For such an environment with 100 actions, standard PPO gets stuck at the suboptimal actions in around 50% of the runs.*

The problem is that when the clipped objective sees only the bad actions (reward 0) and the suboptimal actions (reward 0.5) without seeing the optimal action, it tends to increase the probability ratio of the suboptimal actions by $(1+\epsilon)$, as maximally permitted by the clipping mechanism. After

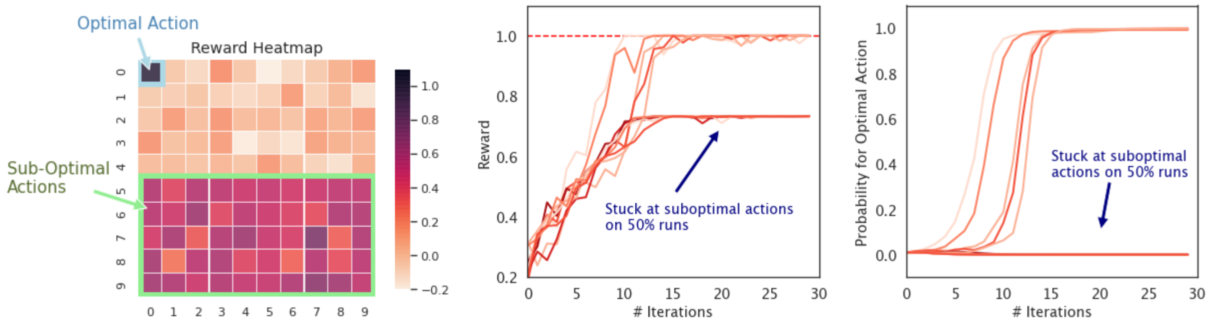


Figure 5: *Failure mode 2:* Half of the actions have zero reward, half have suboptimal rewards around 0.5, and only one optimal action has average reward 1. In this discretized environment with 100 actions, PPO with the clipped objective gets stuck at suboptimal actions around half of the times.

increasing the probability of suboptimal actions in several iterations, the policy is less and less likely to sample the optimal action, and further increases the suboptimal action probabilities.

When testing standard PPO in similar environments with varying number of actions (Figure 8) from 10 to 100, we see that this failure mode is increasingly problematic as the number of actions increases because the probability of seeing the optimal action in each iteration decreases.

We further note that alternative approaches such as reducing learning rate and increasing batch size can only partially mitigate the issue. For example, when reducing the learning rate by 100× from 0.1 to 0.001, PPO with clipping still fails on 15% runs when action dimension is 100, while PPO with reverse KL regularization succeeds on all 20 runs. Moreover, reducing the learning rate or increasing the batch size is not an ideal fix as it slows down training and requires more samples. Even though large batch sizes (larger than size of state-action space) can naively fix this failure mode on the simple single-state bandit example, it does not scale to more complex environments. When the state space $\mathcal{S}$ is large, sampling the full action space $\mathcal{A}$ on all states would require a batch size of $|\mathcal{S}| \times |\mathcal{A}|$. Even with relatively large batch sizes in more complex environments, we can not realistically hope to see all actions on each state.

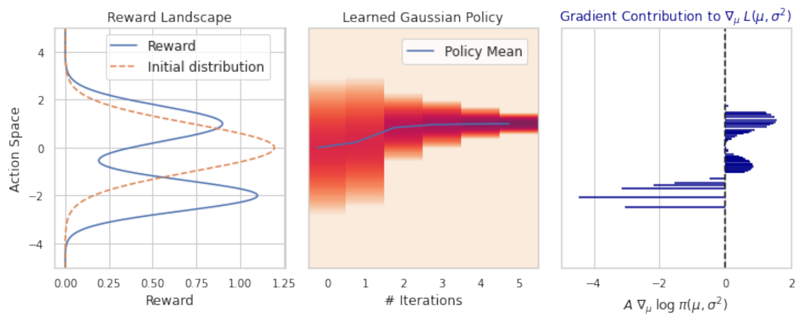
Instead of masking the underlying issue with partial fixes, understanding and analyzing failure modes in a principled way is a necessary first step towards designing robust fixes. In Section 4 we show that using KL regularization as an alternative surrogate model provides a scalable and theoretically grounded fix for this failure mode. It is interesting to see that some alternative algorithm design choices can be provably more robust, and we hope to encourage the community to keep questioning established algorithms, in particular for new environments.

While the high-dimensionality of action spaces is one aspect of the classical exploration-exploitation tradeoff, existing RL research around exploration mostly focuses on continuous, rather than discrete action space [17, 18, 19, 24]. Atari games in OpenAI Gym have 3-18 discrete actions (see Table 4 in the appendix for the exact statistics). They are unable to reveil this failure mode and assess PPOs performance on higher-dimensional discrete action spaces.

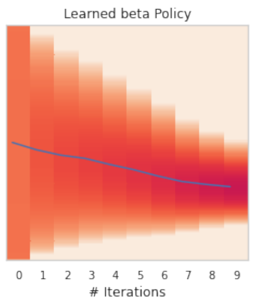
3.3 Locally optimal actions close to initialization

The third example shows that standard PPO can converge to suboptimal actions, even close to initialization. While convergence to suboptimal solutions is a general problem in RL, this example illustrates that for standard Gaussian policy parameterization this is doubly problematic as its suboptimal convergence is additionally aggravated by the initialization.

Failure mode 3. *Consider a simple 1-dimensional example with two reward peaks, as illustrated in Figure 6a; one suboptimal peak centered at $+1$ closer to the initial distribution $\mathcal{N}(0,1)$, and one optimal peak centered at -2 further away from the initial distribution. In this example, we see that the policy converges to the suboptimal action around $+1$, as visualized by the heatmap in Figure 6a.*



(a) Gaussian policy



(b) Beta policy

Figure 6: *Failure mode 3.* A double peak smooth reward landscape. (a) Suboptimal actions close to initialization are problematic for PPO with clipped objectives and Gaussian policies, as the suboptimal peak occupies more mass in the initial distribution and weighs aggregated gradients towards the suboptimal peak. (b) Beta policy with uniform initialization learns the optimal reward peak.

The suboptimal convergence in this example is due to the shape of the initial Gaussian distribution, which is typically set to a diagonal Gaussian policy with zero mean and unit variance in all dimensions. In the initial Gaussian distribution, even though the gradients are of smaller magnitude around the suboptimal peak, they occupy more probability mass and hence cause the aggregate gradient to move towards the suboptimal peak close to 0.

This failure mode might be alleviated by a careful choice of initialization. This, however, would require a priori knowledge of the reward landscape which is often not reasonable in practice, and if available there would be more effective ways to exploit it.

In general, this particular issue is not commonly discussed in the literature because it does not arise in the popular OpenAI Gym [8] implementation of MuJoCo tasks. There, each environment comes with pre-defined action space boundaries and within the action space boundaries most tasks have single-peak reward landscapes, see Figure 4.

In Section 4 we demonstrate that Beta policy parameterization can help avoid this undesirable convergence of PPO to suboptimal actions close to initialization, since it can be initialized uniformly in the action space. Of course, there are cases where both Gaussian and Beta policies would converge sub-optimally, not caused by initialization. For example, when there is a very wide action region with relatively high reward and a very narrow action region with the highest reward, both Gaussian and beta policies tend to converge to the suboptimal wide region.

4 Improvements for PPO failure modes

We demonstrate that the KL-regularized surrogate objective and Beta policy parameterizations provide two simple alternatives to avoid the three failure modes of standard PPO discussed in Section 3. While both alternative design choices have been previously proposed in the RL literature, they have been largely ignored in combination with the PPO algorithm.

4.1 KL-regularized surrogate objectives

We revisit the KL-regularized PPO surrogate objectives (2b) and (2c) as an alternative to the clipped surrogate objective (2a), motivated by its close connection to the mirror descent algorithm and the resulting theoretical guarantees. We show that they make PPO more robust to Failure mode 1 and 2, while having comparable MuJoCo performance.

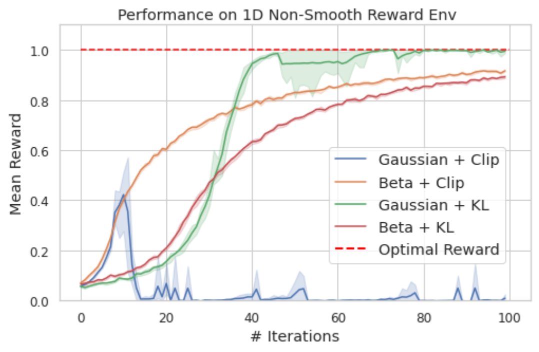


Figure 7: Beta policy and KL regularization both independently avoid Failure mode 1 (Figure 3, continuous bandit with bounded reward signals).

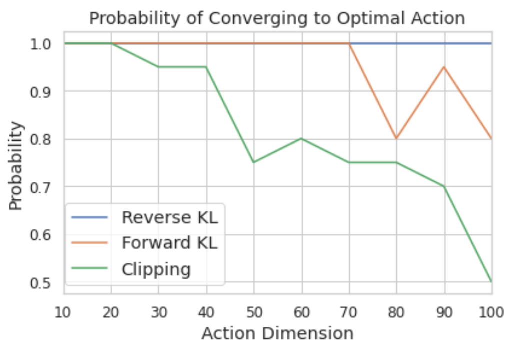


Figure 8: KL-regularized objectives are more robust in Failure mode 2 (Figure 5, high-dimensional discrete action space).

4.1.1 Clarification of KL direction

The KL divergence is asymmetric and it remains unclear from the literature in which direction the KL divergence should be computed for regularizing the PPO surrogate objective. There are two possible versions of the KL-regularized objective, *forward-KL* (2b) and *reverse-KL* (2c). We are not aware of any empirical comparison of their performance. The forward direction is evaluated in the original PPO paper [22] but the reverse direction has solely been studied theoretically.

The KL-divergence is approximately symmetric throughout the algorithm and the KL direction does not significantly affect PPO's performance, due to the regularizing properties of the surrogate objective used in PPO. To verify, we use the Taylor expansion $r(a|s)\log(r(a|s)) \approx r(a|s) - 1 + \frac{1}{2}(r(a|s) - 1)^2$, and the identity $\int_x \pi_\theta(x)\nabla_\theta \log(\pi_\theta(x)) = 0$, to quantify the difference between the two KL terms:

$$\nabla_\theta D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\mathrm{old}}) - \nabla_\theta D_{\mathrm{KL}}(\pi_{\mathrm{old}} \parallel \pi_\theta) \approx \frac{1}{2} \mathop{\mathbb{E}}_{a,s\sim\pi_{\mathrm{old}}} \left[|r(a|s) - 1|^2 \nabla_\theta(\log \pi_\theta(a|s)) \right].$$

The difference between the two KL-regularized objectives is small in practice, since PPO regularizes the probability ratio $r(a|s)$ to be close to 1, Indeed, the correlation between $D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\mathrm{old}})$ and $D_{\mathrm{KL}}(\pi_{\mathrm{old}} \parallel \pi_\theta)$ is close to 1 throughout the training of common benchmark tasks, as illustrated in Figure 13 in the appendix. As a consequence, PPO with either KL penalty is approximately equivalent to natural policy gradient (NPG) [13] (explained in Appendix A), and in practice often result in similar performance.

4.1.2 Advantages of KL-regularization

Fixing Failure mode 2. KL-regularized PPO enjoys convergence guarantees when the parameterized policy class is closed under mixture [14]. This includes finite discrete action spaces such as in the example of Failure mode 2. We verify in Figure 8 that KL-regularized PPO indeed converges reliably on this example. KL-regularized PPO more effectively avoids suboptimal actions compared to standard PPO, especially as the number of actions increases. When comparing the two KL directions, the theoretically principled reverse-KL penalty (see Appendix A.3 for convergence guarantees) has a slight advantage over the forward-KL penalty.

Importantly, the convergence guarantees of PPO for KL-regularized surrogate objectives [14] are not preserved when using the standard clipping heuristic. Failure mode 2 (Figure 5 and Figure 8) illustrates one simple example where standard PPO fails even inside the regime of existing theoretical guarantees for KL-regularized PPO.

The myopic convergence of standard PPO compounds with the sampling effects in exploring high-dimensional action spaces. To clarify, while KL-regularized PPO always converges optimally as we vary the action dimension from 10 to 100 in Figure 8, it is not expected that the convergence

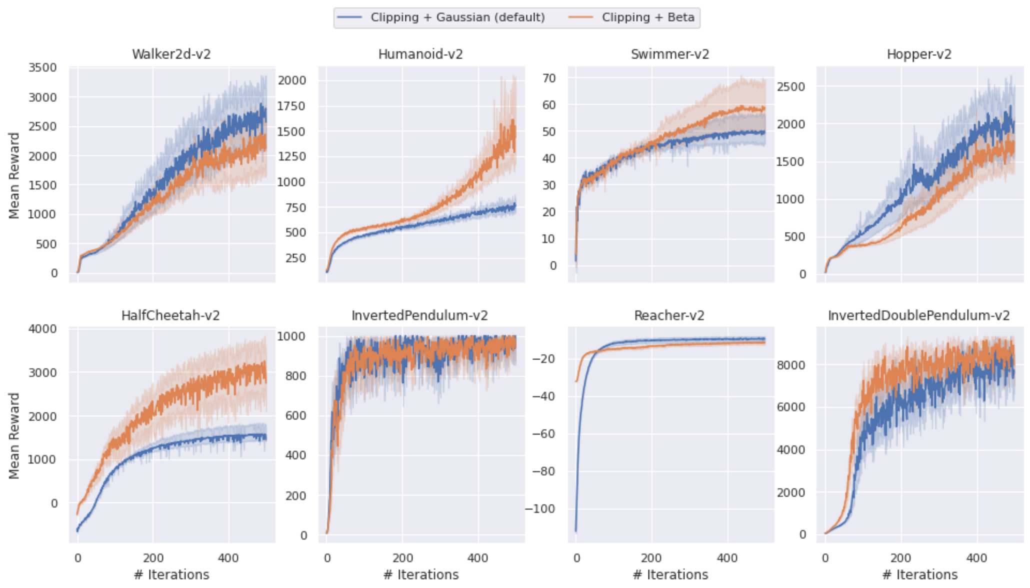


Figure 9: Beta policy significantly improves PPO performance on several MuJoCo tasks compared to Gaussian policy, while maintaining similar performance on other tasks. Most remarkably, Beta policy achieves 2x cumulative reward on Humanoid.

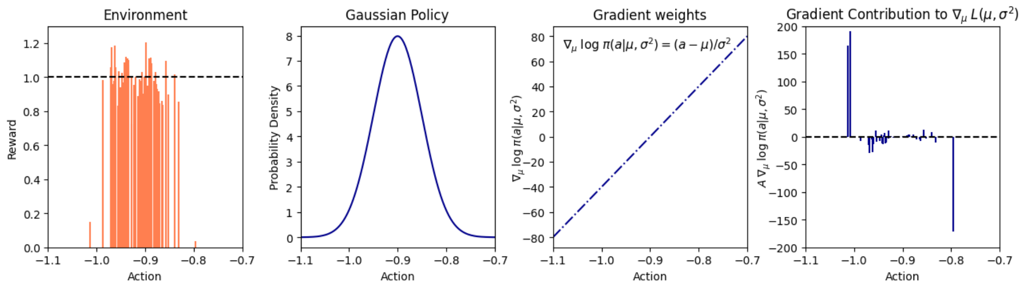
probability will stay at 1 as we arbitrarily increase the dimensions. This does not violate the theoretical regret bound for KL-regularized PPO, since the regret bound is based on the samples that the policy sees during training. When the action dimension is large compared to the batch size, the optimal action is very unlikely to be sampled, and hence the regret bound does not imply perfect convergence. Empirically, when we further increase the action dimension to 1000, KL-regularized PPO converges to optimal action on around 30% of the runs, compared to less than 10% for PPO with clipping.

Fixing Failure mode 1. KL-regularized PPO also mitigates Failure mode 1 on continuous action spaces, as illustrated in Figure 7. Even though its formal convergence guarantees do not extend to parameterized continuous action spaces with Gaussian policies, it is more effective in preserving the trust region, compared to clipping. As pointed out in Section 3.1, the clipped objective gets stuck once a large stochastic gradient step is taken and does not provide a reliable mechanism for the policy to come back to the trust region in future steps of the same round. In contrast, a large gradient step with the KL-regularized objective would cause large opposite gradients from the KL penalty, forcing the policy to move back closer to the old policy, before sampling new data.

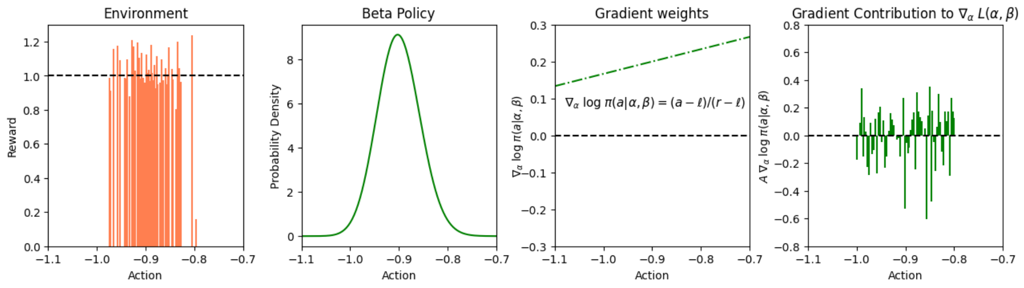
MuJoCo performance. Empirical results on MuJoCo tasks show that KL-regularized PPO has comparable performance to clipped PPO when in combination with Beta policy. As expected, KL direction does not significantly affect PPO performance, neither for Gaussian nor for Beta policies. The corresponding experimental results can be found in Figure 11 and Figure 12 in the appendix.

4.2 Beta policy parameterization on continuous action spaces

A second, stand alone fix for the continuous Failure modes 1 and 3, is to use Beta policy parameterizations instead of Gaussians. The Beta distribution (3b) is defined on a bounded interval, and parameterized by two shape parameters α and β. Following [9], we model α, β by passing neural network outputs through $x \mapsto \log(1+\exp(x))+1$ to ensure $\alpha, \beta \geq 1$. One might think of the finite support of Beta policies as a restriction, but our evidence in Section 3.3 has shown that Gaussian policies also require prior knowledge of the reward landscape to be initialized properly. As we will see, compared to Gaussian parameterization, Beta policy parameterization utilizes such prior knowledge more effectively.



(a) Gaussian policy parameterization



(b) Beta policy parameterization

Figure 10: A single PPO iteration in Failure mode 1 (Figure 3) for Gaussian policies (a) and Beta policies (b), with wide action boundary $[-3, 0]$ for fair comparison. We show: Sampled actions and their rewards (left), policy action distribution (middle left), weighting function of action samples in the aggregated gradients (middle right), and each action sample's contribution to the aggregated gradient (right). While the beta policy illustrated here has approximately the same shape as the Gaussian policy, the beta parameterization leads to more even weighting of the gradients across the actions sampled, thus preventing excessively large gradient updates caused by a few actions in the tail.

A number of existing high-profile PPO applications artificially discretize continuous action spaces [2, 7, 15] and use Softmax policy parameterization, possibly to avoid the failure modes with Gaussian policies. While discretizing the action space can effectively avoid the failure modes, it largely increases the number of policy parameters, especially in high-dimensional spaces. Beta policies, on the other hand, avoid the failure modes more efficiently without increasing the number of policy parameters.

Fixing failure mode 1. In the first failure mode (Section 3.1), the key problem is that Gaussian policies excessively upweigh actions in the tails, especially when the policy standard deviation is small. For Beta policy parameterization the likelihood of given actions in the tails do not change as rapidly and hence it weighs action samples more evenly, see Figure 10b in comparison to Figure 10a. This effectively avoids the first failure mode as demonstrated in Figure 7.

Note that both, the KL-regularized surrogate objective discussed in Section 4.1.2, and Beta policy parameterization, are stand-alone fixes for failure mode 1 through different mechanisms. While the KL-regularized surrogate objective does not change the steep weighting of gradients by the Gaussian score function, it more effectively preserves the trust region, compared to clipping and thereby reduceds the effect of large policy updates caused by actions in the tails.

To eliminate boundary effects, we intentionally set the action boundaries to be broader than the non-zero reward region. Hence, the gain of Beta policy observed in our example is orthogonal to the advantages of Beta policy related to boundary effects pointed out in [9].

Fixing failure mode 3. The culprit of Failure mode 3 (discussed in Section 3.3) is the large probability mass occupied by the suboptimal reward peak in the initial distribution. A natural fix is to initialize the policy more uniformly. Uniform initialization is difficult for Gaussian policies, as it requires a large variance which in turn puts large probability mass outside the action boundaries. With Beta policies, when $\alpha = \beta = 1$, the Beta distribution is by definition uniform across the interval. With the softplus parameterization of $\alpha = \log(1+\exp(x_\alpha))+1$, $\alpha = 1$ corresponds to $x_\alpha = -\infty$, and similarly for β. In practice, we initialize with $x_\alpha, x_\beta \approx -4$ and hence $\alpha, \beta \approx 1.018$ for a close approximation of the uniform distribution. This eliminates suboptimal convergence due to bad initialization and hence fixes the failure mode on multi-peak reward landscapes, as shown in Figure 6b.

MuJoCo performance. In addition to preventing the two above failure modes, we found that Beta policy parameterization also improves benchmark performance on some MuJoCo tasks, compared to Gaussian policies, as illustrated in Figure 9. In particular, PPO achieves 2x cumulative reward on the Humanoid-v2 task (1M steps) with beta policy, compared to Gaussian policy.

5 Discussion

Many algorithmic design choices in RL are made based on empirical comparisons and it is easy to forget that their justification is limited to the regime of the chosen benchmark tasks. Our study highlights, at the example of PPO, that widely accepted design choices do not necessarily generalize to new environments. We conduct failure mode analyses on synthetic test cases in order to understand and diagnose the broader implications of such design choices. We believe that such an approach offers a principled tool to isolate specific convergence issues of an algorithm and it augments classical benchmarks that often confound many different aspects of the environment.

We emphasize two main insights of our PPO study: First, *Beta policy parameterization* has attractive properties compared to standard Gaussian parameterization. i) It avoids excessively large stochastic gradient updates caused by actions in the distribution tails, which are particularly problematic in combination with the clipped surrogate objective. ii) It allows for approximately uniform initialization and is thus less sensitive to suboptimal initialization. iii) As pointed out by prior work [9] it also eliminates the bias towards boundaries in truncated Gaussians on bounded action spaces. Second, the advantage of clipping over the more principled *KL-regularized objective* is limited to the regime of MuJoCo benchmarks with Gaussian policies. On synthetic examples, KL-regularized objective is more robust. On MuJoCo benchmarks with Beta policy, KL-regularized PPO has similar or better performance than clipped PPO in most tasks.

While our main analysis focuses on the choice of surrogate objective and policy parameterization, we include several surprising findings around two other common implementation choices in PPO in Appendix E – advantage normalization and reward scaling. Briefly, we find that constant reward scaling can replace the complex reward scaling scheme studied in [10] without loss in performance; that advantage normalization helps PPO on some tasks but hurts on others; and that advantage normalization effectively anneals the mirror descent step size. We hope that our initial findings provide a valuable starting point for further work towards fully understanding these design choices.

Finally, we believe that our insights on PPO can also help to better understand other policy gradient algorithms such as Natural Policy Gradient (NPG) [13] and V-MPO [23], as they are closely connected to KL-regularized PPO as discussed in Appendix A.

Achnowledgements

The authors would like to thank Ben Recht for insightful discussions and for providing feedback on the initial draft of this work. In addition, the second author wishes to acknowledge support from the Swiss National Science Foundation Early Postdoc Mobility Fellowship Program.

Code availability

Our code is available at `https://github.com/chloechsu/revisiting-ppo`.

References

[1] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. *arXiv preprint arXiv:1908.00261*, 2019.

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[5] Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate bregman projections. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1200. SIAM, 2009.

[6] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[7] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[9] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 834–843, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[10] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations*, 2020.

[11] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. *arXiv preprint arXiv:1901.11275*, 2019.

[12] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[14] Boyi Liu, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural proximal/trust region policy optimization attains globally optimal policy. *arXiv preprint arXiv:1906.10306*, 2019.

[15] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

[16] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

[17] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[18] Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

[19] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

[20] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

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

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

[23] H Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, et al. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*, 2019.

[24] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[25] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Appendix

A KL-regularized PPO: convergence guarantees and connections to mirror descent, NPG, and V-MPO

We first focus on the connection between KL-regularized PPO and mirror descent. We extend and simplify existing expositions of this connection and derive convergence guarantees for PPO with parameterized policy classes.

Then, we also connect KL-regularized PPO to natural policy gradient (NPG) and V-MPO. We note that PPO with KL penalty in either direction is approximately equivalent to both Natural Policy Gradient [13] and mirror descent with I-projection, while V-MPO [23] is equivalent to mirror descent with reverse I-projection (also known as M-projection).

A.1 Connection to mirror descent

Mirror Descent has proven a valuable theoretical tool for deriving convergence results for policy gradient-type methods [1, 11, 14, 16].

We recall that mirror descent [6] in its general form is applied to an objective f and optimizes over a distribution family $\mathcal{P}$. Each mirror descent iteration optimizes for a linear approximation of f plus a KL-regularization term:

$$\pi^{k+1} := \arg\min_{\pi\in\mathcal{P}} \left\{ \langle \nabla f(\pi^k), \pi \rangle + \frac{1}{\eta} D_{\mathrm{KL}}(\pi \parallel \pi^k) \right\}, \tag{5}$$

where η denotes the step size and the negative entropy is used for the Bregman divergence.

When applied to the RL setting where the goal is to maximize expected advantage[1] we choose f to measure the expected advantage at each state s, i.e.,

$$f_s(\pi_\theta) = -\mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)}[A^{\pi_{\theta_k}}(s,a)]\ \forall s.$$

The corresponding infinite-dimensional gradient is $\nabla f_s(\pi_{\theta_k}) = A^{\pi_{\theta_k}}(s,\cdot)$. Hence, one step of mirror descent for this particular choice of f is equivalent to maximizing the $\mathcal{L}^{\mathrm{KL,reverse}}(\theta)$ PPO surrogate objective in (2c).

$$\pi_{\theta_{k+1}}(\cdot|s) := \arg\max_{\theta\in\Theta} \left\{ \mathop{\mathbb{E}}_{a\sim\pi_\theta(\cdot|s)}[A^{\pi_{\theta_k}}(s,a)] - \frac{1}{\eta} D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k}) \right\}. \tag{6}$$

A.2 Connection to projected multiplicytive weights

Mirror descent can be shown to follow the multiplicative weights (MW) update rule. Multiplicative weights (MW) minimize regret in an online learning setting with respect to payoff functions $m_k : \mathcal{X} \to [-\rho, \rho]$ with width ρ. Given a domain $\mathcal{X}$, MW starts with the uniform distribution π_0. In the k-th round, MW observes payoffs $m_k(x)$ for each $x \in \mathcal{X}$ and updates the distribution π_k according to the following update rule:

$$\pi_{k+1}(x) \propto \pi_k(x)\, e^{\eta m_k(x)}, \tag{7}$$

where $\eta \in (0, \frac{1}{2})$ is the MW learning rate. Intuitively, the update rule boosts elements with high payoff and down-weigh elements with low payoff[2].

[1]The advantage $A^\pi(s,a)$ is a proxy for how much better a particular action a is on state s compared to a randomly sampled action from $\pi(\cdot|s)$.

[2]The update rule is sometimes written in multiplicative weights literature with penalty instead of payoff, and with a linear factor $1+\eta m_k$ instead of the exponential factor $e^{\eta m_k}$. The differences lead to slightly different bounds. See Section 2 in [4] for a detailed discussion.

Information projection. The MW update is nonparameteric and hence the updated policy π^{MW} proposed by MW is not necessarily in the parameteric family $\mathcal{P}$. When working with parameterized distributions, information projection is a natural way to project the exact MW update to a parameterized family $\mathcal{P} = \{\pi_\theta : \theta \in \Theta\}$. Information projection finds the closest distribution within the family, as measured by KL-distance.

Definition A.1 (I-projection)**.** *The information projection of a probability distribution q onto a set of distributions $\mathcal{P}$ is*

$$\underset{\mathcal{P}}{\text{I-Proj}}\, q = \arg\min_{p\in\mathcal{P}} D_{\mathrm{KL}}(p \,\|\, q). \tag{8}$$

Therefore, reverse-KL-regularzied PPO also follows projected MW, with target updates proportional to exponentiated advantages:

$$\pi^{MW}(\cdot|s) \propto \pi_{\theta_k}(\cdot|s)\, \exp(\eta A^{\pi_{\theta_k}}(s,\cdot)); \quad \pi_{\theta_{k+1}}(\cdot|s) = \underset{\{\pi_\theta(\cdot|s):\, \theta\in\Theta\}}{\text{I-Proj}} \pi^{MW}, \tag{9}$$

With information projection, we can project the infinite-dimensional target MW update to a parameterized family. In practice, MW with information projection formulates an optimization objective, requiring only an implicit representation of the target MW update.

Using the definition of KL-divergence we can write

$$\begin{aligned} D_{\mathrm{KL}}\Big(\pi_\theta \,\|\, \frac{1}{Z_k}\pi_k\, e^{\eta m_k}\Big) &= \log Z_k + \int \pi_\theta(\log \pi_\theta - \log \pi_k - \eta m_k) \\ &= \log Z_k + D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_k) - \eta \mathop{\mathbb{E}}_{\pi_k}\Big[\frac{\pi_\theta}{\pi_k} m_k\Big]. \end{aligned}$$

Hence, for $\beta = 1/\eta$ minimizing the information projection distance is equivalent to maximizing the KL-regularized PPO objective $\mathcal{L}^{\text{KL,reverse}}$ (2c).

A.3 Convergence guarantees

Building on the connection of KL-regularized PPO to mirror descent and multiplicative weights, we review and extend existing convergence analysis.

The KL-regularized version of PPO inherits convergence guarantees of mirror descent for policy families that are closed under mixture. Thius includes softmax parameterization of discrete action spaces. In the following we will extend existing convergence analyses to cover a more practical setting of PPO, including approximate updates and adaptive step sizes.

Therefore, we extend existing MW convergence guarantees to approximately projected MW on closed and convex distribution families. A parameterized family $\mathcal{P} = \{\pi_\theta : \theta \in \Theta\}$ is convex when $\mathcal{P}$ is closed under mixture. Examples of convex families include distributions with a bounded support and distributions with a bounded probability density range. In contrast to MW , each update step in PPO (9) is only solved approximately via a fixed number of stochastic gradient steps. To capture the approximation, we introduce the notion of approximate projection.

Definition A.2 (Approximate I-projection)**.** *A distribution $\tilde{p} \in \mathcal{P}$ is an α-approximate information projection of q if for any $p \in \mathcal{P}$,*

$$D_{\mathrm{KL}}(p \,\|\, \tilde{p}) \le D_{\mathrm{KL}}(p \,\|\, \underset{\mathcal{P}}{\text{I-Proj}}\, q) + \alpha.$$

Intuitively, $\tilde{p}$ is an α-approximate projection of q if it is close to the exact projection $\text{I-Proj}_{\mathcal{P}}\, q$, such that the KL-distance from any other distribution in the family to $\tilde{p}$ is about the same as its KL-distance to the exact projection $\text{I-Proj}_{\mathcal{P}}\, q$.

When mirror descent does not solve for the arg max in (6) exactly, it corresponds to an approximate information projection of the multiplicate weights update rule. An *α-approximate projected MW* *update* is an α-approximate projection of the implicitly represented infinite-dimensional MW update distribution from the update rule (7).

We also review Bregman's theorem here, which will be used in the convergence analysis.

Theorem A.3 (Bregman)**.** *Let p, q be two distributions such that p is in the non-empty closed convex set Γ of measures. Then,*

$$D_{\mathrm{KL}}(p \parallel \underset{\Gamma}{\text{I-Proj}}\, q) + D_{\mathrm{KL}}(\underset{\Gamma}{\text{I-Proj}}\, q \parallel q) \leq D_{\mathrm{KL}}(p \parallel q).$$

Now we are ready to state the convergence guarantees for projected multiplicative weights. We first state Theorem A.4 in projected multiplicative weights language, and restate the theorem as Theorem A.5 in RL language. The proof extends main ideas from the proof of Lemma 4.1 in [5] with a finer-grain analysis, using Bregman's theorem.

Theorem A.4. *Let $\mathcal{P} = \{\pi_\theta : \theta \in \Theta\}$ be a family of distributions closed under mixture. Let $m_k : \mathcal{X} \to [-\rho_k, \rho_k]$ be payoff functions. Starting from any initial distribution π_{θ_0}, after K rounds of α-approximate projected multiplicative weights update with step size $\eta_1, \cdots, \eta_K$ such that $\eta_k \in (0, \frac{1}{\rho_k})$, the payoff difference between π_{θ^k} and the optimal distribution π_{θ^*} is bounded by*

$$\sum_{k=1}^{K} \eta_k \left(\underset{\pi_{\theta^*}}{\mathbb{E}}[m_k] - \underset{\pi_{\theta_k}}{\mathbb{E}}[m_k] \right) \leq D_{\mathrm{KL}}(\pi_{\theta^*} \parallel \pi_{\theta_0}) + \alpha K + \sum_{k=1}^{K} \eta_k^2 \underset{\pi_{\theta_k}}{\mathbb{E}}[m_k^2],$$

and in the special case of constant step size η and constant payoff function width ρ, we can simplify the bound as

$$\frac{1}{K} \sum_{k=1}^{K} \left(\underset{x \sim \pi_{\theta^*}}{\mathbb{E}}[m_k(x)] - \underset{x \sim \pi_{\theta_k}}{\mathbb{E}}[m_k(x)] \right) \leq \eta \rho^2 + \frac{\alpha}{\eta} + \frac{1}{\eta K} D_{\mathrm{KL}}(\pi_{\theta^*} \parallel \pi_0). \tag{10}$$

Proof. Let p_k^{MW} be the multiplicative weights update $p_k^{MW} = \pi_{\theta_k} e^{\eta m_k}/Z_k$. For any θ, by definition,

$$D_{\mathrm{KL}}(\pi_\theta \parallel p_k^{MW}) - D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k}) = -\int \pi_\theta \log \frac{p_k^{MW}}{\pi_{\theta_k}} = \log Z_k - \eta \underset{\pi_\theta}{\mathbb{E}}[m_k]. \tag{11}$$

Meanwhile, since m_k is bounded by $[-\rho_k, \rho_k]$, $\eta_k m_k$ is bounded by $[-1, 1]$, so using $e^x \leq 1 + x + x^2$ for $x \in [-1, 1]$,

$$Z_k = \underset{\pi_{\theta_k}}{\mathbb{E}}[e^{\eta_k m_k}] \leq 1 + \eta_k \underset{\pi_{\theta_k}}{\mathbb{E}}[m_k] + \eta_k^2 \underset{\pi_{\theta_k}}{\mathbb{E}}[m_k^2],$$

and using $log(1+x) \leq x$,

$$\log Z_k \leq \eta_k \underset{\pi_{\theta_k}}{\mathbb{E}}[m_k] + \eta_k^2 \underset{\pi_{\theta_k}}{\mathbb{E}}[m_k^2]. \tag{12}$$

Substituting into Equation 11, we have

$$D_{\mathrm{KL}}\left(\pi_\theta \parallel p_k^{MW}\right) - D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k}) \leq \eta_k \left(\underset{\pi_{\theta_k}}{\mathbb{E}}[m_k] - \underset{\pi_\theta}{\mathbb{E}}[m_k] \right) + \eta_k^2 \underset{\pi_{\theta_k}}{\mathbb{E}}[m_k^2] \tag{13}$$

By Bregman's Theorem,

$$D_{\mathrm{KL}}\left(\pi_\theta \parallel p_k^{MW}\right) \geq D_{\mathrm{KL}}\left(\pi_\theta \parallel \underset{\mathcal{P}}{\text{I-Proj}}\, p_k^{MW}\right). \tag{14}$$

Since $\pi_{\theta_{k+1}}$ is an α-approximation of $\text{I-Proj}_{\mathcal{P}}\, p_k^{MW}$,

$$D_{\mathrm{KL}}\left(\pi_\theta \parallel \underset{\mathcal{P}}{\text{I-Proj}}\, p_k^{MW}\right) \geq D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_{k+1}}) - \alpha. \tag{15}$$

Therefore, combining Equation 13, Equation 14, and Equation 15,

$$D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_{k+1}}) - D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_k}) \le \eta_k \left(\mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k] - \mathop{\mathbb{E}}_{\pi_\theta}[m_k] \right) + \eta_k^2 \mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k^2] + \alpha. \tag{16}$$

From the telescope sum of Equation 16 from $k = 1$ to K, we have

$$D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_K}) - D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_0}) \le \alpha K + \left(\sum_{k=1}^{K} \eta_k \left(\mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k] - \mathop{\mathbb{E}}_{\pi_\theta}[m_k] \right) \right) + \sum_{k=1}^{K} \eta_k^2 \mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k^2],$$

and hence

$$\sum_{k=1}^{K} \eta_k \left(\mathop{\mathbb{E}}_{\pi_\theta}[m_k] - \mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k] \right) \le D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_0}) + \alpha K + \sum_{k=1}^{K} \eta_k^2 \mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k^2].$$

In the special case of constant step size η, we get the desired bound

$$\frac{1}{K} \sum_{k=1}^{K} \mathop{\mathbb{E}}_{\pi_\theta}[m_k] - \frac{1}{K} \sum_{k=1}^{K} \mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k] \le \frac{\eta}{K} \sum_{k=1}^{K} \mathop{\mathbb{E}}_{\pi_{\theta_k}}[m_k^2] + \frac{\alpha}{\eta} + \frac{1}{\eta K} D_{\mathrm{KL}}(\pi_\theta \parallel \pi_{\theta_0}).$$

If we further substituting with $\mathbb{E}_{\pi_{\theta_k}}[m_k^2] \le \rho^2$ for constant payoff width ρ, we get the simpler but coarser bound. □

The $\frac{\alpha}{\eta}$ term captures the additional error caused by PPO not exactly optimizing the surrogate objective in each iteration. With $\alpha = 0$ and π_0 being the uniform distribution, Theorem A.4 can recover the regret bound for exact MW on finite domain $\mathcal{X}$ (see Theorem 2.3 in [4]), using $D_{\mathrm{KL}}(\pi \parallel \pi_0) \le \ln|\mathcal{X}|$ on discrete spaces.

Theorem A.5 (Restating Theorem A.4). *Let $\mathcal{P} = \{\pi_\theta : \theta \in \Theta\}$ be a family of policies closed under mixtur. Assume bounded advantages $\hat{A}^{\pi_{\theta_k}} \in [-\rho_k, \rho_k]$, and assume the gradient steps in each PPO iteration achieve an α-approximate projection of the surrogate objective (2c) on state s. Starting from initial policy π_{θ_0}, after K iterations with step sizes $\eta_1, \cdots, \eta_K$ such that $\eta_k \in (0, \frac{1}{\rho_k})$, we can bound the difference in advantages between π_{θ_k} and the optimal policy π_{θ^*} on state s by*

$$\sum_{k=1}^{K} \eta_k \left(\mathop{\mathbb{E}}_{a\sim\pi_{\theta^*}(\cdot|s)}[\hat{A}^{\pi_{\theta_k}}(s,a)] - \mathop{\mathbb{E}}_{a\sim\pi_{\theta_k}(\cdot|s)}[\hat{A}^{\pi_{\theta_k}}(s,a)] \right) \le$$
$$D_{\mathrm{KL}}(\pi_{\theta^*}(\cdot|s) \parallel \pi_{\theta_0}(\cdot|s)) + \alpha K + \sum_{k=1}^{K} \eta_k^2 \mathop{\mathbb{E}}_{a\sim\pi_{\theta_k}(\cdot|s)}[\hat{A}^{\pi_{\theta_k}}(s,a)^2],$$

and in the special case of constant step size η and constant advantage width ρ on discrete action spaces, we can simplify the bound as

$$\frac{1}{K} \sum_{k=1}^{K} \left(\max_a \hat{A}^{\pi_{\theta_k}}(s,a) - \mathop{\mathbb{E}}_{a\sim\pi_{\theta_k}(\cdot|s)}[\hat{A}^{\pi_{\theta_k}}(s,a)] \right) \le \eta\rho^2 + \frac{\alpha}{\eta} + \frac{1}{\eta K} D_{\mathrm{KL}}(\pi_{\theta^*}(\cdot|s) \parallel \pi_0(\cdot|s)),$$

meaning the average action under π_{θ_k} is close to the best action under advantage function $\hat{A}^{\pi_{\theta_k}}$.

On discrete action spaces, when using uniform initialization, the upper bound in Theorem A.5 holds with $\frac{1}{\eta K} D_{\mathrm{KL}}(\pi_{\theta^*} \| \pi_{\theta_0}) \le \log|\mathcal{A}|$. As the number of training iterations K increases, $\frac{1}{\eta K} D_{\mathrm{KL}}(\pi_{\theta^*} \| \pi_{\theta_0})$ vanishes, and the impact of initialization diminishes.

A.4 Connection to Natural Policy Gradient

The Hessian of the KL penalty term in either KL-direction is Fisher information matrix $F(\theta_k)$, and the second-order Taylor expansion is

$$D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\theta_k}) \approx D_{\mathrm{KL}}(\pi_{\theta_k} \,\|\, \pi_\theta) \approx \frac{1}{2}(\theta - \theta_k)^T F(\theta_{\mathrm{old}})(\theta - \theta_k).$$

Therefore, optimizing either KL-regularized objective (2b) or (2c) results in

$$\theta_{k+1} \approx \theta_k - \frac{1}{\beta} F^{-1}(\theta_k) \nabla_\theta L(\theta_k),$$

which is the natural gradient update of the unregularized policy gradient objective.

Each iteration of KL-regularized PPO is approximately equivalent to one natural gradient update in general. As a special case, on finite Markov Decision Processes (MDPs) with finite state space and softmax parameterization for finite action space, the Fisher information matrix is constant, and both NPG and KL-regularized PPO correspond to exact multiplicative weights.

A.5 Comparison with V-MPO

Compared to KL-regularized PPO as multiplicative weights with information projection (9), V-MPO corresponds to multiplicative weights with moment projection. Information projection minimizes KL distance $D_{\mathrm{KL}}(\pi_\theta \,\|\, p_{MW})$ to the target nonparametric distribution, where as moment project minimizes the flipped KL distance $D_{\mathrm{KL}}(p_{MW} \,\|\, \pi_\theta)$.

On-Policy Maximum A Posteriori Policy Optimization (V-MPO) [23] is also related to multiplicative weights. The E-step in V-MPO constructs the nonparametric target distribution ψ by using the same multiplicative weight updates rule as in (9), and the M-step projects the target distribution to the parameterized family by maximizing weighted maximum likelihood loss $\sum_{s,a} \psi(s,a) \log \pi_\theta(a|s)$. Maximizing this weighted log likelihood is equivalent to minimizing $D_{\mathrm{KL}}(\psi \,\|\, \pi_\theta)$, known as moment projection. V-MPO only takes samples corresponding to the top half advantages in the weighted maximum likelihood loss, which can be interpreted in the multiplicative weights framework as setting the bottom half advantages to $-\mathrm{inf}$.

While V-MPO and PPO are related in the multiplicative weights update form, information projection and moment projection results in different projections: when the nonparametric target distribution is multi-modal cannot be fit within the parametric family, information projection is more mode-seeking, whereas moment projection tends to be spread out more to cover the entire support.

B Derivation of surrogate objective gradients

We will use $\mathbb{1}$ to represent the indicator function and write $r(a|s) := \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}$ and $\hat{A}_{a,s} := \hat{A}^{\pi_{\text{old}}}(a,s)$ to simplify notation.

B.1 Unregularized objective

The gradient of the unregularized objective $\mathcal{L}(\theta) = \mathbb{E}_{a,s\sim\pi_{\text{old}}}\left[r(a|s)\hat{A}_{a,s}\right]$ corresponds to

$$\nabla_\theta \mathcal{L}(\theta) = \mathop{\mathbb{E}}_{a,s\sim\pi_{\text{old}}}\left[r(a|s)\hat{A}_{a,s}\nabla_\theta \log \pi_\theta(a|s)\right]$$

B.2 Clipped objective

The gradient of the clipped objective (2a) corresponds to:

$$\nabla_\theta \mathcal{L}^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[r(a|s)\hat{A}_{a,s}\nabla_\theta \log \pi_\theta(a|s)\cdot \mathbb{1}\left\{r(a|s)\in(1-\epsilon, 1+\epsilon) \text{ or } \operatorname{sgn}(r(a|s)-1)\neq \operatorname{sgn}(\hat{A}_{a,s})\right\}\right].$$

B.3 KL-regularized objectives

The gradient of the forward-KL objective (2b) is

$$\nabla_\theta \mathcal{L}^{\text{KL,forward}}(\theta) = \nabla_\theta L(\theta) - \beta\nabla_\theta D_{\text{KL}}(\pi_{\text{old}} \,\|\, \pi_\theta) = \mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[\left(\frac{\pi_\theta}{\pi_{\text{old}}}A + \beta\right)\nabla_\theta \log \pi_\theta\right]$$

and the gradient of the reverse-KL objective (2c) is

$$\nabla_\theta \mathcal{L}^{\text{KL,reverse}}(\theta) = \nabla_\theta L(\theta) - \beta\nabla_\theta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{old}}) = \mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[\frac{\pi_\theta}{\pi_{\text{old}}}\left(A - \beta\log(\frac{\pi_\theta}{\pi_{\text{old}}})\right)\nabla_\theta \log \pi_\theta\right].$$

To derive these expressions we first look at the KL penalty term. For forward KL we get

$$\begin{aligned}
\nabla_\theta D_{\text{KL}}(\pi_{\text{old}} \,\|\, \pi_\theta) &= \nabla_\theta \int_x \pi_{\text{old}}(x)\log(\frac{\pi_{\text{old}}(x)}{\pi_\theta(x)}) = \int_x \nabla_\theta\left(\pi_{\text{old}}(x)\log(\frac{\pi_{\text{old}}(x)}{\pi_\theta(x)})\right)\\
&= -\int_x \pi_{\text{old}}(x)\nabla_\theta(\log(\pi_\theta(x)))\\
&= -\mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[\nabla_\theta(\log\pi_\theta)\right]
\end{aligned}$$

and for reverse-KL

$$\begin{aligned}
\nabla_\theta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{old}}) &= \nabla_\theta \int_x \pi_\theta(x)\log(\frac{\pi_\theta(x)}{\pi_{\text{old}}(x)}) = \int_x \nabla_\theta\left(\pi_\theta(x)\log(\frac{\pi_\theta(x)}{\pi_{\text{old}}(x)})\right)\\
&= \int_x \nabla_\theta(\pi_\theta(x))\log(\frac{\pi_\theta(x)}{\pi_{\text{old}}(x)}) + \int_x \pi_\theta(x)\nabla_\theta\log(\pi_\theta(x))\\
&= \mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[\frac{\pi_\theta}{\pi_{\text{old}}}\log(\frac{\pi_\theta}{\pi_{\text{old}}})\nabla_\theta(\log\pi_\theta)\right],
\end{aligned}$$

where we used $\int_x \pi_\theta(x)\nabla_\theta\log(\pi_\theta(x)) = \int_x \nabla_\theta(\pi_\theta(x)) = \nabla_\theta\int_x \pi_\theta(x) = \nabla_\theta(1) = 0$.

When $\frac{\pi_{\text{old}}}{\pi_\theta} \approx 1$, by Taylor expansion

$$\frac{\pi_\theta}{\pi_{\text{old}}} \log\left(\frac{\pi_\theta}{\pi_{\text{old}}}\right) \approx \frac{\pi_\theta}{\pi_{\text{old}}} - 1 + \frac{1}{2}\left(\frac{\pi_\theta}{\pi_{\text{old}}} - 1\right)^2,$$

so using the identity $\int_x \pi_\theta(x) \nabla_\theta \log(\pi_\theta(x)) = 0$ again,

$$\begin{aligned}\nabla_\theta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{old}}) &\approx -\mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}[\nabla_\theta(\log \pi_\theta)] + \frac{1}{2}\mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[\left|\frac{\pi_\theta}{\pi_{\text{old}}} - 1\right|^2 \nabla_\theta(\log \pi_\theta)\right] \\ &= \nabla_\theta D_{\text{KL}}(\pi_{\text{old}} \,\|\, \pi_\theta) + \frac{1}{2}\mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[\left|\frac{\pi_\theta}{\pi_{\text{old}}} - 1\right|^2 \nabla_\theta(\log \pi_\theta)\right]\end{aligned}$$

Or equivalently, the difference between the two KL penalties is

$$\nabla_\theta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{old}}) - \nabla_\theta D_{\text{KL}}(\pi_{\text{old}} \,\|\, \pi_\theta) \approx \frac{1}{2}\mathop{\mathbb{E}}_{x\sim\pi_{\text{old}}}\left[|\frac{\pi_\theta}{\pi_{\text{old}}} - 1|^2 \nabla_\theta(\log \pi_\theta)\right]$$

This difference in gradients is only large if $\frac{\pi_\theta}{\pi_{\text{old}}}$ is far away from 1 where the score function $\nabla_\theta(\log \pi_\theta)$ has large magnitude. We empirically evaluate the correlation of $D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{old}})$ and $D_{\text{KL}}(\pi_{\text{old}} \,\|\, \pi_\theta)$ in Figure 13

B.4 Weighting of examples

We state in Table 1 the weighting of examples in the gradient calculation for each of the four surrogate objectives. Since $\mathbb{E}_{x\sim\pi_{\text{old}}}[\frac{\pi_\theta}{\pi_{\text{old}}}\nabla_\theta \log \pi_\theta] = 0$, the weightings are up to $\frac{\text{constant}}{A(x)}$. We chose the constants such that the weighting is 1 when $\pi_\theta = \pi_{\text{old}}$.

Table 1: Gradients interpreted as weighting of examples.

	Weighting of examples
$\nabla_\theta L$	1
$\nabla_\theta \mathcal{L}^{\text{KL,reverse}}$	$1 - \frac{\beta}{A(x)} \log\left(\frac{\pi_\theta(x)}{\pi_{\text{old}}(x)}\right) = 1 + \frac{\beta}{A(x)} \log\left(\frac{\pi_{\text{old}}(x)}{\pi_\theta(x)}\right)$
$\nabla_\theta \mathcal{L}^{\text{KL,forward}}$	$1 + \frac{\beta}{A(x)}\left(\frac{\pi_{\text{old}}(x)}{\pi_\theta(x)} - 1\right)$
$\nabla_\theta \mathcal{L}^{\text{CLIP}}$	$\mathbb{1}\left\{\frac{\pi_\theta(x)}{\pi_{\text{old}}(x)} \in (1-\epsilon, 1+\epsilon) \text{ or } \operatorname{sgn}\left(\frac{\pi_\theta(x)}{\pi_{\text{old}}(x)} - 1\right) \neq \operatorname{sgn}(A(x))\right\}$

Environment	Observation Dimension	Action Dimension	Action Range (per dim)
Walker2d	17	6	[-1, 1]
Humanoid	376	17	[-0.4, 0.4]
Swimmer	8	2	[-1, 1]
Hopper	11	3	[-1, 1]
HalfCheetah	17	6	[-1, 1]
InvertedPendulum	4	1	[-3, 3]
Reacher	11	2	[-1, 1]
InvertedDoublePendulum	11	1	[-1, 1]

Table 2: MuJoCo environment description.

C Experimental setups

C.1 Constructed environments

In all constructed environments, we use the same training configuration: In each PPO iteration, we sample 512 timesteps (16 batches, 32 in each minibatch) and run 10 epochs with learning rate 0.1. We chose a relatively high learning rate in order to illustrate the failure behavior in a small number PPO iterations for illustrative purposes. Similar failure examples also happen with varying batch sizes and learning rates.

Single-peak 1D environment (Figure 3). The reward is 1.0 on the interval $(-1.0, -0.8)$, and zero otherwise, with Gaussian noise of standard deviation 0.1. For Beta policy and discretized policy, we set the action bounds to be $[-1.5, 1.5]$. The discrete policy discretizes the action space uniformly in 0.1 increments.

Double-peak 1D environment (Figure 6). The reward is given by

$$r(a) = 1.1 \times \exp\{-1.2 \times (a+2)^2\} + 0.9 \times \exp\{-0.9 \times (a-1)^2\},$$

with Gaussian noise of standard deviation 0.1. The action bounds are $[-5, 5]$ for Beta policy.

Discrete grid environment (Figure 5). We evaluate PPO on discrete grid environments of different sizes, varying the number of actions from 10 to 100 in increments of 10. In an environment with n actions, the rewards are zero on $n/2$ actions, 0.5 on $(n-1)/2$ actions, and 1 on a single optimal action. We then also add Gaussian noise of scale 0.1 to rewards. This discrete environment satisfies the assumptions in Theorem A.5 for convergence guarantees. When evaluting the probability of converging to optimal action, we repeat 20 runs for each settin, and define 'converging to optimal action' as $\geq 95\%$ probability on the optimal action in the policy after 50 iterations. When inspecting the learned policies, we find a bimodal behavior that after 50 iterations the policy has either $\geq 95\%$ probability or $\leq 5\%$ probability on the optimal action.

C.2 MuJoCo experiments

We choose hyperparameters based on the default hyperparameter values in the original PPO paper [22] and a recent PPO ablation study [10]. For important hyperparameters related to policy optimization (clipping threshold, KL penalty coefficient, number of epochs per PPO iteration, minibatch size, etc.) we consider a range of hyperparameters, and find that the default values work well. See Table 3 for the range considered and the final values used.

Following [9], we parameterize the α and β parameters in Beta distribution by softplus, with an added constant 1 to ensure $\alpha, \beta \geq 1$.

Hyperparameter	Value	Range Considered
Total timesteps	1M	N/A
Timesteps per iteration (horizon)	2000	1000 - 4000
Discount factor (γ)	0.99	N/A
GAE discount (λ)	0.95	N/A
Minibatch size	64	32 - 512
Clipping Threshold (ϵ)	0.2	0.0 - 0.5
KL penalty coeff (β)	3.0	0.1 - 10.0
Policy # epochs	10	1 - 40
Policy LR	3×10^{-4}	1×10^{-5} - 5×10^{-4}
Policy network hidden layers	$[64, 64]$	N/A
Value # epochs	10	N/A
Value LR	2×10^{-5}	N/A
Value network hidden layers	$[64, 64]$	N/A

Table 3: PPO hyperparameters used for MuJoCo tasks. The clipping threshold and the KL penalty coefficient are exclusive, i.e., in the clipped objective version, the clipping threshold is 0.2 and the KL penalty coefficient is 0, while in the KL-regularized version the clipped threshold is 1×10^8 and the KL penalty coefficient is 3.

We evaluate on eight MuJoCo tasks in OpenAI gym as listed in Table 2. For each task, we report the mean cumulative episode reward and the 95% confidence interval over 10 runs in all the figures.

D Supplementary figures

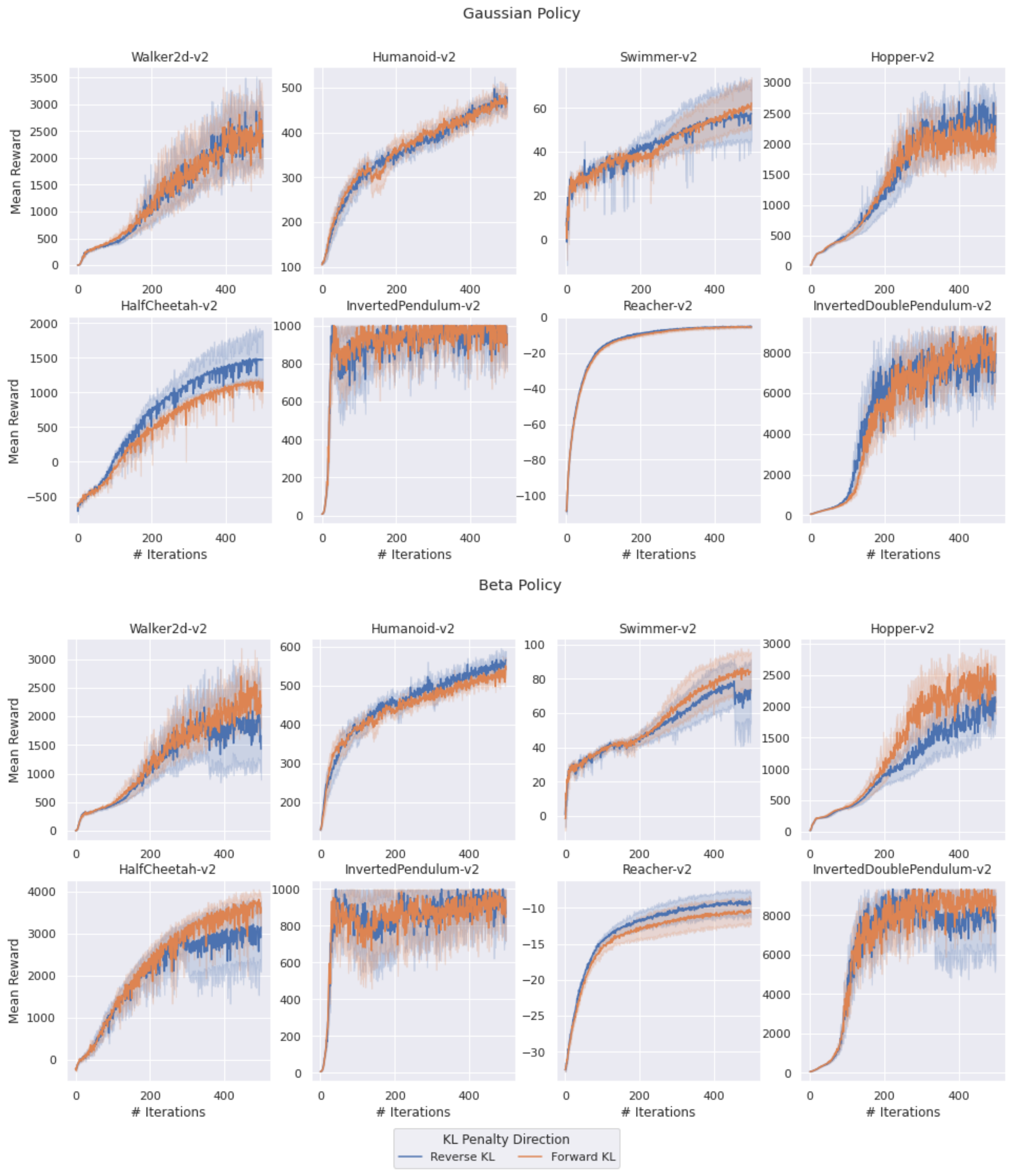


Figure 11: KL direction in KL penalty does not change PPO performance as measured by mean episode rewards on most MuJoCo locomotion tasks, whether using Gaussian policy or Beta policy.

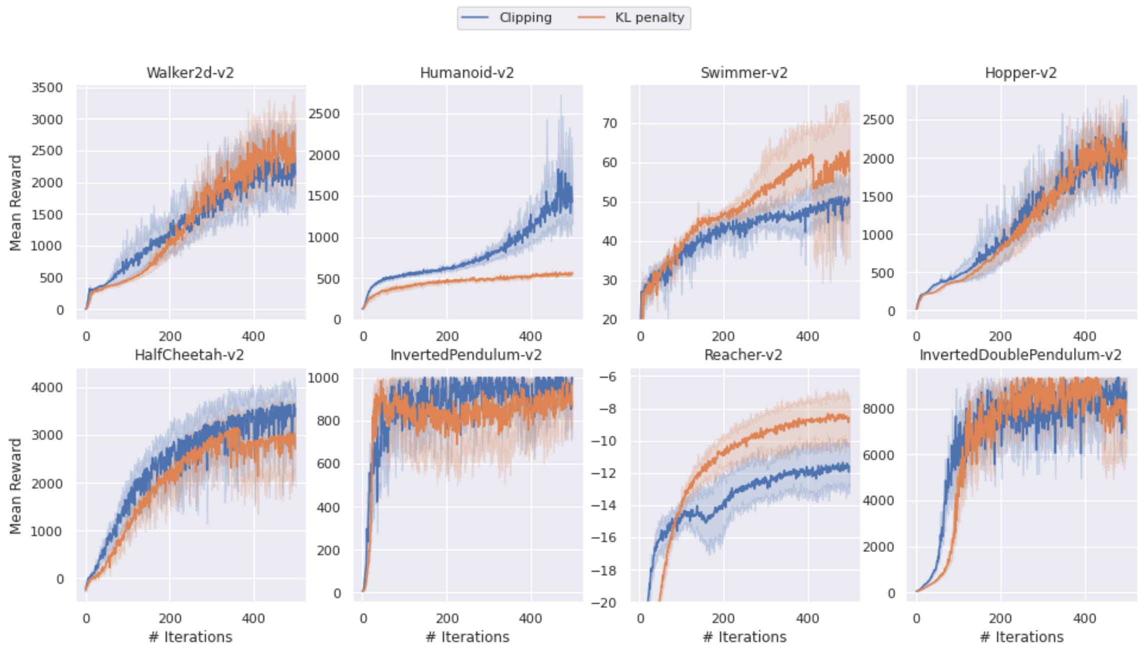


Figure 12: With Beta policy, KL-regularized PPO has similar or better performance than PPO with clipped surrogate objective on most MuJoCo tasks, with Humanoid being the exception. Humanoid performance here may not be the optimal for KL-regularized PPO, as it is higher dimensional than other tasks and may require significantly different hyperparameters tuned specifically for Humanoid.

# Actions	Games
3	Freeway, Skiing
4	Atlantis, Breakout
6	Bowling, Carnival, Pong, Pooyan, Qbert
7	Assault
8	Gopher, Phoenix, Tutankham
9	Asterix, Enduro
10	Amidar
14	Asteroids
18	Alien, Berzerk, Boxing, Centipede, Frostbite, Gravitar, Hero, Jamesbond, Kangaroo, Krull, Pitfall, Riverraid, Robotank, Seaquest, Solaris, Tennis, Venture, Zaxxon

Table 4: Action space dimensions of all Atari environments in Open AI Gym.

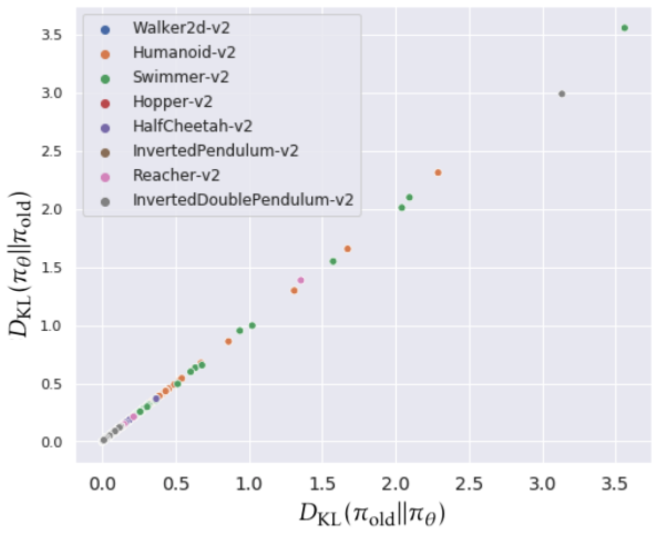


Figure 13: The empirically observed KL divergence in the two directions is nearly identical when evaluated on MuJoCo tasks.

E Additional studies on PPO: reward scaling and advantage normalization

We include additional studies on the effects of reward scaling and advantage normalization, two common design choices in most PPO implementations.

E.1 Reward scaling

Previous work [10] has highlighted a surprising observation that reward scaling is crucial for PPO's success. The reward scaling scheme studied in [10] is complex and unintuitive, normalizing rewards by the standard deviation of cumulative returns. We clarify that a simple constant reward scaling scheme can replace the more complex scaling and achieve the same performance on MuJoCo tasks (Figure 14).

We hypothesize that the constant reward scaling is important for matching the scale of rewards with the scale of initialized value functions, since the rewards are compared to the value scores in advantage estimation. To test this hypothesis, we visualize the distribution of estimated advantages during the initial 100 steps of training (Figure 15). If this hypothesis were to be true, the advantage distributions are "bumpy" before reward scaling because they were dominated by the reward values, and become "smoother" after reward scaling as they strike more of a balance between rewards and value functione estimates.

In the Humanoid, Walker, and Hopper tasks, reward scaling results in a huge difference in PPO performance, more than doubling the average reward. Interestingly, when comparing Figure 14 and Figure 15, Hopper and Humanoid, and Walker are the same tasks where the reward scaling scheme has the largest effect on the normalized advantage distributions. Reward scaling centers and smooths the advantage distributions in those tasks.

Further work is required to determine the affects of reward scaling and whether it can be compensated by better value function initialization.

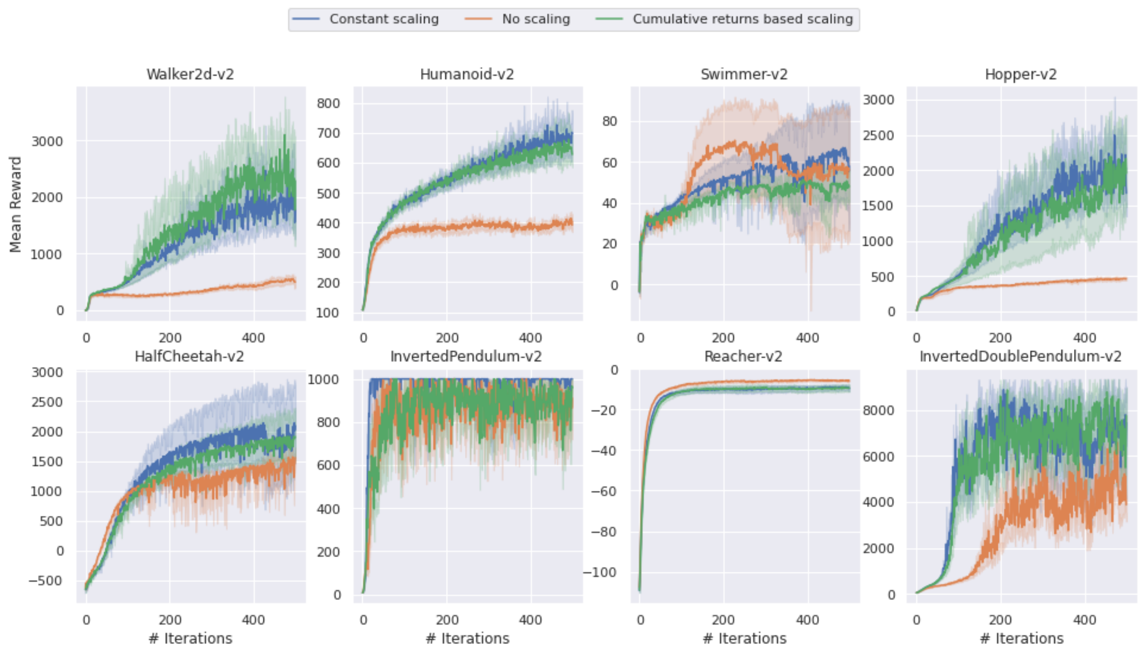


Figure 14: Reward scaling is critical for PPO's success. Simple constant scaling matches more complex cumulative returns based scaling.

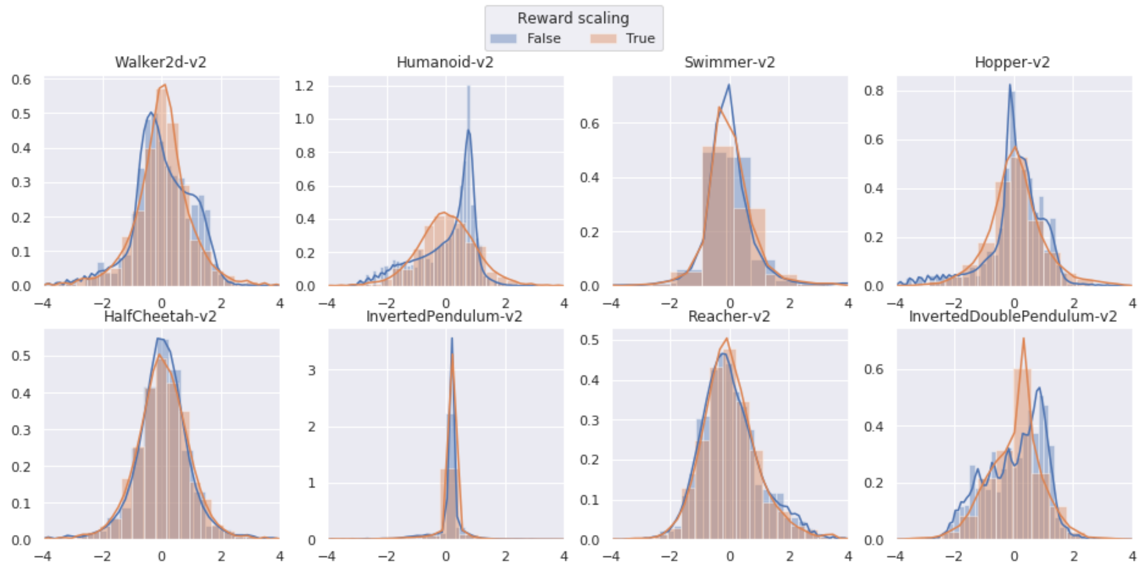


Figure 15: The distribution of normalized advantages, with and without the reward scaling scheme, in the initial phase of training. The histograms are aggregated over normalized advantage values from step 0, 10, 20, $\cdots$, 100.

E.2 Advantage normalization

Advantage normalization is another common heuristic in PPO. While it increases PPO performance on some MuJoCo tasks, it in fact reduces performance on some other tasks (Figure 16).

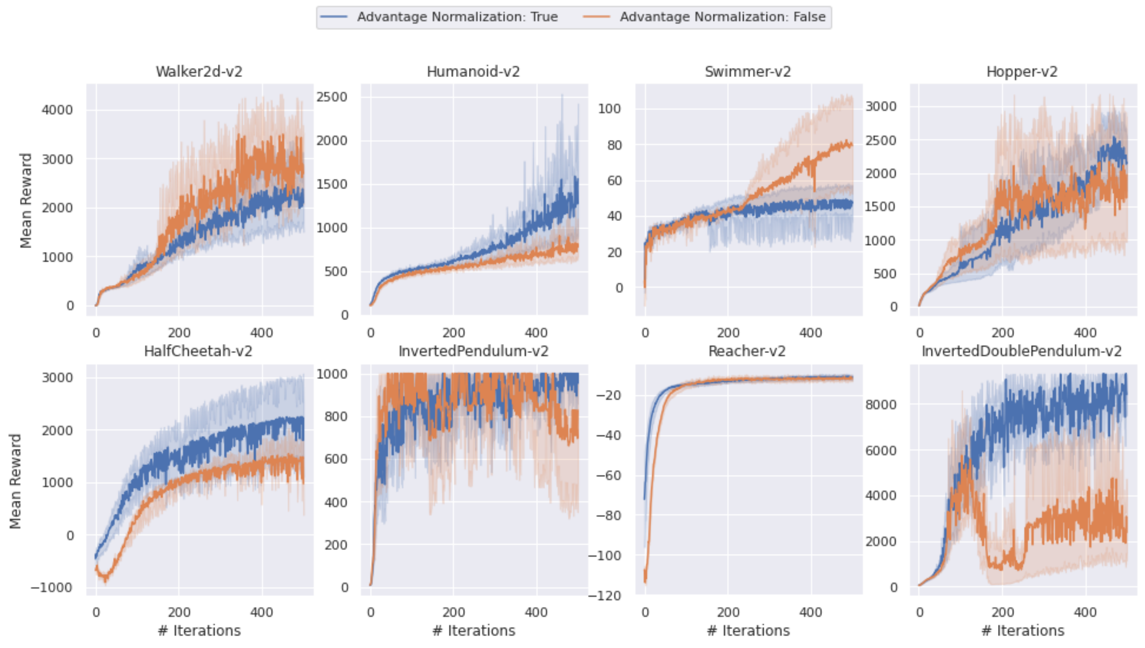


Figure 16: Comparison of PPO performance with vs without advantage normalization.

In light of the mirror descent connection (Appendix Section A), we can intrepret advantage normalization as dynamically adjusted mirror descent step size. Our extended analysis in Theorem A.5

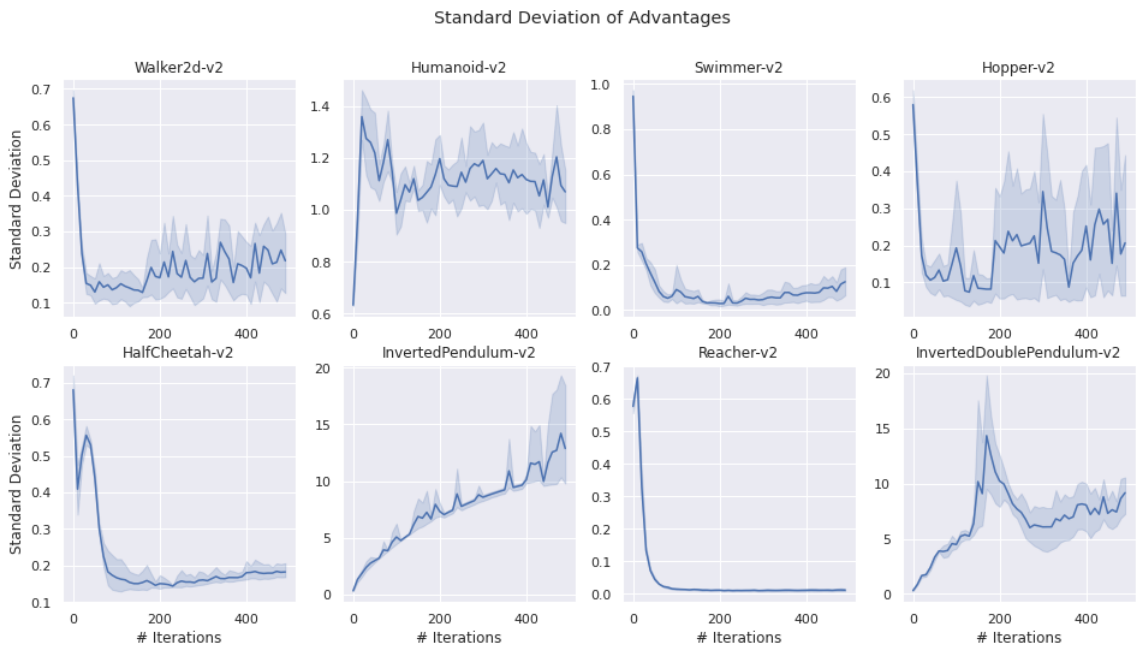


Figure 17: Standard deviation of advantage estimates. Over the course of training, the standard deviation either decreases rapidly and flattens, or gradually increases, depending on the environment.

accounts for advantage normalization and provides insights on how it influences PPO convergence bounds.

When inspecting the standard deviation of advantage estimates before normalization (Figure 17), we notice that the standard deviation either decreases rapidly and flattens, or gradually increases, depending on the environment. This suggests that advantage normalization effectively anneals the mirror descent step size over the course of training.

It remains interesting in future work to determine more precise conditions of when advantage normalization helps or hurts.