A. Detailed backgroud

We define a Markov Decision Process (MDP) as a tuple $(\mathcal{S}, \mathcal{A}, R, \gamma, P)$, where $\mathcal{S}$ represent the set of environments states, $\mathcal{A}$ represent the set of agent actions, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, γ is the discount factor, and $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the state transition probability distribution. The goal in reinforcement learning is to learn a policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$, parameterized by θ, such that the expected cumulative discounted reward (known as returns) is maximized. Formally,

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] . \tag{5}$$

Policy gradient methods directly optimize a paraterized policy function (also known as *actor network*). The central idea behind policy gradient methods is to perform stochastic gradient ascent on expected return (Eq. 5) to learn parameters θ. Under mild conditions (Sutton et al., 2000), the gradient of the Eq. 5 can be written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \nabla_\theta \log(\pi_\theta(a_t|s_t)) \right] ,$$

where $\tau \sim \pi_\theta$ are trajectories sampled according to $\pi_\theta(\tau)$ and $J(\theta)$ is the objective maximised in Eq. 5. With the observation that action a_t only affects the reward from time t onwards, we re-write the objective $J(\theta)$, replacing returns using the Q-function, i.e., the expected discounted reward after taking an action a at state s and following π_θ afterwards. Mathematically, $Q_{\pi_\theta}(s, a) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) | a_t = a, s_t = s \right]$. Using the Q-function, we can write the gradient of the objective function as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} Q_{\pi_\theta}(s_t, a_t) \nabla_\theta \log(\pi_\theta(a_t|s_t)) \right] .$$

However, the variance in the above expectation can be large, which raises difficulties for estimating the expectation empirically. To reduce the variance of this estimate, a baseline is subtracted from the Q-function—often the value function or expected cumulative discounted reward starting at a certain state and following a given policy i.e., $V_{\pi_\theta}(s) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) | s_t = s \right]$. The network that estimates the value function is often referred to as *critic*. Define $A_{\pi_\theta}(s_t, a_t) = Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t)$ as the *advantage* of performing action a_t at state s_t. Incorporating an advantage function, the gradient of the objective function can be written:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} A_{\pi_\theta}(s_t, a_t) \nabla_\theta \log(\pi_\theta(a_t|s_t)) \right] . \tag{6}$$

Eq. 6 is the näive actor-critic objective and is used by A2C.

Trust region methods and PPO. Since directly optimizing the cumulative rewards can be challenging, modern policy gradient optimization algorithms often optimize a surrogate reward function in place of the true reward. Most commonly, the surrogate reward objective includes a likelihood ratio to allow importance sampling from a behavior policy π_0 while optimizing policy π_θ, such as the surrogate reward used by Schulman et al. (2015a):

$$\max_\theta \mathbb{E}_{(s_t, a_t) \sim \pi_0} \left[\frac{\pi_\theta(a_t, s_t)}{\pi_0(a_t, s_t)} \hat{A}_{\pi_0}(s_t, a_t) \right] , \tag{7}$$

where $\hat{A}_\pi = \frac{A_\pi - \mu(A_\pi)}{\sigma(A_\pi)}$ (we refer to this as the *normalized advantages*). However, the surrogate is indicative of the true reward function only when π_θ and π_0 are close in distribution. Different policy gradient methods (Schulman et al., 2015a; 2017; Kakade, 2002) attempt to enforce the closeness in different ways. In Natural Policy Gradients (Kakade, 2002) and Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a), authors utilize a conservation policy iteration with an explicit divergence constraint which provides provable lower bounds guarantee on the improvements of the parameterized policy. On the other hand, PPO (Schulman et al., 2017) implements a clipping heuristic on the likelihood ratio of the surrogate reward function to avoid excessively large policy updates. Specifically, PPO optimizes the following objective:

$$\max_{\theta_t} \mathbb{E}_{(s_t, a_t) \sim \pi_{\theta_{t-1}}} \left[\min \left(\rho_t \hat{A}_{\pi_{\theta_{t-1}}}(s_t, a_t), \text{clip}(\rho_t, 1 - \epsilon, 1 + \epsilon) \hat{A}_{\pi_{\theta_{t-1}}}(s_t, a_t) \right) \right] , \tag{8}$$

where $\rho_t := \frac{\pi_{\theta_t}(a_t,s_t)}{\pi_{\theta_{t-1}}(a_t,s_t)}$ and $\text{clip}(x, 1-\epsilon, 1+\epsilon)$ clips x to stay between $1+\epsilon$ and $1-\epsilon$. We refer to ρ_t as *likelihood-ratios*. Due to a minimum with the unclipped surrogate reward, the PPO objective acts as a pessimistic bound on the true surrogate reward. As in standard PPO implementation, we use Generalized Advantage Estimation (GAE) (Schulman et al., 2015b). Moreover, instead of fitting the value network via regression to target values (denoted by V_{trg}):

$$\min_{\theta_t} \mathbb{E}_{s_t \sim \pi_{\theta_{t-1}}} \left[\left(V_{\theta_t}(s_t) - V_{trg}(s_t) \right)^2 \right], \tag{9}$$

standard implementations fit the value network with a PPO-like objective:

$$\min_{\theta_t} \mathbb{E}_{s_t \sim \pi_{\theta_{t-1}}} \max \left\{ \left(V_{\theta_t}(s_t) - V_{trg}(s_t) \right)^2, \left(\text{clip} \left(V_{\theta_t}(s_t), V_{\theta_{t-1}}(s_t) - \varepsilon, V_{\theta_{t-1}}(s_t) + \varepsilon \right) - V_{trg}(s_t) \right)^2 \right\}, \tag{10}$$

where ϵ is the same value used to clip probability ratios in PPO's loss function (Eq. 8). PPO uses the following training procedure: At any iteration t, the agent creates a clone of the current policy π_{θ_t} which interacts with the environment to collects rollouts $\mathcal{S}$ (i.e., state-action pair $\{(s_i, a_i)\}_{i=1}^N$). Then the algorithm optimizes the policy π_θ and value function for a fixed K gradient steps on the sampled data $\mathcal{S}$. Since at every iteration the first gradient step is taken on the same policy from which the data was sampled, we refer to these gradient updates as *on-policy* steps. And as for the remaining $K-1$ steps, the sampling policy differs from the current agent, we refer to these updates as *off-policy* steps.

Throughout the paper, we consider a stripped-down variant of PPO (denoted PPO-NoClip) that consists of policy gradient with importance weighting (Eq. 7), but has been simplified as follows: i) no likelihood-ratio clipping, i.e., no *objective function clipping*; ii) value network optimized via regression to target values (Eq. 9) without *value function clipping*; and iii) no *gradient clipping*. Overall PPO-NoClip uses the following objective:

$$\max_{\theta} \mathbb{E}_{(s_t,a_t)\sim\pi_0} \left[\frac{\pi_\theta(a_t,s_t)}{\pi_0(a_t,s_t)} \hat{A}_{\pi_0}(s_t,a_t) - c(V_{\theta_t} - V_{targ})^2 \right].$$

where c is a coefficient of the value function loss (tune as a hyperparameter). Moreover, no gradient clipping is incorporated in PPO-NoClip. One may argue that since PPO-NoClip removes the clipping heuristic from PPO, the unconstrained maximization of Eq. 1 may lead to excessively large policy updates. In App. E, we empirically justify the use of Eq. 1 by showing that with the small learning rate used in our experiments (optimal hyperparameters in Table 1), PPO-NoClip maintains a KL based trust-region like PPO throughout the training. We elaborate this in App. E.

B. Details on estimators

We now formalize our setup for studying the distribution of gradients. Throughout the paper, we use the following definition of the heavy-tailed property:

Definition 2 (Resnick (2007))**.** *A non-negative random variable w is called* heavy-tailed *if its tail probability $F_w(t) := P(w \geq t)$ is asymptotically equivalent to $t^{-\alpha^*}$ as $t \to \infty$ for some positive number α^*. Here α^* determines the heavy-tailedness and α^* is called tail index of w.*

For a heavy-tailed distribution with index α^*, its α-th moment exist only if $\alpha < \alpha^*$, i.e., $\mathbb{E}[w^\alpha] < \infty$ iff $\alpha < \alpha^*$. A value of $\alpha^* = 1.0$ corresponds to a Cauchy distribution and $\alpha^* = \infty$ (i.e., all moments exist) corresponds to a Gaussian distribution. Intuitively, as α^* decreases, the central peak of the distribution gets higher, the valley before the central peak gets deeper, and the tails get heavier. In other words, the lower the tail-index, the more heavy-tailed the distribution. However, in the finite sample setting, estimating the tail index is notoriously challenging (Simsekli et al., 2019; Danielsson et al., 2016; Hill, 1975).

In this study, we explore three estimators as heuristic measures to understand heavy tails and non-Gaussianity of gradients.

- *Alpha-index estimator*. This estimator was proposed in (Mohammadi et al., 2015) for symmeteric α-stable distributions and was used by (Simsekli et al., 2019) to understand the noise behavior of SGD. This estimator is derived under the (strong) assumption that the stochastic Gradient Noise (GN) vectors are coordinate-wise independent and follow a symmetric alpha-stable distribution. Formally, let $\{X_i\}_{i=1}^N$ be a collection of $N = mn$ (centered) random variables. Define $Y_i = \sum_{j=1}^m X_{j+(i-1)m}$ for $i \in [n]$. Then, the estimator is given by

$$\frac{1}{\alpha} := \frac{1}{\log m} \left(\frac{1}{n} \sum_{i=1}^n \log |Y_i| - \frac{1}{n} \sum_{i=1} N \log |X_i| \right).$$

Instead of treating each co-ordinate of gradient noise as an independent scalar, we use these estimators on gradient norms as discussed in Zhang et al. (2019b). With alpha-index estimator, smaller alpha-index value signify higher degree of heavy-tailedness.

- *Anderson-Darling test* (Anderson & Darling, 1954) on random projections of GN to perform Gaussianity testing. Panigrahi et al. (2019) proposed the Gaussianity test on the projections of GN along 1000 random directions. Their estimate is then the fraction of directions accepted by the Anderson Darling test. While this estimator is informative about the Gaussian behavior, it is not useful to quantify and understand the trends of heavy-tailedness if the predictor nature is non-Gaussian.

- To our knowledge, the deep learning literature has only exploredthese two estimators for analyzing the heavy-tailed nature of gradients. *(iii)* Finally, in our work, we propose using kurtosis *Kurtosis.* To quantify the heavy-tailedness relative to a normal distribution, we measure kurtosis (fourth standardized moment) of the gradient norms. Given samples $\{X_i\}_{i=1}^N$, the kurtosis κ is given by

$$\kappa = \frac{\sum_{i=1}^N (X_i - \bar{X})^4 / N}{\left(\sum_{i=1}^N (X_i - \bar{X})^2 / N\right)^2},$$

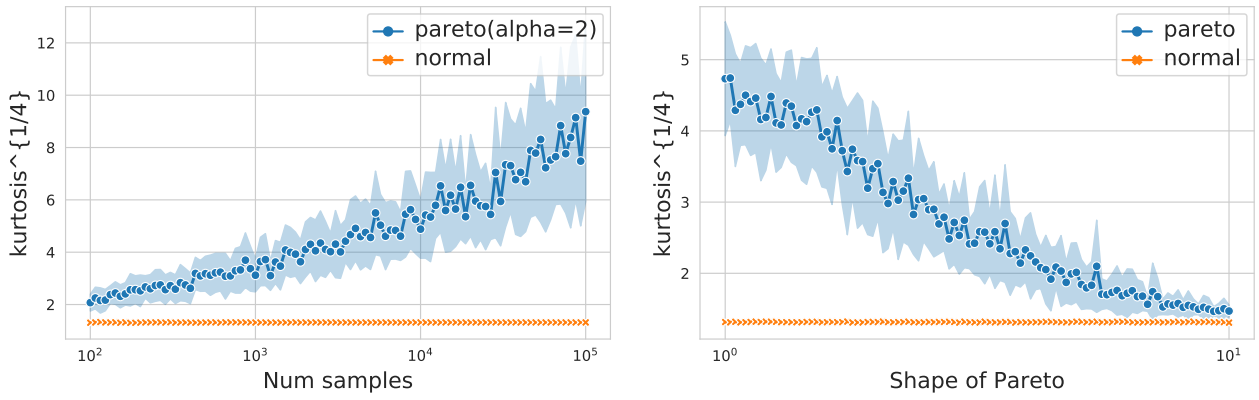
  where $\bar{X}$ is the empirical mean of the samples.

Note that both α-index and Anderson-Darling need very strong assumptions to be valid. (i) α-index requires that the true distribution is symmetric and α-stable. In the multivariate setting, the test proposed by Simsekli et al. (2019) relies on the covariance being isotropic. These theoretical limitations also lead to practical consequences. Specifically, we found that for low-rank Gaussian distributions (for which ideally $\alpha = 2$), the existing estimators report an $\alpha = 1.1$, wrongly suggesting heavy-tailendess. Similar limitations were pointed out in recent works (Zhang et al., 2019b; Xie et al., 2021). (ii) Anderson-Darling tests for Gaussianity and is not useful in quantifying the degree of heavy-tailedness. Moreover, the test fails for sub-Gaussian distributions such as uniform distribution.

On the other hand, our proposed estimator doesn't require symmetry or Gaussianity, and works well in the aforementioned pathological situations arising in practice. Moreover, well-known fat tailed distributions such as Student's t-distribution, exponential distribution, etc., have higher Kurtosis than normal distrbution. Even for distributions with less than 4 moments, empirical Kurtosis can be used to understand the "relative" trends in tail behavior for different distributions at fixed sample sizes (figure 6).

B.1. Synthetic study

In Figure 6, we show the trends with varying tail index and sample sizes. Clearly as the tail-index increases, i.e., the shape parameter increases, the kurtosis decreases (signifying its correlation to capture tail-index). Although for tail-index smaller than 4 the kurtosis is not defined, we plot empirical kurtosis and show its increasing trend sample size. We fix the tail index of Pareto at 2 and plot finite sample kurtosis and observe that it increases almost exponentially with the sample size. These two observations together hint that kurtosis is a neat surrogate measure for heavy-tailedness.



(a) (b)

Figure 6. Kurtosis plots. Analysis on norms of 100-dimensional vectors such that each coordinate is sampled iid from Pareto distribution or normal distribution. (a) Variation in kurtosis ($\kappa^{1/4}$) as the sample size is varied for samples from normal distribution and Pareto with tail index 2 (i.e, $\alpha = 2$). (b) Variation in kurtosis ($\kappa^{1/4}$) as the shape of Pareto is varied at fix sample size.

C. GMOM Algorithm

Algorithm 2 GMOM

input : Samples $S = \{x_1, \ldots, x_n\}$, number of blocks b
$m = \lfloor n/b \rfloor$.
for i in $1 \ldots b$ **do**
  $\hat{\mu}_i = \sum_{j=0}^{m} x_{j+i*m}$/ m.
end for
$\hat{\mu}_{\text{GMOM}} = \text{WEISZFELD}(\hat{\mu}_1, \ldots, \hat{\mu}_b)$.
output : Estimate $\hat{\mu}_{\text{GMOM}}$

Algorithm 3 WEISZFELD

input : Samples $S = \{\mu_1, \ldots, \mu_b\}$, number of blocks b
Initialize μ arbitrarily.
for iteration $\leftarrow 1, \ldots, n$ **do**
  $d_j := \frac{1}{\|\mu - \mu_j\|_2}$ for j in $1, \ldots, b$.
  $\mu := \left(\sum_{j=1}^{b} \mu_j d_j\right) / \left(\sum_{j=1}^{b} d_j\right)$
end for
output : Estimate μ

D. Experimental setup for gradient distribution study

Recall that PPO uses the following training procedure: At any iteration t, the agent creates a clone of the current policy π_{θ_t} which interacts with the environment to collects rollouts $\mathcal{S}$ (i.e., state-action pair $\{(s_i, a_i)\}_{i=1}^{N}$). Then the algorithm optimizes the policy π_θ and value function for a fixed K gradient steps on the sampled data $\mathcal{S}$. Since at every iteration the first gradient step is taken on the same policy from which the data was sampled, we refer to these gradient updates as **on-policy steps**. And as for the remaining $K - 1$ steps, the sampling policy differs from the current agent, we refer to these updates as **off-policy steps**. For all experiments, we aggregate our estimators across 30 seeds and 8 environments. We do this by first computing the estimators for individual experiments and then taking the sample mean across all runs. We now describe the exact experimental details.

In all of our experiments, for each gradient update, we have a batch size of 64. Hence for an individual estimate, we aggregate over 64 samples (batch size in experiments) to compute our estimators. For Anderson Darling test, we use 100 random directions to understand the behavior of stochastic gradient noise.

On-policy heavy-tailed estimation. At every on-policy gradient step (i.e. first step on newly sampled data), we freeze the policy and value network, and save the sample-wise gradients of the actor and critic objective. The estimators are calculated at every tenth on-policy update throughout the training.

Off-policy heavy-tailed estimation At every off-policy gradient step (i.e. the gradient updates made on a fixed batch of data when the sampling policy differs from the policy being optimized), we freeze the policy and value network, and save the sample-wise gradients of the actor and critic objective. Then at various stages of training, i.e., initialization, 50% max

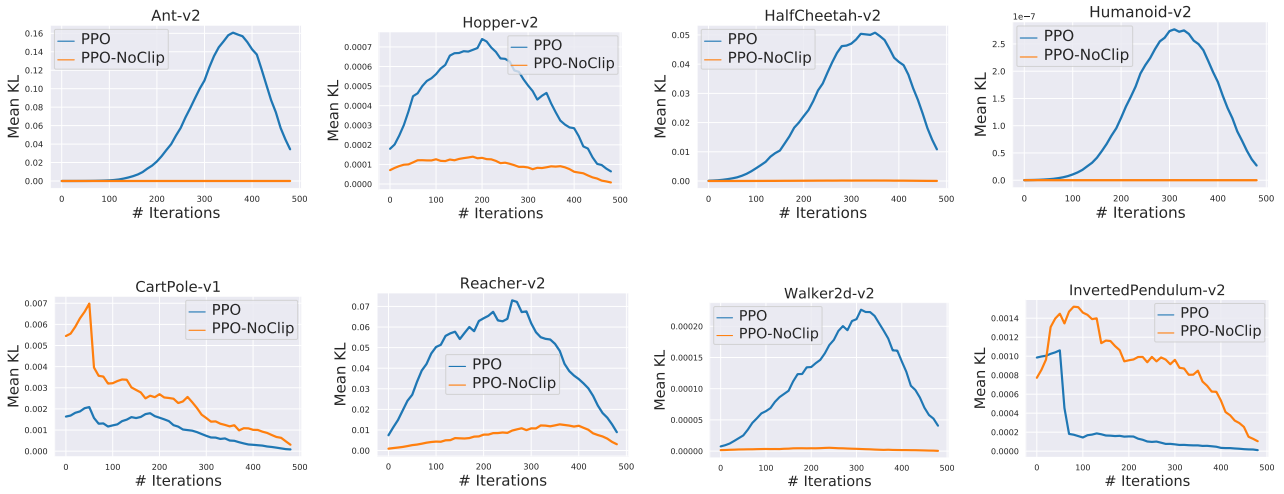


Figure 7. **KL divergence between current and previous policies with the optimal hyperparameters (parameters in Table 1).** We measure the mean empirical KL divergence between the policy obtained at the end of off-policy training (after every 320 gradient steps) and the sampling policy at the beginning of every training iteration. The quantities are measured over the state-action pairs collected in the training step (Engstrom et al. (2019) observed similar results with both unseen data and training data). We observe that both the algorithms maintain a KL based trust region. The trend with KL divergence in PPO matches with the observations made in Engstrom et al. (2019) where they also observed that it peeks in halfway in training.

reward and max reward (which corresponds to different batches of sampled data), we fix the collected trajectories and collect sample-wise gradients for the 320 steps taken. We now elaborate the exact setup with one instance, at 50% of the maximum reward. First, we find the training iteration where the agent achieves approximately 50% of the maximum reward individually for each environment. Then at this training iteration, we freeze the policy and value network and save the sample-wise gradients of the actor and critic objective for off-policy steps.

Analysis of PPO-NoClip with progressively applying PPO heuristics. We compute the gradients for the off-policy steps taken with the PPO-NoClip objective as explained above. Then at each gradient step, we progressively add heuristics from PPO and re-compute the gradients for analysis. Note that we still always update the value and policy network with PPO-NoClip objective gradients.

E. Mean KL divergence between current and previous policy

Enforcing a trust region is a core algorithmic property of PPO and TRPO. While the trust-region enforcement is not directly clear from the reward curves or heavy-tailed analysis, inspired by Engstrom et al. (2019), we perform an additional experiment to understand how this algorithmic property varies with PPO and our variant PPO-NoClip with optimal hyperparameters. In Fig 7, we measure mean KL divergence between successive policies of the agent while training with PPO and PPO-NoClip. Recall that while PPO implements a clipping heuristics in the likelihood ratios (as a surrogate to approximate the KL constraint of TRPO), we remove that clipping heuristics in PPO-NoClip.

Engstrom et al. (2019) pointed out that trust-region enforced in PPO is heavily dependent on the method with which the clipped PPO objective is optimized, rather than on the objective itself. Corroborating their findings, we indeed observe that with optimal parameters (namely small learning rate used in our experiments), PPO-NoClip indeed manages to maintain a trust region with mean KL metric (Fig 7) on all 8 MuJoCo environments. This highlights that instead of the core algorithmic objective used for training, the size of the step taken determines the underlying objective landscape, and its constraints. On a related note, Ilyas et al. (2018) also highlighted that the objective landscape of PPO algorithm in the typical sample-regime in which they operate can be very different from the true reward landscape.

F. Trends with advantages

F.1. Kurtosis for returns, value estimate and advantages grouped with sign

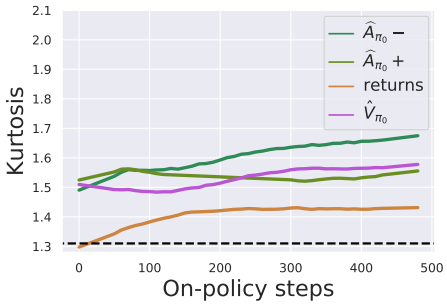


Figure 8. Heavy-tailedness in advantages grouped by their sign, rewards and value estimates. Clearly, as the training progresses the negative advantages become heavy-tailed. For returns, we observe an initial slight increase in the heavy-tailedness which quickly plateaus to a small magnitude of heavytailedness. The heavytailedness in the value estimates and positive advantages remain almost constant throughout the training.

F.2. Heavy-tailedness in A2C and PPO in onpolicy iterations

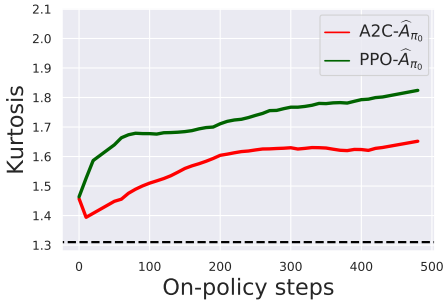


Figure 9. Heavy-tailedness in advantages for A2C and PPO during on-policy iterates. Clearly, as the training progresses heavy-tailedness in PPO advantages increases rapidly when compared with A2C advantages. The observed behavior arises to the off-policy training of the agent in PPO. This explains why we observe heightened heavy-tailedness in PPO during onpolicy iterations in Fig 1(a).

F.3. Histograms of advantages on HalfCheetah over training iterations

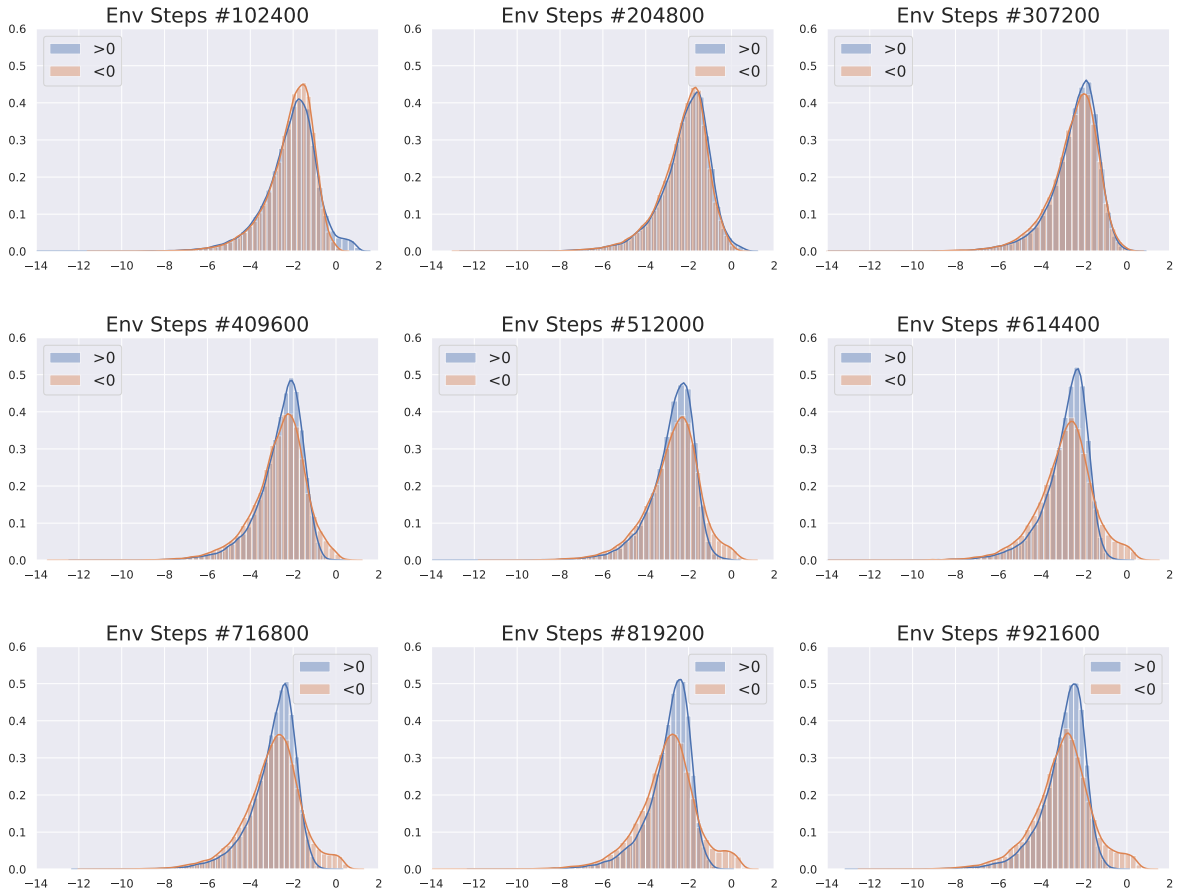
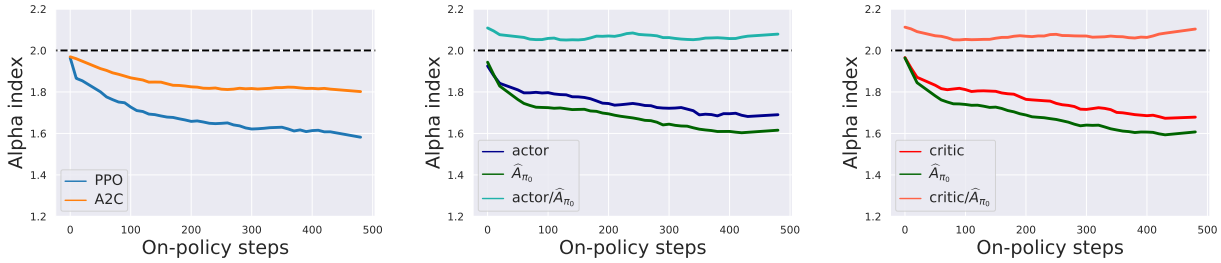


Figure 10. Distribution of $\log(|A_{\pi_\theta}|)$ over training grouped by sign of $\log(|A_{\pi_\theta}|)$ for HalfCheetah-v2 . To elaborate, we collect the advantages and separately plot the grouped advantages with their sign, i.e., we draw histograms separately for negative and positive advantages. As training proceeds, we clearly observe the increasing heavy-tailed behavior in negative advatanges as captured by the higher fraction of $\log(|A_{\pi_\theta}|)$ with large magnitude. Moreover, the histograms for positive advantages (which resembel Gaussain pdf) stay almost the same throughout training. This highlights the particular heavy-tailed (outlier-rich) nature of negative advantages corroborating our experiments with kurtosis and tail-index estimators.

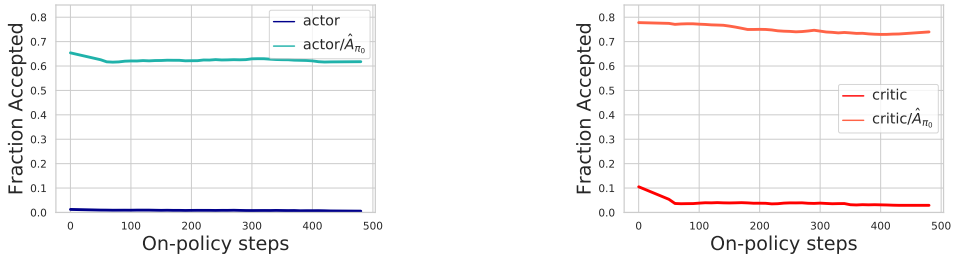
G. Analysis with other estimators

G.1. On-policy gradient analysis



(a) (b) (c)

Figure 11. **Heavy-tailedness in PPO during on-policy iterations**. All plots show mean alpha index aggregated over 8 MuJoCo environments. A decrease in alpha-index implies an increase in heavy-tailedness. (a) Alpha index vs on-policy iterations for A2C and PPO. Evidently, as training proceeds, the gradients become more heavy-tailed for both the methods. (b) Alpha index vs on-policy iterations for actor networks in PPO. (c) Alpha index vs on-policy iterations for critic networks in PPO. Both critic and actor gradients become more heavy-tailed on-policy steps as the agent is trained. Note that as the gradients become more heavy-tailed, we observe a corresponding increase of heavy-tailedness in the advantage estimates ($\hat{A}_{\pi_0}$) . However, "actor/$\hat{A}_{\pi_0}$" and "critic/$\hat{A}_{\pi_0}$" (i.e., actor or critic gradient norm divided by GAE estimates) remain light-tailed throughout the training.



(a) (b)

Figure 12. **Heavy-tailedness in PPO during on-policy iterations**. All plots show mean fraction of directions accepted by Anderon-Darling test over 8 MuJoCo environments. A higher accepted fraction indicates a Gaussian behavior. (b) Fraction accepted vs on-policy iterations for actor networks in PPO. (c) Fraction accepted vs on-policy iterations for critic networks in PPO. Both critic and actor gradients remain non-Gaussian as the agent is trained. However, "actor/$\hat{A}_{\pi_0}$" and "critic/$\hat{A}_{\pi_0}$" (i.e., actor or critic gradient norm divided by GAE estimates) have fairly high fraction of directions accepted, hinting their Gaussian nature.

G.2. Off-policy gradient analysis

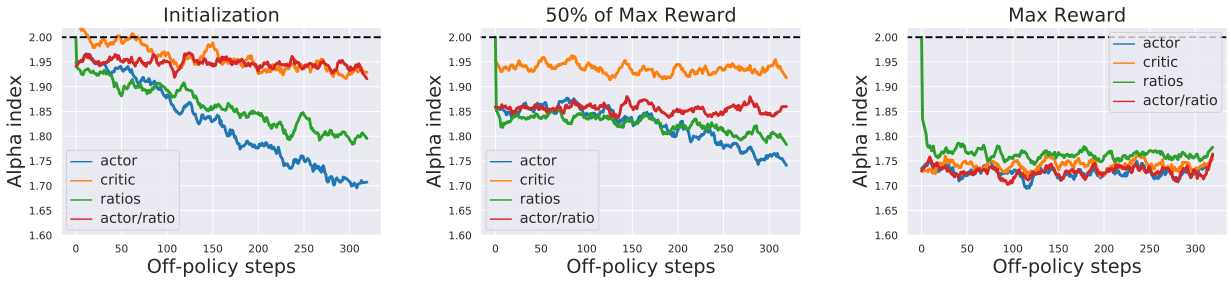


Figure 13. **Heavy-tailedness in PPO-NoClip during off-policy steps** at various stages of training iterations in MuJoCo environments. All plots show mean alpha index aggregated over 8 Mujoco environments. A decrease in alpha index implies an increase in heavy-tailedness. As off-policyness increases, the actor gradients get substantially heavy-tailed. This trend is corroborated by the increase of heavy-tailedness in ratios. Moreover, consistently we observe that the heavy-tailedness in "actor/ratios" stays constant. While initially during training, the heavy-tailedness in the ratio's increases substantially, during later stages the increase tapers off. The overall increase across training iterations is explained by the induced heavy-tailedness in the advantage estimates (cf. Sec. 3.1).

H. Hyperparameter settings and Rewards curves on individual enviornments

Hyperparameter	Values
Steps per PPO iteration	2048
Number of minibatches	32
PPO learning rate	0.0003
ROBUST-PPO-NOCLIP learning rate	0.00008
PPO-NOCLIP learning rate	0.00008
Discount factor γ	0.99
GAE parameter λ	0.95
Entropy loss coefficient	0.0
PPO value loss coefficient	2.0
ROBUST-PPO-NOCLIP value loss coefficient	2.0
PPO-NOCLIP value loss coefficient	2.0
Max global L2 gradient norm (only for PPO)	0.5
Clipping coefficient (only for PPO)	0.2
Policy epochs	10
Value epochs	10
GMOM number of blocks	8
GMOM Weiszfeld iterations	100

Table 1. Hyperparameter settings. Sweeps were run over learning rates { 0.000025, 0.00005, 0.000075, 0.00008, 0.00009 , 0.0001, 0.0003, 0.0004 } and value loss coefficient { 0.1, 0.5, 1.0, 2.0, 10.0} with 30 random seeds per learning rate.

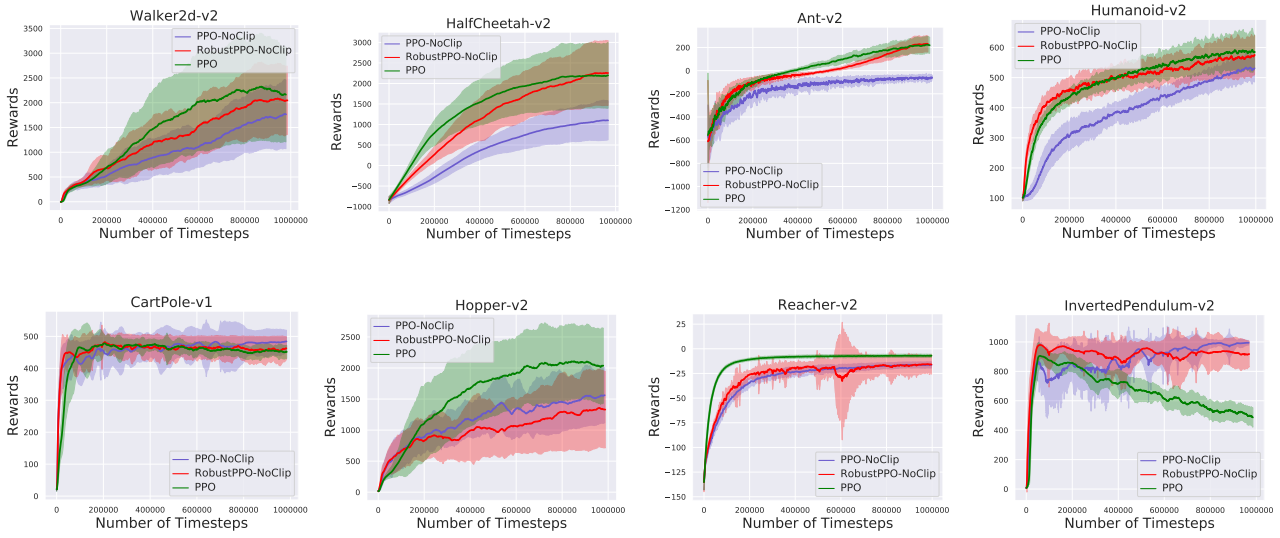


Figure 14. **Reward curves as training progresses in 8 different Mujoco Environments** aggregated across 30 random seeds and for hyperparameter setting tabulated in Table 1. The shaded region denotes the one standard deviation across seeds. We observe that except in Hopper-v2 environment, the mean reward with ROBUST-PPO-NOCLIP is significantly better than PPO-NOCLIP and close to that achieved by PPO with optimal hyperparameters. Aggregated results shown in Fig. 5.

I. Analysis on individual enviornments.

Overall, in the figures below, we show that the trends observed in aggregated plots in Section 3 with Kurtosis hold true on individual environments. While the degree of heavy-tailedness varies in different environments, the trend of increase in heavy-tailedness remains the same.

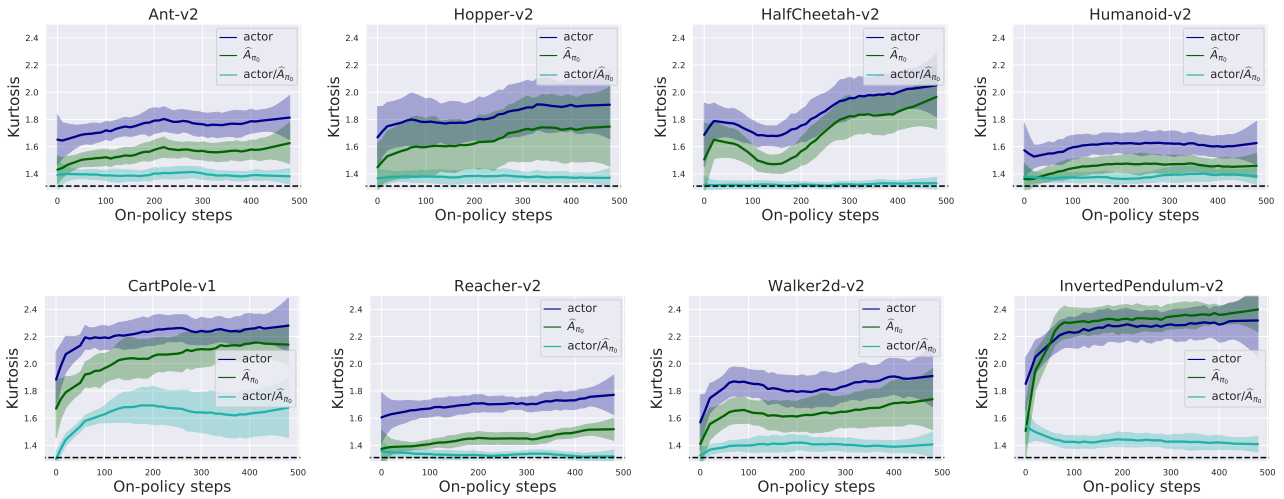


Figure 15. **Heavy-tailedness in actor gradients for PPO during on-policy steps** for 8 MuJoCo environments. All plots show mean and std of kurtosis aggregated over 30 random seeds. As the agent is trained, actor gradients become more heavy-tailed. Note that as the gradients become more heavy-tailed, we observe a corresponding increase of heavy-tailedness in the advantage estimates ($\hat{A}_{\pi_0}$). However, "actor/$\hat{A}_{\pi_0}$" (i.e., actor gradient norm divided by advantage) remain light-tailed throughout the training.

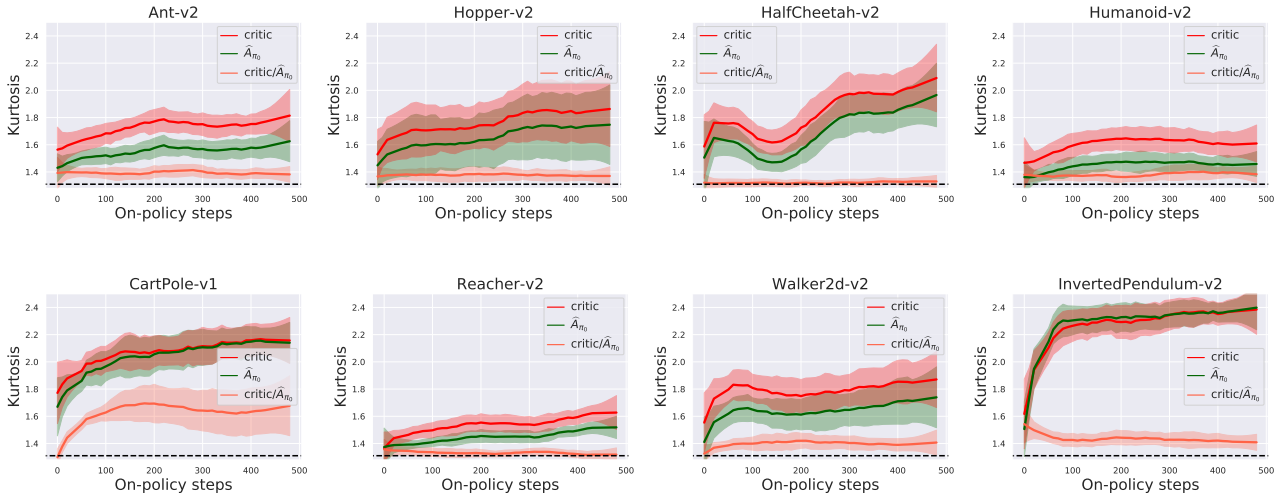


Figure 16. **Heavy-tailedness in critic gradients for PPO during on-policy steps** for 8 MuJoCo environments. All plots show mean and std of kurtosis aggregated over 30 random seeds. As the agent is trained, critic gradients become more heavy-tailed. Note that as the gradients become more heavy-tailed, we observe a corresponding increase of heavy-tailedness in the advantage estimates ($\hat{A}_{\pi_0}$). However, "critic/$\hat{A}_{\pi_0}$" (i.e., critic gradient norm divided by advantage) remain light-tailed throughout the training.

Figure 17. **Heavy-tailedness in PPO-NoClip during off-policy steps at Initialization** for 8 MuJoCo environments. All plots show mean and std of kurtosis aggregated over 30 random seeds. As off-policyness increases, the actor gradients get substantially heavy-tailed. This trend is corroborated by the increase of heavy-tailedness in ratios. Moreover, consistently we observe that the heavy-tailedness in "actor/ratios" stays constant. The trend in heavy-tailedness at later training iteration follow similar trends but the increase in heavy-tailedness tapers off. The overall increase across training iterations is explained by the induced heavy-tailedness in the advantage estimates (cf. Sec. 3.1).

J. How do heavy-tailed policy-gradients affect training?

J.1. Effect of heavy-tailedness in advantages

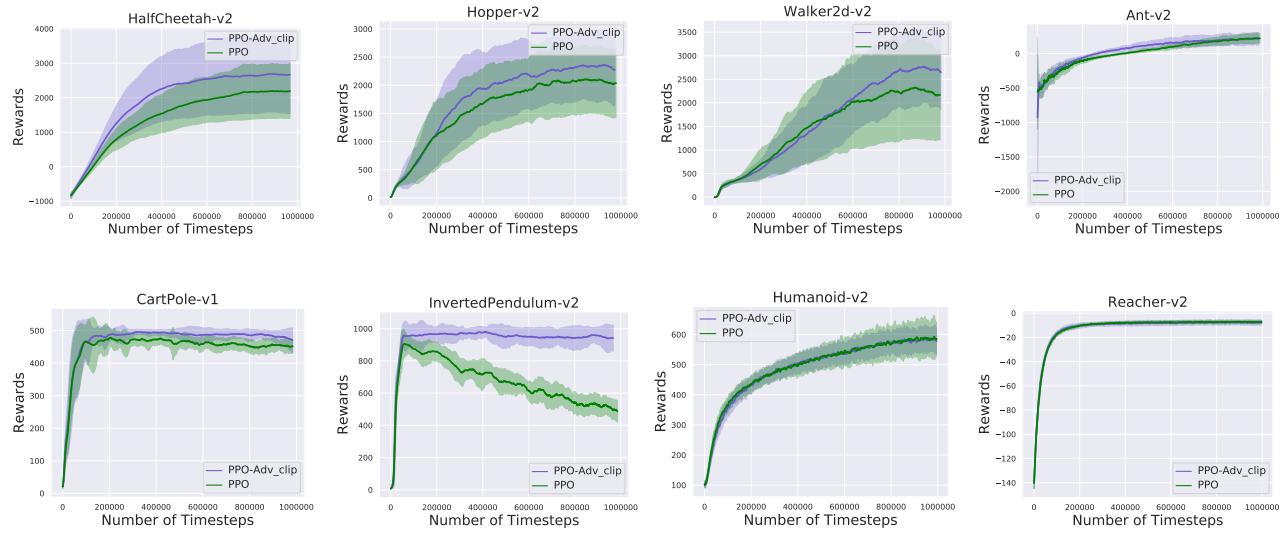


Figure 18. **Reward curves with advantage clipping in 8 different Mujoco Environments** aggregated across 30 random seeds. The shaded region denotes the one standard deviation across seeds. The clipping threshold is tuned per environment. We observe that by clipping *outlier* advantages, we substantially improve the mean rewards for 5 environments. While for the remaining three environments, we didn't observe any differences in the agent performance.

J.2. Effect of heavy-tailedness in likelihood-ratios

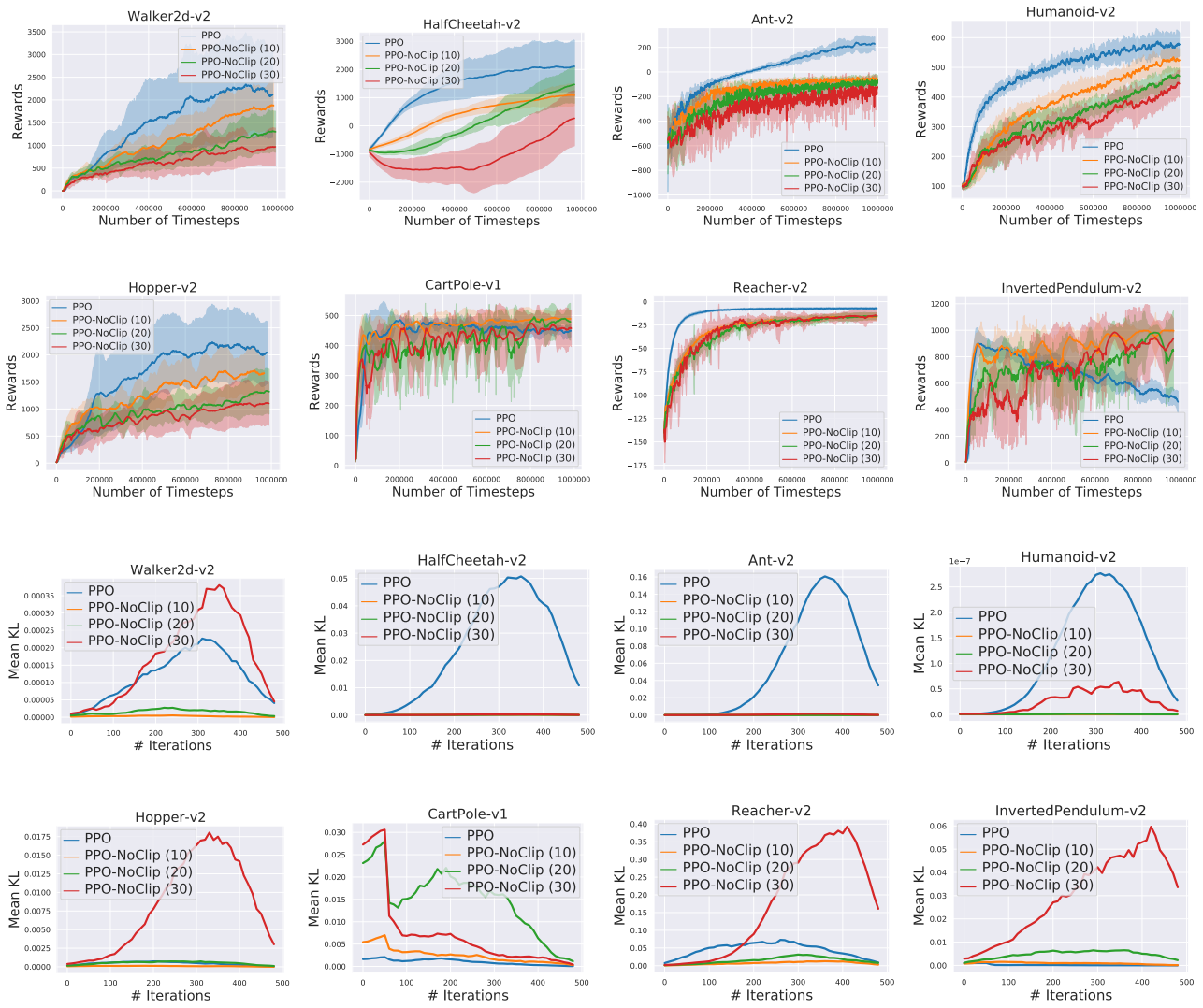


Figure 19. **(Top two rows)** Reward curves with the varying number of offline epochs in 8 different Mujoco Environments aggregated across 10 random seeds. Bracketed quantity in the legend denotes the number of offline epochs used for PPO-NoClip training. Clearly, as the number of offline epochs increases, the performance of the agent drops (consistent behavior across all environments). Furthermore, at 30 epochs the training also gets unstable. We also show the PPO performance curve for comparison. **(Bottom two rows)** KL divergence between current and previous policies with the optimal hyperparameters (parameters in Table 1) for PPO and PPO-NoClip with varying number of offline epochs. We measure mean empirical KL divergence between the policy obtained at the end of off-policy training and the sampling policy at the beginning of every training iteration. The quantities are measured over the state-action pairs collected in the training step. We observe that till 30 offline epochs PPO-NoClip maintains a trust-region with mean KL metric.

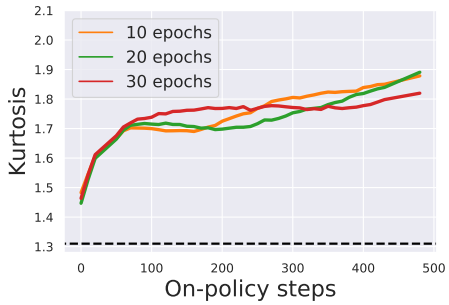


Figure 20. **Heavy-tailedness in PPO-NoClip advantages throughout the training as the degree of off-policyness is varied** in MuJoCo environments. Kurtosis is aggregated over 8 Mujoco environments. We plot kurtosis vs on-policy iterates. As the number of off-policy epochs increases, the heavy-tailedness in advantages remains the same showing an increase in the number of offline epochs has a minor effect on the induced heavy-tailedness in the advantage estimates.