ReLU to the Rescue: Improve Your On-Policy Actor-Critic with Positive Advantages

Andrew Jesson [1]   Chris Lu [2]   Gunshi Gupta [1]   Nicolas Beltran-Velez [3]   Angelos Filos [1]   Jakob N. Foerster [2]
Yarin Gal [1]

Abstract

This paper proposes a step toward approximate Bayesian inference in on-policy actor-critic deep reinforcement learning. It is implemented through three changes to the Asynchronous Advantage Actor-Critic (A3C) algorithm: (1) applying a ReLU function to advantage estimates, (2) spectral normalization of actor-critic weights, and (3) incorporating *dropout as a Bayesian approximation*. We prove under standard assumptions that restricting policy updates to positive advantages optimizes for value by maximizing a lower bound on the value function plus an additive term. We show that the additive term is bounded proportional to the Lipschitz constant of the value function, which offers theoretical grounding for spectral normalization of critic weights. Finally, our application of dropout corresponds to approximate Bayesian inference over both the actor and critic parameters, which enables *adaptive state-aware* exploration around the modes of the actor via Thompson sampling. We demonstrate significant improvements for median and interquartile mean metrics over A3C, PPO, SAC, and TD3 on the MuJoCo continuous control benchmark and improvement over PPO in the challenging Proc-Gen generalization benchmark.

1. Introduction

Deep Reinforcement Learning (DRL) finds approximate solutions to complex sequential decision-making problems in domains such as robotics (Ibarz et al., 2021), autonomous driving (Kiran et al., 2021), strategy games (Mnih et al.,

2015; Silver et al., 2017; Arulkumaran et al., 2019), and human-computer interaction (Ziegler et al., 2019). DRL algorithms achieve state-of-the-art performance on many challenging benchmarks (Young & Tian, 2019; Lange, 2022; Todorov et al., 2012; Brockman et al., 2016). However, their use in real-world applications depends on their capacity to execute tasks while making policy updates in the face of finite observations of a changing world. On-policy algorithms, such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) or Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016), update differentiable policies based on recent interactions with the environment. This recency bias and the capacity to actively sample informative observations make on-policy approaches compelling candidates for applications in dynamic real-world environments.

Exploration is a component of active sampling. On-policy actor-critic methods typically incorporate exploration through entropy regularization or by learning a homogeneous variance parameter for continuous action spaces (Schulman et al., 2015a; Mnih et al., 2016; Schulman et al., 2017). While effective, these exploration methods are *state-agnostic*, promoting exploration equally regardless of the novelty or familiarity of a given state. Alternatively, the paradigm of maximum entropy reinforcement learning (Ziebart et al., 2008; Haarnoja et al., 2018; Levine, 2018) promotes *state-aware* exploration through the inclusion of an actor entropy term in the optimization objective. However, this term is not *adaptive* in that it promotes higher actor entropy (more exploration) regardless of the number of state visits, and thus may be agnostic to the knowledge already gained about the system.

This work incorporates adaptive state-aware exploration into the on-policy actor-critic framework to improve the performance and efficiency of on-policy actor-critic algorithms. Approximate Bayesian inference over actor weights would satisfy this goal, but its implementation is not straightforward due to the policy-gradient objective for optimizing the actor, as we will show. Thus, we ask, "What is the minimal step we can take toward approximate Bayesian inference?" and answer with VSOP (standing for Variational [b]ayes, Spectral-normalized, On-Policy reinforce-

[1]OATML, Department of Computer Science, University of Oxford, Oxford, UK [2]FLAIR, Department of Computer Science, University of Oxford, Oxford, UK [3]Department of Computer Science, Columbia University, New York, NY. Correspondence to: Andrew Jesson <andrew.jesson@cs.ox.ac.uk>.

ment learning). VSOP consists of three simple modifications to the A3C algorithm: (1) applying a ReLU function to advantage estimates, (2) spectral normalization of actor-critic weights, and (3) incorporating *dropout as a Bayesian approximation* (Gal & Ghahramani, 2016). Under standard assumptions, we prove that restricting policy updates to positive advantages maximizes value by maximizing a lower bound on the value function plus an additive term. We show that the additive term is bounded proportional to the Lipschitz constant of the value function, grounding spectral normalization use as a Lipschitz constant regularizer. Finally, we show that approximate Bayesian inference over the actor and critic parameters enables adaptive state-aware exploration via Thompson sampling.

Through our thorough empirical assessments on the Gymnasium and Brax MuJoCo benchmarks for continuous control (Brockman et al., 2016; Freeman et al., 2021), we show that VSOP can significantly outperform existing DRL algorithms such as A3C, PPO, SAC, and TD3 for median and interquartile mean (IQM) metrics (Agarwal et al., 2021). We further show through ablation studies that the union of our proposed improvements contributes significantly to increased performance and efficiency. Finally, we show that VSOP significantly outperforms PPO on the challenging ProcGen generalization benchmark, demonstrating improved performance when deployed under distribution shift.

2. Background

Notation. We consider a discounted, T-horizon Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the state transition probability, r is the immediate reward upon transitioning from state $\mathbf{s}$ to state $\mathbf{s}'$ under action $\mathbf{a}$, and $\gamma \in [0, 1]$ is the discount factor. MDPs provide a framework for modeling sequential decision-making problems, where an agent interacts with an environment over discrete time steps to achieve a goal (Puterman, 2014). Following the notation of Sutton & Barto (2018), we define states at time $t \in \mathrm{T}$ by the d-dimensional, real-valued, random variable, $\mathbf{S}_t : \Omega \to \mathcal{S} \subseteq \mathbb{R}^d$, with observable instances $\mathbf{s}_t = \mathbf{S}_t(\omega_t) : \forall \omega_t \in \Omega$. We define actions by the m-dimensional random variable $\mathbf{A}_t : \Omega \to \mathcal{A}$, with observable instances, $\mathbf{a}_t = \mathbf{A}_t(\omega_t) : \forall \omega_t \in \Omega$. Rewards are defined by the continuous-valued random variable, $\mathrm{R}_t : \Omega \to \mathcal{R} \subseteq \mathbb{R}$, with observable instances, $\mathrm{r}_t = \mathrm{R}_t(\omega_t) : \forall \omega_t \in \Omega$. Let the random variable, $\mathrm{G}_t := \sum_{k=t+1}^{\mathrm{T}} \gamma^{k-1-t} \mathrm{R}_k$, denote the discounted return. We use the standard definitions for the conditional action distribution/density (policy), $\pi(\mathbf{a} \mid \mathbf{s})$, the state value function under the policy, $v_\pi(\mathbf{s}) := \mathbb{E}_\pi [\mathrm{G}_t \mid \mathbf{S}_t = \mathbf{s}]$, and state-action value function under the policy, $q_\pi(\mathbf{s}, \mathbf{a}) := \mathbb{E}_\pi [\mathrm{G}_t \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}]$.

On-Policy Actor-Critic Reinforcement Learning. On-

policy, Actor-critic approaches to reinforcement learning are called *policy-gradient* methods. They directly optimize a policy function, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$, differentiable with respect to parameters, $\boldsymbol{\theta}$, to maximize the expected discounted return under the policy, $v_\pi(\mathbf{s})$. On-policy approaches differ from off-policy approaches in that they only use recent samples from the current policy to achieve this objective. Actor-critic methods differ from other policy-gradient methods because they fit an approximate value function (critic), $v(\mathbf{s}, \mathbf{w})$, to the data collected under the policy, in addition to optimizing the policy function (actor). The critic is typically used in actor optimization but not generally for decision-making.

Deep reinforcement learning implements the actor and critic using neural network architectures, where the function parameters correspond to network weights. We denote the parameters of the actor and critic networks as $\boldsymbol{\theta}$ and $\mathbf{w}$, respectively. The output likelihood of the actor makes distributional assumptions informed by characteristics of the action space, $\mathcal{A}$. A common choice for continuous action spaces is an independent multivariate normally distributed likelihood with homogeneous noise variance, $\pi(\mathbf{a}_t \mid \mathbf{s}_t, \boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{a} \mid \boldsymbol{\mu}(\mathbf{s}, \boldsymbol{\theta}), \mathbf{I}\boldsymbol{\sigma}^2(\boldsymbol{\theta}))$, where $\boldsymbol{\sigma}^2(\boldsymbol{\theta}) = (\sigma_1^2, \ldots, \sigma_m^2)$ is the vector of inferred action noise variances. For discrete action spaces, the likelihood is often a categorical distribution, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) \sim \mathrm{Categorical}(\mathbf{a} \mid \boldsymbol{\mu}(\mathbf{s}, \boldsymbol{\theta}))$. In both cases, the mean parameter of the likelihood, $\boldsymbol{\mu}(\mathbf{s}, \boldsymbol{\theta})$, is the m-dimensional, vector-valued output of a neural network architecture with parameters, $\boldsymbol{\theta}$. Critic networks are commonly fit using a mean squared error objective, which implies a univariate normally distributed output likelihood with unit variance, $\mathrm{G} \mid \mathbf{s}, \mathbf{w} \sim \mathcal{N}(\mathrm{g} \mid v(\mathbf{s}, \mathbf{w}), 1)$, where the mean parameter is the approximate value function, $v(\mathbf{s}, \mathbf{w})$, and is given by the scalar-valued output of any neural network architecture with parameters, $\mathbf{w}$.

The baseline on-policy, actor-critic policy gradient algorithm performs gradient ascent with respect to the "performance" function, $J(\boldsymbol{\theta}) := v_\pi(\mathbf{s}_0, \boldsymbol{\theta})$, where $v_\pi(\mathbf{s}_0, \boldsymbol{\theta})$ is the value function with respect to the parameters $\boldsymbol{\theta}$ (Williams, 1992). By the policy gradient theorem (Sutton et al., 1999), we have: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} v_\pi(\mathbf{s}_0) \propto \int_{\mathcal{S}} \rho(\mathbf{s}) \int_{\mathcal{A}} q_\pi(\mathbf{s}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) d\mathbf{a} \, d\mathbf{s}$. Sutton & Barto (2018) show that a generalization of this result includes a comparison of the state-action value function, $q_\pi(\mathbf{s}, \mathbf{a})$, to an arbitrary baseline that does not vary with the action, $\mathbf{a}$.

When the baseline is the state value function, $v_\pi(\mathbf{s})$, we have an objective in terms of the *advantage function* (Baird, 1993; Schulman et al., 2015b), $h_\pi(\mathbf{s}, \mathbf{a}) := q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi(\mathbf{s})$, namely: $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \int_{\mathcal{S}} \rho(\mathbf{s}) \int_{\mathcal{A}} h_\pi(\mathbf{s}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) d\mathbf{a} \, d\mathbf{s}$. This formulation in terms of *all actions* can be further simplified in terms of sampled actions and states as $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \mathbb{E}_\pi [h_\pi(\mathbf{S}_t, \mathbf{A}_t) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta})]$. We use $\mathbb{E}_\pi$ to denote an expectation over states $\mathbf{S}_t$ and actions $\mathbf{A}_t$

collected under the policy $\pi(\mathbf{a} \mid \mathbf{s})$.

In general, because neither the state-action, $q_\pi(\mathbf{s}, \mathbf{a})$, nor the state value, $v_\pi(\mathbf{s})$, functions are known, we need an estimator for the advantage function. For compactness, we will focus on the generalized advantage estimator (GAE) proposed by Schulman et al. (2015b): $h(\mathbf{s}_t, r_t, \mathbf{w}) = \sum_{k=t+1}^{T} (\gamma\lambda)^{k-1-t} \delta_{t-k+1}^{\mathbf{w}}$, where $0 < \lambda \le 1$, and $\delta_t^{\mathbf{w}} = r_t + \gamma v(\mathbf{s}_{t+1}; \mathbf{w}) - v(\mathbf{s}_t; \mathbf{w})$ is the temporal difference (TD) residual of the value function with discount, γ (Sutton & Barto, 2018). The GAE then yields a low-variance gradient estimator for the policy function: $\widehat{\nabla_\theta J}(\theta) := \mathbb{E}_\pi [h(\mathbf{S}_t, R_t, \mathbf{w}) \nabla_\theta \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \theta)]$. Finally, the actor and critic networks are generally optimized by using mini-batch stochastic gradient descent (Robbins & Monro, 1951) to fit the functions induced by the network weights to a batch of data collected under the current policy, $\mathcal{D}_\pi^b = \{\mathbf{s}_i, \mathbf{a}_i, r_i\}_{i=1}^{b}$. The parameter updates are given by,

$$\theta \leftarrow \theta - \eta \frac{1}{b} \sum_{i=1}^{b} h(\mathbf{s}_i, r_i, \mathbf{w}) \nabla_\theta \log \pi(\mathbf{a}_i \mid \mathbf{s}_i, \theta), \quad (1a)$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{1}{b} \sum_{i=1}^{b} \nabla_\mathbf{w} \log p(g(\mathbf{s}_i, r_i, \tilde{\mathbf{w}}) \mid \mathbf{s}_i, \mathbf{w}), \quad (1b)$$

where, η, is the learning rate, $g(\mathbf{s}_i, r_i, \tilde{\mathbf{w}}) = h(\mathbf{s}_i, r_i, \tilde{\mathbf{w}}) + v(\mathbf{s}_i, \tilde{\mathbf{w}})$, and $\tilde{\mathbf{w}}$ are previous parameters.

3. Methods

This work takes a top-down approach to state-aware exploration for on-policy actor-critic DRL. To employ principled exploration strategies, we would like to have approximate posteriors, $q(\Theta \mid \mathcal{D}_{n-1})$ and $q(\mathbf{W} \mid \mathcal{D}_{n-1})$, for the weights of the actor and critic given data, $\mathcal{D}_{n-1} = \{\mathbf{s}_i, \mathbf{a}_i, r_i\}_{i=1}^{|\mathcal{T}_{n-1}|}$, collected under the policy, $\pi(\mathbf{a} \mid \mathbf{s}, \Theta_{n-1})$, over a set of horizons, $\mathcal{T}_{n-1} = T_1^{n-1} \cup T_2^{n-1} \cup \dots$. However, fastidiously Bayesian, bottom-up approaches often yield strategies less effective than the state-of-the-art. Leaving debates on evaluation aside, we start from the A3C algorithm and ask, "what minimum changes can we make to get close to an approximate posterior?"

Starting with the critic, $v(\mathbf{s}, \mathbf{w})$, this task seems straightforward because we train the critic with mean squared error loss. Hence, we can use the implied likelihood, $\mathcal{N}(g \mid v(\mathbf{s}, \mathbf{w}), 1)$, and use the *dropout as a Bayesian approximation* (Gal & Ghahramani, 2016) framework to infer an approximate posterior density over critic weights, $q(\mathbf{w} \mid \widehat{\mathbf{w}}, p_d)$, where $\widehat{\mathbf{w}}$ is the variational parameter for the network weights, and p_d is the dropout rate. We outline the resulting optimization procedure in lines 16-17 of Algorithm 1 for a unit Normal prior over critic weights, $p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid 0, \mathbf{I})$.

The inference task is less straightforward for the

actor because the A3C objective, $\nabla_\theta J(\theta) \propto \mathbb{E}_\pi [h_\pi(\mathbf{S}_t, \mathbf{A}_t) \nabla_\theta \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \theta)]$, is not merely maximization of the log-likelihood. Instead, the log-likelihoods, $\log \pi(\mathbf{a} \mid \mathbf{s}, \theta)$, are scaled by the advantage function, $h_\pi(\mathbf{s}, \mathbf{a})$. The *dropout as Bayesian approximation* framework estimates the integral over the log evidence lower bound objective using Monte-Carlo integration. For a single sample from the approximate posterior density, $\theta \sim q(\theta \mid \widehat{\theta}, p_d)$, the integrand is of the general form:

$$\mathcal{L} = \sum_{i=1}^{|\mathcal{D}|} \log p(\cdot \mid \mathbf{s}_i, \theta) - \text{KL}(q(\theta) \| p(\theta)). \quad (2)$$

We make the same prior assumption, $p(\theta) = \mathcal{N}(\theta \mid 0, \mathbf{I})$, for the actor as for the critic, so we only need to focus on the first r.h.s., log-likelihood term. Here we introduce a normal-gamma distribution over the actions r.v., $\mathbf{A}_t$, and a random variable H_t:

$$p(\mathbf{A}_t, H_t \mid \mathbf{s}, \theta, \tau, \alpha, \beta)$$
$$:= \begin{cases} \mathbf{A}_t \mid H_t, \mathbf{s}, \theta, \tau \sim \mathcal{N}\left(\mathbf{a} \mid \mu(\mathbf{s}, \theta), (\tau H_t)^{-1}\right) \\ H_t \mid \alpha, \beta \sim \text{Gamma}(\alpha, \beta). \end{cases}$$

Given a dataset, $\mathcal{D} = \{\mathbf{A}_i, \mathbf{S}_i, H_i\}_{i=1}^{N}$, and differentiating the log-likelihood with respect to θ, we have:

$$\nabla_\theta \log p(\mathbf{A}_t, H_t \mid \mathbf{S}_t, \theta, \dots)$$
$$= \sum_{i=1}^{N} \nabla_\theta \log \left(\frac{\beta^\alpha \sqrt{\tau}}{\Gamma(\alpha)\sqrt{2\pi}} h_i^{\alpha-\frac{1}{2}} \exp\left(-\beta h_i\right) \right.$$
$$\left. \exp\left(-\frac{1}{2} h_i \tau (\mathbf{a}_i - \mu(\mathbf{s}_i, \theta))^2\right) \right) \quad (3)$$
$$= -\frac{1}{2} \sum_{i=1}^{N} h_i \nabla_\theta \tau (\mathbf{a}_i - \mu(\mathbf{s}_i, \theta))^2$$
$$= -\frac{1}{2} \sum_{i=1}^{N} h_i \nabla_\theta \log \pi(\mathbf{a}_t \mid \mathbf{s}_t, \theta).$$

Thus, the normal-gamma assumption allows us to recover the form of the A3C gradient estimator in Equation (1a) while enabling approximate Bayesian inference over the actor parameters. However, the advantage function, $h_\pi(\mathbf{s}, \mathbf{a})$, is not gamma distributed, as it can take on values in the range $(-\infty, 0]$, so we cannot plug it directly into Equation (3). This discrepancy motivates our second, though not strictly valid due to the introduction of zeros, minimal step of passing the advantages through a ReLU function, enabling approximate Bayesian inference over the parameters of the actor to obtain the approximate posterior of the actor parameters, $q(\theta \mid \widehat{\theta}, p_d)$.

Sampling a policy from this approximate posterior involves sampling a dropout mask and running a forward pass of the

Algorithm 1 VSOP for Dropout Bayesian Neural Networks

Input: initial state, $\mathbf{s}'$, environment, $p(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a})$, rollout buffer, $\mathcal{D}$, initial actor parameters, $\widehat{\boldsymbol{\theta}}$, initial critic parameters, $\widehat{\mathbf{w}}$, dopout rate, p_d, learning rate, η, minibatch size, b.

repeat
    $\mathcal{D} \leftarrow \emptyset$ ▷ reset rollout buffer
    repeat
        $\mathbf{s} \leftarrow \mathbf{s}'$ ▷ update current state
        $\boldsymbol{\theta} \sim q(\boldsymbol{\theta} \mid \widehat{\boldsymbol{\theta}}, p_d)$ **if** TS **else** $\boldsymbol{\theta} \leftarrow \widehat{\boldsymbol{\theta}}$ ▷ sample actor params if Thompson sampling (TS)
        $\mathbf{a} \sim \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$ ▷ sample action from policy
        $\mathbf{s}', \mathrm{r} \sim p(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a})$ ▷ sample next state and reward from environment
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}, \mathbf{a}, \mathrm{r})\}$ ▷ update rollout buffer
    until rollout ends
    $\mathbf{w}^* \leftarrow \widehat{\mathbf{w}}$ ▷ freeze critic weights for advantage estimates
    $\beta \leftarrow (1 - p_d)/(2|\mathcal{D}|)$ ▷ set parameter precision
    repeat
        $\{\mathbf{s}_i, \mathbf{a}_i, \mathrm{r}_i\}_{i=1}^b \sim \mathcal{D}$ ▷ sample minibatch from rollout buffer
        $\widetilde{\mathbf{w}} \sim q(\mathbf{w} \mid \mathbf{w}^*, p)$ **if** TS **else** $\widetilde{\mathbf{w}} \leftarrow \mathbf{w}^*$ ▷ sample advantage params if TS
        $\boldsymbol{\theta} \sim q(\boldsymbol{\theta} \mid \widehat{\boldsymbol{\theta}}, p_d)$ ▷ sample actor parameters
        $\widehat{\boldsymbol{\theta}} \leftarrow \widehat{\boldsymbol{\theta}} - \eta \frac{1}{b} \sum_{i=1}^b h^+(\mathbf{s}_i, \mathrm{r}_i, \widetilde{\mathbf{w}}) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_i \mid \mathbf{s}_i, \boldsymbol{\theta}) + 2\beta\boldsymbol{\theta}$ ▷ update actor
        $\mathbf{w} \sim q(\mathbf{w} \mid \widehat{\mathbf{w}}, p_d)$ ▷ sample critic parameters
        $\widehat{\mathbf{w}} \leftarrow \widehat{\mathbf{w}} - \eta \frac{1}{b} \sum_{i=1}^b \nabla_{\mathbf{w}} \log p(\mathrm{g}(\mathbf{s}_i, \mathrm{r}_i, \widetilde{\mathbf{w}}) \mid \mathbf{s}_i, \mathbf{w}) + 2\beta\mathbf{w}$ ▷ update critic
    until actor and critic update complete
until finished

---

network, yielding the policy, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$. Then, sampling an action is done by sampling an action from the sampled policy, $\mathbf{a} \sim \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$. We outline this Thompson sampling (Thompson, 1933) procedure in lines 5-6 of Algorithm 1.

We hypothesize that this is a better state-aware exploration method for two reasons. First, it is adaptive: for less frequently visited states the diversity of the sampled parameters of the policy will be greater, promoting more exploration. As a state is visited more often under actions that yield positive advantages, the diversity of samples will concentrate, promoting less exploration. Thus, we get more exploration for states where the actor has less experience of good actions and less exploration in states where the actor's decisions have led to good expected returns. Second, this exploration is done around the mode of the policy distribution, resulting in more conservative exploration.

We outline an optimization step of the actor in lines 14-15 of Algorithm 1, where $h^+(\mathbf{s}_i, \mathrm{r}_i, \widetilde{\mathbf{w}}) := \max\big(0, h(\mathbf{s}_i, \mathrm{r}_i, \widetilde{\mathbf{w}})\big)$. Note that the clipped advantages as an actor precision modifier have a very intuitive interpretation. When advantage estimates are low (no evidence of past good actions), the variance of the policy will be high, indicating that the actor should explore more. Conversely, when the advantage estimates are high (evidence of past good actions), the variance of the policy will be low, indicating that the actor should explore less. We provide further commentary in Appendix C.2.

What Function Does VSOP Optimize? Clipping the advantage estimates to be non-negative has been explored in many policy-gradient contexts (Srinivasan et al., 2018; Oh et al., 2018; Petersen et al., 2019; Ferret et al., 2020). Here, we examine how this augmentation affects the optimization procedure in the context of on-policy actor-critic RL and offer a theoretical hypothesis to ground both our method and the Regret Matching Policy Gradient (RMPG) method of Srinivasan et al. (2018).

Theorem 3.1. *Let,* $G_t := \sum_{k=t+1}^T \gamma^{k-1-t} R_k$, *denote the discounted return. Let* $q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi [G_t \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}]$, *denote the state-action value function, and* $v_\pi(\mathbf{s}) = \mathbb{E}_\pi [G_t \mid \mathbf{S}_t = \mathbf{s}]$, *denote the state value function, under policy* $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$. *Define the ReLU function as* $(x)^+ := \max(0, x)$. *Assume that rewards,* R_t, *are non-negative and the gradient of the policy,* $\nabla\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$, *is a conservative vector field. Then, performing gradient ascent with respect to,*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) :=$$
$$\mathbb{E}_\pi \left[\big(q_\pi(\mathbf{S}_t, \mathbf{A}_t) - v_\pi(\mathbf{S}_t)\big)^+ \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta}) \right],$$

maximizes a lower-bound, $v_\pi^*(\mathbf{s})$, *on the state value function,* $v_\pi(\mathbf{s})$, *plus an additive term:*

$$v_\pi^*(\mathbf{s}) \leq v_\pi(\mathbf{s}) + C_\pi(\mathbf{s}). \qquad (4)$$

Where, $C_\pi(\mathbf{s}) = \iint \big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\big)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{S}_t =$

$\mathbf{s}, \mathbf{A}_t = \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{S}_t = \mathbf{s})$, *is the expected, clipped differ-ence in the state value function,* $\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})$, *over all actions,* $\mathbf{a}$, *and next states,* $\mathbf{s}'$, *under the policy given state,* $\mathbf{s}$. *Here, we use* $\int \ldots d\Pi(\mathbf{a} \mid \mathbf{s})$ *to denote* $\sum_\mathbf{a} \ldots \pi(\mathbf{a} \mid \mathbf{s})$ *for discrete action spaces and* $\int \ldots \pi(\mathbf{a} \mid \mathbf{s})d\mathbf{a}$ *for contin-uous action spaces. Similarly, we use* $\int \ldots d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ *to denote* $\sum_{\mathbf{s}'} \ldots p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ *for discrete state spaces and* $\int \ldots p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})d\mathbf{s}'$ *for continuous state spaces. We provide the proof in Appendix B.1.*

Bounding $C_\pi(\mathbf{s})$**.** For a K_π-Lipschitz value function and $\gamma = 1$, the additive term is bounded proportional to the expected absolute difference between states:

$$
\begin{aligned}
C_\pi(\mathbf{s}) &= \iint \left(v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) \right)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}) \\
&\leq = \frac{1}{2} \iint \left| v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) \right| d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}) \\
&\leq \frac{1}{2} \iint K_\pi \left| \left| \mathbf{s}' - \mathbf{s} \right| \right| d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}),
\end{aligned}
$$

where the second line follows from $Lemma\ B.4$. This interpretation motivates using spectral normalization (Miy-ato et al., 2018) of the value function estimator weights, $v(\mathbf{s}, \mathbf{w})$, which regulates the Lipschitz constant, K_π, of the estimator and can improve off-policy DRL performance (Bjorck et al., 2021; Gogianu et al., 2021). Moreover, this bound is not vacuous for the continuous (nor the dis-crete) action setting. Under weak assumptions, $f(\mathbf{a}, \mathbf{s}) := \int K_\pi \left| \left| \mathbf{s}' - \mathbf{s} \right| \right| d\mathbb{P}(\mathbf{s}' \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a})$, is finite for all $\mathbf{a}$. Therefore, $f^*(\mathbf{s}) = \max_\mathbf{a}(\int f(\mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{S}_t = \mathbf{s}))$, ex-ists and is finite, and $C_\pi(\mathbf{s}) \leq \frac{1}{2}f^*(\mathbf{s})$. We provide further commentary concerning the constant $C_\pi(\mathbf{s})$ and Lipschitz continuity in Appendix C.1.

We term this method VSOP for Variational [b]ayes, Spectral-normalized, On-Policy reinforcement learning. Algorithm 1 details VSOP for dropout BNNs.

4. Related Works

VSOP is an on-policy RL algorithm. Table 1 in Appendix A compares the gradient of the performance function, $\nabla J(\boldsymbol{\theta})$, for VSOP with those for relevant on-policy algorithms.

Maximum Entropy RL. Levine (2018) establishes the the-oretical connection between maximum entropy RL and pa-rameter agnostic probabilistic inference of policies over optimal trajectories. Further, in Section 4.1 of that work, he shows how approximate inference in maximum entropy pol-icy gradients setting corresponds precisely to the gradient estimator of the advantage actor-critic algorithm (A2C or A3C). It is clear from Theorem 3.1 that our gradient estima-tor is distinct from the advantage actor-critic estimator and that we optimize a fundamentally different objective than the maximum entropy objective. Therefore, we maintain

that our step towards approximate inference over policy pa-rameters represents a distinct alternative to the maximum entropy paradigm.

Mirror Learning. *Proximal Policy Optimization (PPO)* (Schulman et al., 2017), improves upon the baseline pol-icy gradient method by constraining the maximum size of policy updates. PPO employs a clipped surrogate objective function to limit the size of policy updates. PPO simplifies the optimization procedure compared to TRPO (Schulman et al., 2015a), making it more computationally efficient and easier to implement. While PPO constrains policy updates based on the ratio between the new and old policies, VSOP constrains policy updates according to the sign of the esti-mated advantage function. As such, PPO is an instance of the *mirror learning* framework (Kuba et al., 2022), whereas VSOP does not inherit the same theoretical guarantees. Lu et al. (2022) explores the Mirror Learning space by meta-learning a "drift" function. They term their immediate result Learned Policy Optimization (LPO). Through its analysis, they arrive at *Discovered Policy Optimisation (DPO)*, a novel, closed-form RL algorithm.

Regret Matching Policy Gradient (RMPG). Srinivasan et al. (2018) present a method inspired by their regret policy gradient (RPG) objective, which maximizes a lower-bound on the advantages: $(h(\mathbf{s}, \mathbf{a}))^+ \leq h(\mathbf{s}, \mathbf{a})$. RPG directly optimizes the policy for an estimator of the advantage lower-bound, denoted as $\nabla_{\boldsymbol{\theta}} J^{\mathrm{RPG}}(\boldsymbol{\theta})$. RMPG, being inspired by RPG, has a different objective, $\nabla_{\boldsymbol{\theta}} J^{\mathrm{RMPG}}(\boldsymbol{\theta})$. In both cases, $q(\mathbf{s}, \mathbf{a}, \mathbf{w})$ is a parametric estimator of the state-action value function, $q_\pi(\mathbf{s}, \mathbf{a})$. RMPG has demonstrated improved sam-ple efficiency and stability in learning compared to standard policy gradient methods. VSOP is closely related to RMPG; however, we provide the missing theoretical foundations to ground RMPG (Appendix B.1), extend RMPG from the *all actions* formulation making it more suitable for continuous control (Appendix B.2), and employ the GAE rather than the state-action value function estimator, $q(\mathbf{s}, \mathbf{a}, \mathbf{w})$.

Thompson Sampling in Deep Reinforcement Learning. Thompson sampling has been extensively explored in con-ventional and Deep Q-Learning (Strens, 2000; Wang et al., 2005; Osband et al., 2016; Moerland et al., 2017; Aziz-zadenesheli et al., 2018) to improve exploration and sample efficiency. Clements et al. (2019) and Nikolov et al. (2018) propose similar sampling-based exploration strategies for Deep Q-Learning. Jiang et al. (2023) propose a Thompson sampling strategy based on an ensemble of quantile estima-tors of the state-action value distribution. In the context of *policy gradient* methods, related Upper Confidence Bound (UCB) (Ciosek et al., 2019) and Hamiltonian Monte-Carlo (HMC) (Xu & Fekri, 2022) approaches are proposed for off-policy Soft Actor-Critic (SAC) (Haarnoja et al., 2018), and Henaff et al. (2022) proposes an elliptical episodic re-
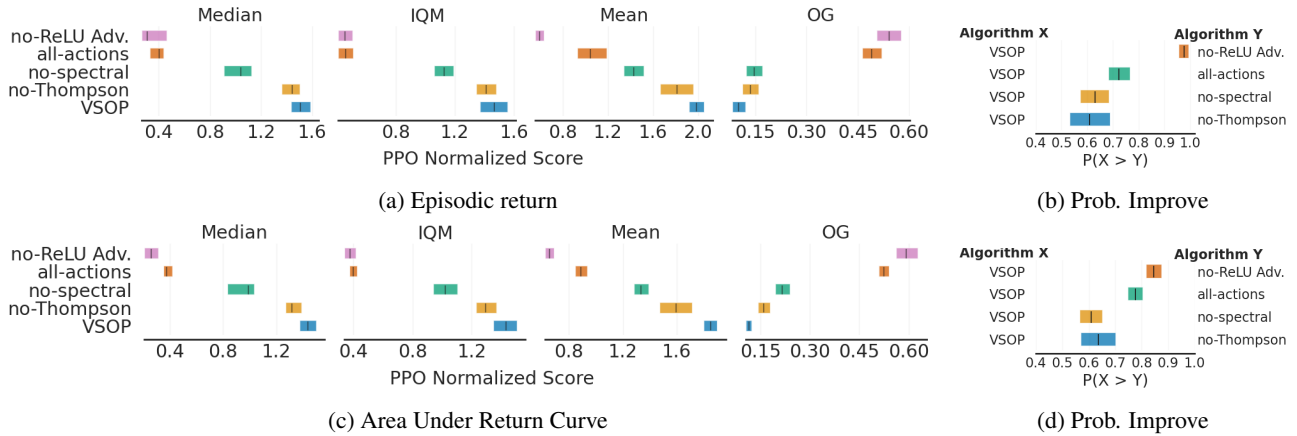
Figure 1: MuJoCo. Ablating the effect of the proposed mechanisms. Here, we compare VSOP to VSOP without spectral normalization (no-spectral), VSOP without Thompson sampling (no-Thompson), VSOP without advantage clipping (no-ReLU Adv.), and VSOP using all-actions policy optimization (all actions). We see that no single mechanism contributes greater than the sum of all changes, lending credence to the validity of our theory. The overall performance (a-b) and sample efficiency (c-d) metrics illustrate this result. Metrics are computed wrt to the average episodic return of the last 100 episodes and the area under the episodic return curve over ten random seeds

ward for general use. Hausknecht & Wagener (2022) and Igl et al. (2019) use fixed dropout masks to sample policies and actions but stopped short of formalizing this as Thompson sampling. Our work formalizes and shows the benefit of Thompson sampling for on-policy actor-critic methods.

5. Experiments

We evaluate VSOP against on-policy RL methods across various domains, including continuous and discrete action spaces and diverse dimensionalities in both the action and observation spaces. In Section 5.1, we evaluate VSOP on continuous control tasks using the Gymnasium (Brockman et al., 2016) and Gymnax (Lange, 2022) implementations of MuJoCo (Todorov et al., 2012). In Section 5.2, we assess the capacity of VSOP to learn policies that generalize to unseen environments at test time using the ProcGen benchmark (Cobbe et al., 2020). We use the rliable package to evaluate robust normalized median (Median), interquartile mean (IQM), mean (Mean), optimality gap (OG), and probability of improvement (Prob. Improve) metrics (Agarwal et al., 2021). We provide additional results in Appendix E and make code available at https://github.com/anndvision/vsop.

5.1. MuJoCo

For this evaluation, we build off of Huang et al. (2022)'s CleanRL package which provides reproducible, user-friendly implementations of state-of-the-art reinforcement learning algorithms using PyTorch (Paszke et al., 2019), Gymnasium (Brockman et al., 2016; Todorov et al., 2012), and Weights & (Biases, 2018). We give full implementation

details in Appendix D.1.

Ablation of Mechanisms. First, we investigate the influence of our four proposed mechanisms on the performance of VSOP. For reference, the mechanisms are positive advantages, single-action setting, spectral normalization, and Thompson sampling. To ablate each mechanism, we compare VSOP to four variants: VSOP without advantage clipping (no-ReLU Adv.), VSOP in the all-actions setting (all-actions), VSOP without spectral normalization (no-spectral), and VSOP without Thompson sampling (no-Thompson). We hyperparameter tune each variant by the same procedure used for VSOP (see Table 2 for details). Figure 1 summarizes these results, and we see clearly that no single mechanism contributes greater than the sum of all changes, lending credence to our theoretical analysis. We see that positive advantages and operating in the single-action regime impact performance on MuJoCo significantly. Spectral normalization and Thompson sampling also influence performance on MuJoCo positively, especially in high-dimensional action and observation space settings such as Humanoid, Humanoid Stand-Up, and Ant, as shown in Figure 2. The performance gains for spectral normalization align with results given by Bjorck et al. (2021) and Gogianu et al. (2021) for DDPG (Lillicrap et al., 2015), DRQ (Kostrikov et al., 2020), Dreamer (Hafner et al., 2019), DQN (Mnih et al., 2015; Wang et al., 2016) and C51 (Bellemare et al., 2017).

Comparison to Baselines. Next, we compare VSOP to baseline implementations: PPO, A3C, SAC, and TD3. We use the CleanRL (Huang et al., 2022) implementation of PPO, SAC, and TD3; the StableBaselines3 (Raffin et al., 2021) hyper-parameter settings for A3C. We also include

(a) Ant　　(b) HalfCheetah　　(c) Hopper　　(d) Humanoid　　(e) HumanoidStandup

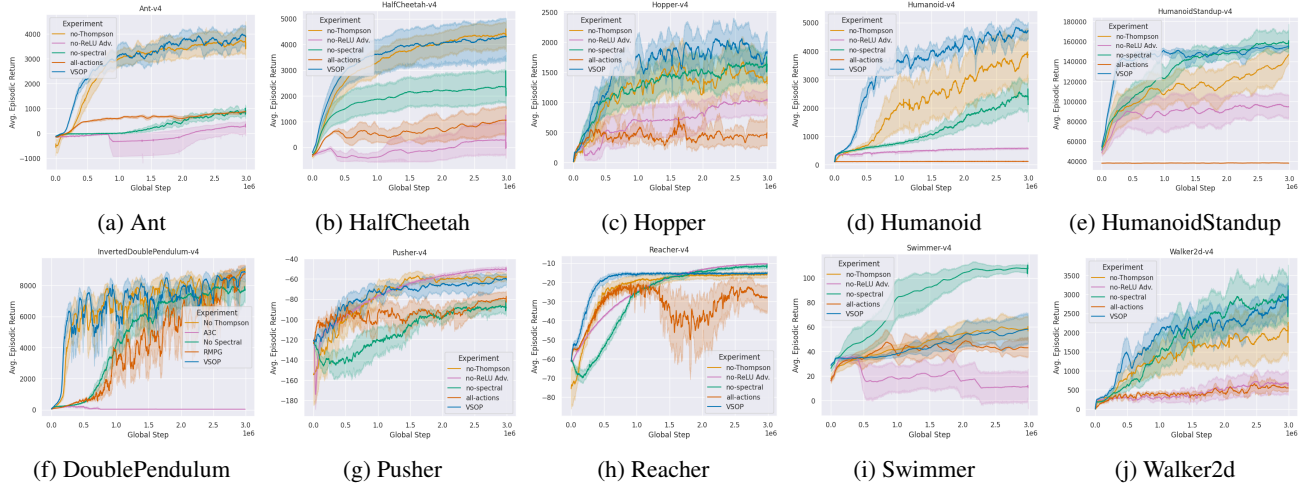(f) DoublePendulum　　(g) Pusher　　(h) Reacher　　(i) Swimmer　　(j) Walker2d

Figure 2: Comparing the effect of VSOP mechanisms on Mujoco continuous control performance. Using the single action framework and updating the policy only on positive advantage estimates have the largest effects, followed by spectral normalization, and finally Thompson sampling. Blue lines (VSOP) show the optimized proposed method. Orange lines (no-Thompson) show VSOP without Thompson sampling. Green lines (no-Spectral) show VSOP without spectral normalization. Pink lines (all actions) show VSOP with "all actions". Red lines (no ReLU Adv.) show VSOP without restricting policy updates to positive advantages.

comparisons to RMPG (adapted for continuous control) and VSPPO (PPO with spectral normalization, and Thompson sampling via dropout). We tune RMPG and VSPPO using the same Bayesian optimization (Snoek et al., 2012) protocol as VSOP. Figure 3 summarizes our results, where we see that VSOP shows significant improvement over each baseline concerning each metric, except for the SAC and TD3's mean scores. See Figure 9 in Appendix E.1 for training curves of these results.
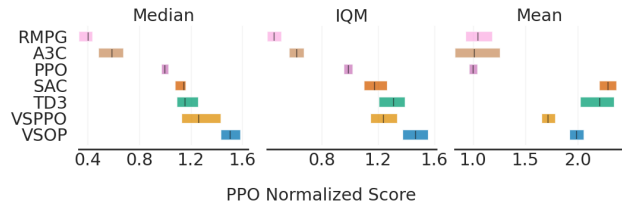

Figure 3: MuJoCo. Comparison to baselines. We see that VSOP (blue) shows significant improvement over each baseline concerning the Median and IQM metrics. VSOP only trails SAC and TD3 for the mean metric. Metrics are computed wrt to the average episodic return of the last 100 episodes over 10 random seeds

Effect of Asynchronous Parallelization. Following Lu et al. (2022), we build on PureJaxRL to evaluate VSOP on the Brax implementation of MuJoCo in a massively parallel setting. Where in the above experiments we set the number of asynchronous threads to 1 and the number of steps per rollout to 2048, here we set the number of asynchronous

threads to 2048 and the number of steps to 10. We see in Figure 4 that while VSOP still outperforms A3C significantly, it trails PPO. Full training curves are shown in Figure 13 of Appendix E.3.
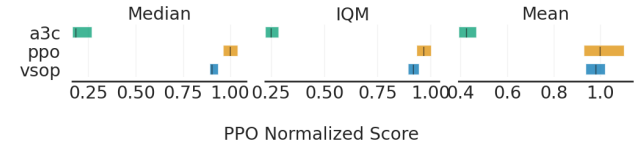

Figure 4: MuJoCo. Comparison to on-policy baselines with extreme parallelization. We compare VSOP to on-policy baselines on MuJoCo with 2048 threads and 10 steps per rollout. Metrics are computed wrt to the average episodic return of the last 100 episodes over 20 random seeds

Interestingly, hyper-parameter tuning showed spectral normalization to be detrimental to the performance of VSOP in this massively parallel setting. We investigate the effect of parallelization on VSOP effectiveness and efficiency in Figure 5. Here we set the rollout size to 2048 environment interactions and sweep the number of threads and number of steps. For each configuration, we do a hyper-parameter sweep in MuJoCo Brax using the reacher, hopper, and humanoid environments over 1 million environment interactions. We then evaluate 10 MuJoCo environments over 3 million environment interactions. The blue bars show metrics for VSOP with spectral normalization. We see that VSOP is most effective and efficient with spectral normalization with a low thread count and that for a fixed rollout

(a) Episodic return
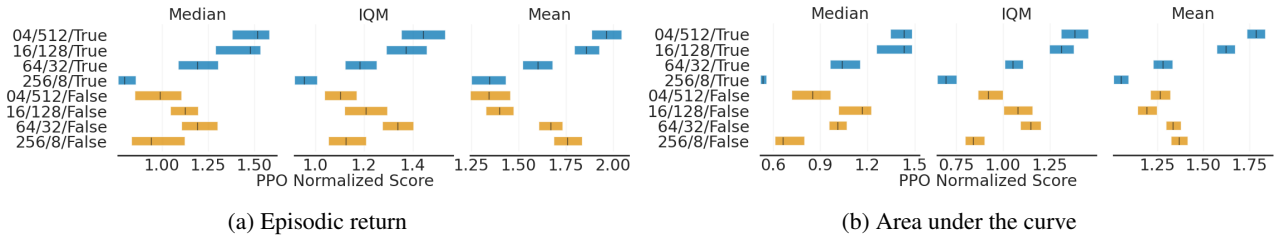
(b) Area under the curve

Figure 5: MuJoCo: effect of parallelization on VSOP. Naming convention: #threads/#steps/spectral norm. We see that VSOP is most effective 5a and most efficient 5b in lower thread settings for a fixed rollout size of 2048 steps when using spectral normalization. Metrics are computed wrt to the average episodic return or area under the curve for the last 100 episodes over 5 random seeds
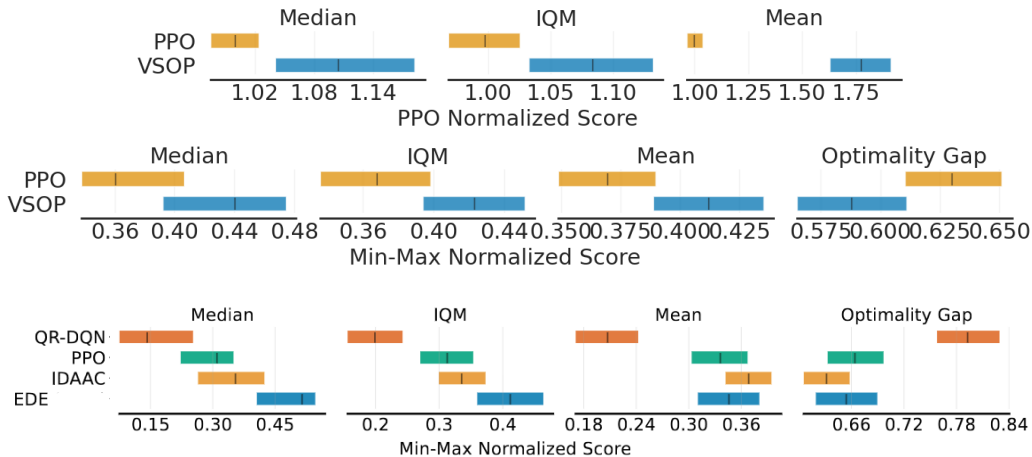

Figure 6: ProcGen comparison to PPO. In the top pane, we see significant improvement over PPO concerning all metrics for the PPO normalized scores. In the middle pane, we see significant improvement over PPO in terms of the IQM, mean, and optimality gap metrics for the Min-Max normalized scores. In the bottom pane, we include results reported by Jiang et al. (2023). It appears as though we improved over EDE with respect to the IQM, mean, and optimality gap metrics. Metrics are computed wrt to the average episodic return of the last 100 episodes over 5 random seeds

size, these measures fall with increasing parallelization. For VSOP without spectral normalization, the trend is less clear but appears to be generally the opposite for a fixed rollout size. This indicates that spectral normalization will be beneficial in applications where it is not feasible to run many parallel agents.

5.2. ProcGen

In lieu of finding a suitable benchmark for continuous control, we assess the capacity of VSOP to generalize to unseen environments using ProcGen (Cobbe et al., 2020). ProcGen is a set of 16 environments where game levels are procedurally generated, creating a virtually unlimited set of unique levels. We follow the "easy" generalization protocol where, for a given environment, models are trained on 200 levels for 25 million time steps and evaluated on the full distribution of environments. We use the same architecture as PPO in the CleanRL library, and do a Bayesian optimiza-

tion hyper-parameter search using the bossfight environment. We search over the learning rate, GAE λ, number of minibatches per epoch, number of epochs per rollout, the dropout rate, and the entropy regularization coefficient. Full implementation details are given in Appendix D.3. Figure 6 summarizes our results. We see broad significant improvement over PPO across both the PPO and Min-Max normalized metrics. Furthermore, we see improvement over EDE (Jiang et al., 2023) with respect to the IQM, mean, and optimality gap metrics. Figure Figure 7 shows training and test curves for the average episodic return against the number of environment interactions for each environment. These results present strong evidence for the suitability of VSOP for deployment in non-stationary environments and lend further evidence for the hypothesis of Jiang et al. (2023) that exploration plays a significant positive role in generalization.
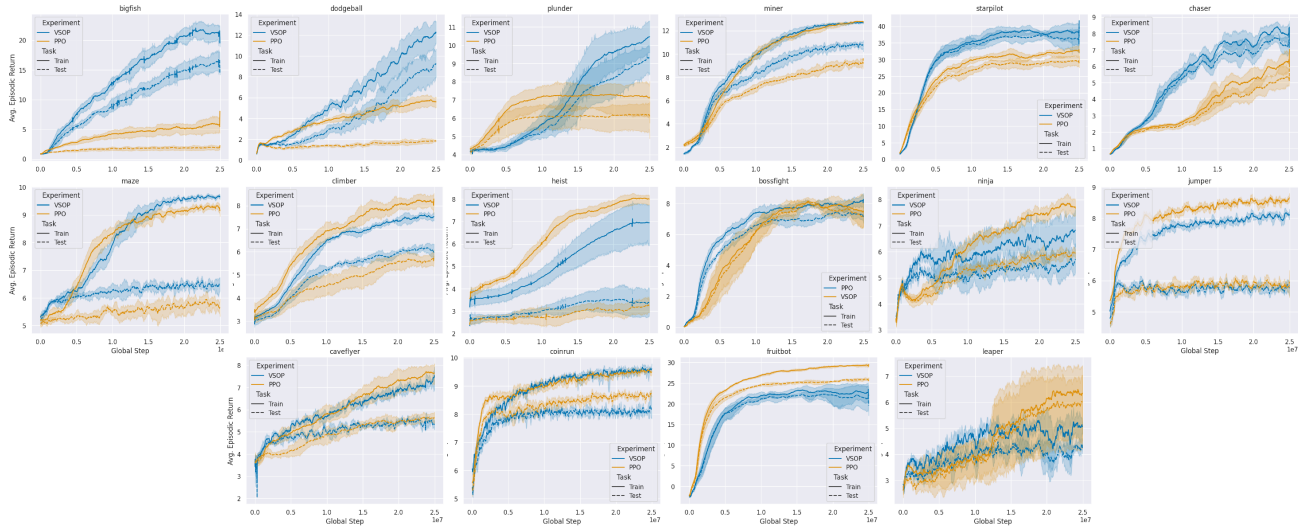
Figure 7: ProcGen training and test curves. We see significant improvement in test set performance on 8 environments, statistical equivalence on 5 environments, and VSOP trails PPO on just 3 environments.

6. Conclusion

This work represents a step towards principled approximate Bayesian inference in the on-policy actor-critic setting. Our method is realized through simple modifications to the A3C algorithm, optimizes a lower bound on value plus an additive term and integrates *adaptive state-aware* exploration via Thompson sampling. Our empirical evaluations across several diverse benchmarks confirm our approach's improved performance compared to existing on-policy algorithms.

Establishing convergence rates, especially compared to other algorithms (Shen et al., 2023), remains an important next step. Incorporating theory related to asynchronous settings (Shen et al., 2023) could improve our results, seeing as our most significant efficiency gains are in the single-threaded setting. Finally, since VSOP does not have significant computational overhead when compared to PPO, it would be interesting to evaluate whether our observed performance gains translate to contemporary large-scale settings where PPO is deployed, such as RLHF (Ziegler et al., 2019), online gaming (Berner et al., 2019), or robotics (Andrychowicz et al., 2020).

Acknowledgments

Impact Statement

Algorithmic decision-making is becoming increasingly present in many areas of our life. While this has the potential for benefit, it is also known to automate and perpetuate historical patterns that are often unjust and discriminatory (Buolamwini & Gebru, 2018; Noble, 2018; Benjamin, 2020; Birhane, 2021). We have proposed an RL algorithm and demonstrated empirically in simulated environments that it can achieve desired capabilities in a manner that requires fewer interactions with the environment when compared with existing algorithms. We believe that improving such algorithmic efficiency can help with harm reduction, but we acknowledge that technological solutions alone will not suffice.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Aghasadeghi, N. and Bretl, T. Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals. In *2011 IEEE/RSJ International Conference*

on Intelligent Robots and Systems, pp. 1561–1566. IEEE, 2011.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. What matters for on-policy deep actor-critic methods? a large-scale study. In *International conference on learning representations*, 2021.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Arulkumaran, K., Cully, A., and Togelius, J. Alphastar: An evolutionary computation perspective. In *Proceedings of the genetic and evolutionary computation conference companion*, pp. 314–315, 2019.

Azizzadenesheli, K., Brunskill, E., and Anandkumar, A. Efficient exploration through bayesian deep q-networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Baird, L. C. Advantage updating. Technical report, Technical Report WL-TR-93-1146, Wright-Patterson Air Force Base Ohio: Wright . . . , 1993.

Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

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

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. PMLR, 2017.

Benjamin, R. Race after technology: Abolitionist tools for the new jim code, 2020.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Biases, W. . Weights & biases. https://wandb.ai/site, 2018.

Birhane, A. Algorithmic injustice: a relational ethics approach. *Patterns*, 2(2):100205, 2021.

Bjorck, N., Gomes, C. P., and Weinberger, K. Q. Towards deeper deep reinforcement learning with spectral normalization. *Advances in Neural Information Processing Systems*, 34:8242–8255, 2021.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Buolamwini, J. and Gebru, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pp. 77–91. PMLR, 2018.

Ciosek, K., Vuong, Q., Loftin, R., and Hofmann, K. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019.

Clements, W. R., Van Delft, B., Robaglia, B.-M., Slaoui, R. B., and Toth, S. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Eimer, T., Lindauer, M., and Raileanu, R. Hyperparameters in reinforcement learning and how to tune them. *arXiv preprint arXiv:2306.01324*, 2023.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Implementation matters in deep policy gradients: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

Ferret, J., Pietquin, O., and Geist, M. Self-imitation advantage learning. *arXiv preprint arXiv:2012.11989*, 2020.

Filos, A., Vértes, E., Marinho, Z., Farquhar, G., Borsa, D., Friesen, A., Behbahani, F., Schaul, T., Barreto, A., and Osindero, S. Model-value inconsistency as a signal for epistemic uncertainty. In *International Conference on Machine Learning*, pp. 6474–6498. PMLR, 2022.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Gogianu, F., Berariu, T., Rosca, M. C., Clopath, C., Busoniu, L., and Pascanu, R. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *International Conference on Machine Learning*, pp. 3734–3744. PMLR, 2021.

Greenberg, I., Chow, Y., Ghavamzadeh, M., and Mannor, S. Efficient risk-averse reinforcement learning. *arXiv preprint arXiv:2205.05138*, 2022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Hausknecht, M. and Wagener, N. Consistent dropout for policy gradient reinforcement learning. *arXiv preprint arXiv:2202.11818*, 2022.

Henaff, M., Raileanu, R., Jiang, M., and Rocktäschel, T. Exploration via elliptical episodic bonuses. *Advances in Neural Information Processing Systems*, 35:37631–37646, 2022.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning. Lecture Slides, 2012. URL http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., and Levine, S. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., and Hofmann, K. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

Jiang, Y., Kolter, J. Z., and Raileanu, R. On the importance of exploration for generalization in reinforcement learning. *arXiv preprint arXiv:2306.05483*, 2023.

Kenton, Z., Filos, A., Evans, O., and Gal, Y. Generalizing from a few environments in safety-critical reinforcement learning. *arXiv preprint arXiv:1907.01475*, 2019.

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

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23 (6):4909–4926, 2021.

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Kuba, J. G., de Witt, C. S., and Foerster, J. Mirror learning: A unifying framework of policy optimisation. *arXiv preprint arXiv:2201.02373*, 2022.

Lange, R. T. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

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

Lu, C., Kuba, J., Letcher, A., Metz, L., Schroeder de Witt, C., and Foerster, J. Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468, 2022.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Moerland, T. M., Broekens, J., and Jonker, C. M. Efficient exploration with double uncertain value networks. *arXiv preprint arXiv:1711.10789*, 2017.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Nikolov, N., Kirschner, J., Berkenkamp, F., and Krause, A. Information-directed exploration for deep reinforcement learning. *arXiv preprint arXiv:1812.07544*, 2018.

Noble, S. U. Algorithms of oppression. In *Algorithms of oppression*. New York University Press, 2018.

Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Pytorch: An imperative style, high-performance deep learning library. https://pytorch.org, 2019.

Petersen, B. K., Landajuela, M., Mundhenk, T. N., Santiago, C. P., Kim, S. K., and Kim, J. T. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*, 2019.

Prashanth, L., Fu, M. C., et al. Risk-sensitive reinforcement learning via policy gradient search. *Foundations and Trends® in Machine Learning*, 15(5):537–693, 2022.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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

Shen, H., Zhang, K., Hong, M., and Chen, T. Towards understanding asynchronous advantage actor-critic: Convergence and linear speedup. *IEEE Transactions on Signal Processing*, 2023.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Srinivasan, S., Lanctot, M., Zambaldi, V., Pérolat, J., Tuyls, K., Munos, R., and Bowling, M. Actor-critic policy optimization in partially observable multiagent environments. *Advances in neural information processing systems*, 31, 2018.

Strens, M. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pp. 943–950, 2000.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Tamar, A., Glassner, Y., and Mannor, S. Optimizing the cvar via sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Tang, Y. C., Zhang, J., and Salakhutdinov, R. Worst cases policy gradients. In *Conference on Robot Learning*, pp. 1078–1093. PMLR, 2020.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. http://www.mujoco.org, 2012.

Wang, T., Lizotte, D., Bowling, M., and Schuurmans, D. Bayesian sparse sampling for on-line reward optimization. In *Proceedings of the 22nd international conference on Machine learning*, pp. 956–963, 2005.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

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

Willse, T. What is the inverse operation of a gradient? Mathematics Stack Exchange, 2019. URL https://math.stackexchange.com/q/3111825. URL:https://math.stackexchange.com/q/3111825 (version: 2019-02-13).

Xu, D. and Fekri, F. Improving actor-critic reinforcement learning via hamiltonian monte carlo method. *IEEE Transactions on Artificial Intelligence*, 2022.

Young, K. and Tian, T. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. Supplementary Related Works

Table 1: Comparison of performance functions for on-policy methods

Method	$\nabla J(\boldsymbol{\theta})$	
A3C	$\mathbb{E}_\pi \left[h_\pi(\mathbf{S}_t, \mathbf{A}_t) \nabla \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta}) \right];$	$h_\pi(\mathbf{S}_t, \mathbf{A}_t) = q_\pi(\mathbf{S}_t, \mathbf{A}_t) - v_\pi(\mathbf{S}_t)$
VSOP	$\mathbb{E}_\pi \left[h_\pi^+(\mathbf{S}_t, \mathbf{A}_t) \nabla \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta}) \right];$	$h_\pi^+(\mathbf{S}_t, \mathbf{A}_t) = \max\left(0, h_\pi(\mathbf{S}_t, \mathbf{A}_t)\right)$
RMPG	$\mathbb{E}_\pi \left[\int h_\pi^+(\mathbf{S}_t, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{S}_t, \boldsymbol{\theta}) \right]$	
TRPO	$\mathbb{E}_\pi \left[h_\pi(\mathbf{S}_t, \mathbf{A}_t) \nabla \rho(\mathbf{S}_t, \mathbf{A}_t, \boldsymbol{\theta}) \right];$	$\rho(\mathbf{S}_t, \mathbf{A}_t, \boldsymbol{\theta}) = \frac{\pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta})}{\pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta}_{\text{old}})}$
PPO	$\mathbb{E}_\pi \left[\min\left(h_\pi(\mathbf{S}_t, \mathbf{A}_t) \nabla \rho(\mathbf{S}_t, \mathbf{A}_t, \boldsymbol{\theta}), \text{clip}\left(h_\pi(\mathbf{S}_t, \mathbf{A}_t) \nabla \rho(\mathbf{S}_t, \mathbf{A}_t, \boldsymbol{\theta}), 1-\epsilon, 1+\epsilon \right) \right) \right]$	
DPO	$\mathbb{E}_\pi \left[\nabla \begin{cases} \left(h_\pi(\rho(\boldsymbol{\theta}) - 1) - a\tanh(h_\pi(\rho(\boldsymbol{\theta}) - 1)/a) \right)^+ & h_\pi(\mathbf{S}_t, \mathbf{A}_t) \geq 0 \\ \left(h_\pi \log(\rho(\boldsymbol{\theta})) - b\tanh(h_\pi \log(\rho(\boldsymbol{\theta})/b)) \right)^+ & h_\pi(\mathbf{S}_t, \mathbf{A}_t) < 0 \end{cases} \right]$	
CVaR	$\mathbb{E}_\pi \left[\left(\nu_\alpha - \mathrm{G}_t \right)^+ \nabla \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta}) \right];$	$\nu_\alpha := \alpha\text{-quantile of return, } \mathrm{G}_t$
RSPG	$\mathbb{E}_\pi \left[\left(\mathrm{G}_t - \nu_\alpha \right)^+ \nabla \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \boldsymbol{\theta}) \right];$	$\mathrm{G}_t := \sum_{k=t+1}^{\mathrm{T}} \gamma^{k-1-t} \mathrm{R}_k$
EPOpt	$\mathbb{E}_\pi \left[\mathbb{1}\left(\mathrm{G}_t \leq \nu_\alpha \right) \nabla J(\boldsymbol{\theta}, \mathbf{S}_t, \mathbf{A}_t) \right];$	$J(\boldsymbol{\theta}, \mathbf{S}_t, \mathbf{A}_t)$ on-policy perf. function

Off-policy Methods with Clipped Advantages. *Self Imitation Learning (SIL)* (Oh et al., 2018) is a hybrid method that uses clipped advantage estimates to improve the performance of on-policy algorithms such as PPO and A2C by learning from its successful off-policy trajectories. By leveraging experience replay, SIL encourages the agent to imitate its high-reward actions. *Self Imitation Advantage Learning (SIAL)* (Ferret et al., 2020) extends SIL to the off-policy domain. SIAL uses the clipped advantage function to weigh the importance of different actions during self-imitation, enabling the agent to focus on actions that yield higher long-term rewards. Importantly, even though SIL and SIAL only update policies when advantage estimates are positive, they differ from VSOP in that they are off-policy algorithms that learn from successful past trajectories and optimize different objectives based on max-entropy reinforcement learning (Aghasadeghi & Bretl, 2011; Haarnoja et al., 2018).

Mirror Learning. *Trust Region Policy Optimization (TRPO)* (Schulman et al., 2015a) is an on-policy, actor-critic method that improves upon the baseline policy gradient method by incorporating a constraint on the maximum size of policy updates. TRPO takes small steps toward improvement and limits the step size to ensure that the new policy does not deviate significantly from the old policy. TRPO achieves this by optimizing a surrogate objective function that approximates the expected reward under the new policy while imposing a constraint on the KL divergence between the new and old policies. TRPO is effective in various high-dimensional and continuous control tasks.

Risk Sensitive Reinforcement Learning. Instead of optimizing expected value, risk-sensitive RL methods optimize a risk measure. Tamar et al. (2015) propose the risk-averse *CVaR-PG* which seeks to minimize the Conditional Value at Risk (CVaR), $\Phi(\theta) := \mathbb{E}_\pi \left[\mathrm{G}_t \mid \mathrm{G}_t \leq \nu_\alpha \right]$, where ν_α is the α-quantile of the return, G_t, distribution under the policy, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$. Relatedly, Tang et al. (2020) have used the CVaR as a baseline function for standard policy updates. By focusing only on the worse case trajectories, CVaR-PG is susceptible to "blindness to success," thus Greenberg et al. (2022) propose a Cross-entropy Soft-Risk algorithm (CeSoR) to address this. Kenton et al. (2019) and Filos et al. (2022) also propose uncertainty aware, risk-averse methods. For model-based policy gradient methods, Rajeswaran et al. (2016) propose *Ensemble Policy Optimization (EPOpt)*, which incorporates restricting policy updates to be risk-averse based on the CVaR and uses ensembles to sample hypothesized models. In contrast to the above risk-averse methods, Petersen et al. (2019) present *Risk Seeking Policy Gradient (RSPG)* which focuses on maximizing best-case performance by only performing gradient updates when rewards exceed a specified quantile of the reward distribution. Prashanth et al. (2022) provide a comprehensive discussion on risk-sensitive RL.

B. Theoretical Results

B.1. Proof of Theorem 3.1

Theorem B.1. *Let,* $\mathrm{G}_t := \sum_{k=t+1}^{\mathrm{T}} \gamma^{k-1-t} \mathrm{R}_k$, *denote the discounted return. Let* $q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[\mathrm{G}_t \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a} \right]$, *denote the state-action value function, and* $v_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[\mathrm{G}_t \mid \mathbf{S}_t = \mathbf{s} \right]$, *denote the state value function, under policy*

$\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$. *Define the ReLU function as* $(x)^+ := \max(0, x)$. *Assume that rewards,* R_t, *are non-negative and the gradient of the policy,* $\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$, *is a conservative vector field. Then, performing gradient ascent with respect to,*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\left(q_\pi(\mathbf{S_t}, \mathbf{A_t}) - v_\pi(\mathbf{S_t}) \right)^+ \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A_t} \mid \mathbf{S_t}, \boldsymbol{\theta}) \right], \tag{6}$$

maximizes a lower-bound, $v_\pi^*(\mathbf{s})$, *on the state value function,* $v_\pi(\mathbf{s})$, *plus an additive term:*

$$v_\pi^*(\mathbf{s}) \le v_\pi(\mathbf{s}) + C_\pi(\mathbf{s}), \tag{7}$$

where, $C_\pi(\mathbf{s}) = \iint \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) \right)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{S_t} = \mathbf{s}, \mathbf{A_t} = \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{S_t} = \mathbf{s})$, *is the expected, clipped difference in the state value function,* $\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})$, *over all actions,* $\mathbf{a}$, *and next states,* $\mathbf{s}'$, *under the policy given state,* $\mathbf{s}$. *Here, we use* $\int \ldots d\Pi(\mathbf{a} \mid \mathbf{s})$ *to denote* $\sum_{\mathbf{a}} \ldots \pi(\mathbf{a} \mid \mathbf{s})$ *for discrete action spaces and* $\int \ldots \pi(\mathbf{a} \mid \mathbf{s}) d\mathbf{a}$ *for continuous action spaces. Similarly, we use* $\int \ldots d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ *to denote* $\sum_{\mathbf{s}'} \ldots p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$ *for discrete state spaces and* $\int \ldots p(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) d\mathbf{s}'$ *for continuous state spaces.*

Proof. Lemma B.1 shows that the policy-gradient theorem (Sutton et al., 1999) can be expressed in terms of the clipped advantage function,

$$h_\pi^+(\mathbf{s}, \mathbf{a}) = \left(q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi(\mathbf{s}) \right)^+ := \max(0, q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi(\mathbf{s})),$$

as,

$$\begin{aligned}
\nabla v_\pi(\mathbf{s}) = &\int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^k \int_{\mathcal{A}} h_\pi^+(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x}) \right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi) \\
&+ \int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^k \int_{\mathcal{A}} \mathbb{1}\left(q_\pi(\mathbf{x}, \mathbf{a}) > v_\pi(\mathbf{x}) \right) v_\pi(\mathbf{x}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x}) \right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi) \\
&+ \int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^k \int_{\mathcal{A}} \mathbb{1}\left(q_\pi(\mathbf{x}, \mathbf{a}) \le v_\pi(\mathbf{x}) \right) q_\pi(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x}) \right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi),
\end{aligned} \tag{8}$$

where, $\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi)$, is the probability of transitioning from state $\mathbf{s}$ to state $\mathbf{x}$ in k steps under policy π.

The first right hand side term above defines the gradient of the lower-bound, $v_\pi^*(\mathbf{s})$, with respect to $\boldsymbol{\theta}$:

$$\nabla v_\pi^*(\mathbf{s}) := \int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^k \int_{\mathcal{A}} h_\pi^+(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x}) \right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi). \tag{9}$$

Letting, $\nabla v_\pi^*(\mathbf{s_0}) = \int_{\mathcal{S}} \sum_{k=0}^{\infty} \gamma^k \int_{\mathcal{A}} h_\pi^+(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) d\mathbb{P}(\mathbf{s_0} \to \mathbf{s}; k, \pi)$, a straightforward continuation of the policy gradient theorem (Sutton et al., 1999) will show that

$$\nabla J(\boldsymbol{\theta}) := \nabla v_\pi^*(\mathbf{s_0}) \propto \iint h_\pi^+(\mathbf{s}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) dP(\mathbf{s}).$$

We then arrive at Equation (6) by moving from the all states/actions to single state/action formulation:

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &:= \nabla v_\pi^*(\mathbf{s_0}), & \text{by definition} \\
&\propto \iint \left(q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi(\mathbf{s}) \right)^+ \nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) dP(\mathbf{s}), & \text{Sutton et al. (1999)} \\
&= \mathbb{E}_\pi \left[\int \left(q_\pi(\mathbf{S_t}, \mathbf{a}) - v_\pi(\mathbf{S_t}) \right)^+ \nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta}) \right], \\
&= \mathbb{E}_\pi \left[\int \left(q_\pi(\mathbf{S_t}, \mathbf{a}) - v_\pi(\mathbf{S_t}) \right)^+ \frac{\nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta})}{d\Pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta}) \right], \\
&= \mathbb{E}_\pi \left[\int \left(q_\pi(\mathbf{S_t}, \mathbf{A_t}) - v_\pi(\mathbf{S_t}) \right)^+ \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A_t} \mid \mathbf{S_t}, \boldsymbol{\theta}) \right].
\end{aligned}$$

Now we need to show that,

$$v_\pi^*(\mathbf{s}) \le v_\pi(\mathbf{s}) + \iint \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\right)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t) d\Pi(\mathbf{a} \mid \mathbf{S}_t = \mathbf{s}).$$

To do so, we will first prove that it holds for episodes, T, of length 1, then that it holds for episodes of length 2. These two proofs will then prove Equation (7) for episodes of arbitrary length by mathematical induction and conclude the proof.

For episodes of length 1, $|T| = 1$, we have

$$
\begin{aligned}
\nabla v_\pi(\mathbf{s}) &= \int q_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \nabla q_\pi(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), \\
&= \int q_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \left(\nabla \int \mathrm{r} d\mathbb{P}(\mathrm{r} \mid \mathbf{s}, \mathbf{a})\right) d\Pi(\mathbf{a} \mid \mathbf{s}), \\
&= \int q_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}), \\
&= \int h_\pi^+(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \left(\mathbb{1}(q_\pi > v_\pi) v_\pi(\mathbf{s}) + \mathbb{1}(q_\pi \le v_\pi) q_\pi(\mathbf{s}, \mathbf{a})\right) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}).
\end{aligned}
\tag{11}
$$

Therefore, for $|T| = 1$,

$$\nabla v_\pi^*(\mathbf{s}) = \int h_\pi^+(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})$$

In order to recover $v_\pi^*(\mathbf{s})$, we need to use the work of Willse (2019) to define an inverse function for the gradient. Assume that the policy, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$, is a smooth, infinitely differentiable function with respect to $\boldsymbol{\theta}$. Further, let the gradient of the policy,

$$\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \pi(\mathbf{a} \mid \mathbf{s}, \theta_1), \\ \vdots \\ \frac{\partial}{\partial \theta_k} \pi(\mathbf{a} \mid \mathbf{s}, \theta_k) \end{pmatrix}, \tag{12}$$

be a conservative vector field. We call $\tilde{\beta}\big(\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})\big)$ the inverse of the gradient operation, $\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$. Assuming that $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$ is a representative of $\tilde{\beta}$, we have that,

$$
\begin{aligned}
\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) &= \tilde{\beta}\big(\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})\big), \\
&= \int_\gamma \nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) d\mathbf{x}, \\
&= \int_\gamma \frac{\partial}{\partial \theta_1} \pi(\mathbf{a} \mid \mathbf{s}, \theta_1) d\theta_1 + \cdots + \frac{\partial}{\partial \theta_k} \pi(\mathbf{a} \mid \mathbf{s}, \theta_k) d\theta_k,
\end{aligned}
\tag{13}
$$

where γ is a path from the fixed reference point, $\boldsymbol{\theta}_0$, to $\boldsymbol{\theta}$. The conservativeness of $\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$ guarantees that the integrals are path independent.

Now we have,

$$
\begin{aligned}
v_\pi^*(\mathbf{s}) &= \tilde{\beta}\left(\int h_\pi^+(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})\right), \\
&= \int h_\pi^+(\mathbf{s}, \mathbf{a}) \tilde{\beta}\big(\nabla d\Pi(\mathbf{a} \mid \mathbf{s})\big), && \text{linearity} \\
&= \int h_\pi^+(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), && \text{Equation (13)} \\
&\le \iint \left(\mathrm{r} + \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\right)^+\right) d\mathbb{P}(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), && \text{Lemma B.2} \\
&= v_\pi(\mathbf{s}) + \iint \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\right)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), && |T| = 1
\end{aligned}
$$

which concludes the proof for episodes of length 1.

For episodes of length 2, $|T| = 2$, we have

$$\nabla v_\pi(\mathbf{s}) = \int q_\pi(\mathbf{s}, \mathbf{a})\nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \nabla q_\pi(\mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}),$$

$$= \int q_\pi(\mathbf{s}, \mathbf{a})\nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \iiint q_\pi(\mathbf{s}', \mathbf{a}')\nabla d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s})$$

$$+ \iiint \left(\nabla \int \mathrm{r}'d\mathbb{P}(\mathrm{r}' \mid \mathbf{s}', \mathbf{a}')\right)d\Pi(\mathbf{a}' \mid \mathbf{s}'),$$

$$= \int q_\pi(\mathbf{s}, \mathbf{a})\nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \iiint q_\pi(\mathbf{s}', \mathbf{a}')\nabla d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s}),$$

$$= \int h_\pi^+(\mathbf{s}, \mathbf{a})\nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \iiint h_\pi^+(\mathbf{s}', \mathbf{a}')\nabla d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s})$$

$$+ \int \left(\mathbb{1}(q_\pi > v_\pi)v_\pi(\mathbf{s}) + \mathbb{1}(q_\pi \leq v_\pi)q_\pi(\mathbf{s}, \mathbf{a})\right)\nabla d\Pi(\mathbf{a} \mid \mathbf{s})$$

$$+ \iiint \left(\mathbb{1}(q_\pi > v_\pi)v_\pi(\mathbf{s}') + \mathbb{1}(q_\pi \leq v_\pi)q_\pi(\mathbf{s}', \mathbf{a}')\right)\nabla d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s}).$$

Therefore, for $|T| = 2$,

$$\nabla v_\pi^*(\mathbf{s}) = \int h_\pi^+(\mathbf{s}, \mathbf{a})\nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \iiint h_\pi^+(\mathbf{s}', \mathbf{a}')\nabla d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s}).$$

Finally, we apply the $\tilde{\beta}$ operator:

$$v_\pi^*(\mathbf{s}) = \tilde{\beta}\left(\int h_\pi^+(\mathbf{s}, \mathbf{a})\nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \iiint h_\pi^+(\mathbf{s}', \mathbf{a}')\nabla d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s})\right),$$

$$= \int h_\pi^+(\mathbf{s}, \mathbf{a})\tilde{\beta}\left(\nabla d\Pi(\mathbf{a} \mid \mathbf{s})\right) + \iiint h_\pi^+(\mathbf{s}', \mathbf{a}')\tilde{\beta}\left(\nabla d\Pi(\mathbf{a}' \mid \mathbf{s}')\right)d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s}), \quad \text{linearity}$$

$$= \int h_\pi^+(\mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}) + \iiint h_\pi^+(\mathbf{s}', \mathbf{a}')d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s}), \quad \text{Equation (13)}$$

$$\leq \iint \mathrm{r}d\mathbb{P}(\mathrm{r} \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}) + \iint \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\right)^+d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s})$$

$$+ \iiint h_\pi^+(\mathbf{s}', \mathbf{a}')d\Pi(\mathbf{a}' \mid \mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s}), \quad \text{Lemma B.2}$$

$$\leq \iint \mathrm{r}d\mathbb{P}(\mathrm{r} \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}) + \iint \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\right)^+d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s})$$

$$+ \iint \gamma v_\pi(\mathbf{s}')d\mathbb{P}(\mathbf{s}' \mid \mathbf{a}, \mathbf{s})d\Pi(\mathbf{a} \mid \mathbf{s}), \quad \text{Lemma B.3}$$

$$= v_\pi(\mathbf{s}) + \iint \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\right)^+d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})d\Pi(\mathbf{a} \mid \mathbf{s}). \quad \text{rearranging terms}$$

$\square$

Negative Rewards Add a Known Constant to the Lower Bound, but Do Not Impact Optimization. Consider the scenario where rewards can be negative ($r_{\min} \leq 0$), and we adjust observed rewards by subtracting $r_{\min}$. This adjustment leads to the following expressions for the state-value and action-value functions, respectively:

$$v_\pi^{r_{\min}}(s) := E_\pi\left[\sum_{k=t+1}^{T} \gamma^{k-t-1}(R_k - r_{\min}) \mid S_t = s\right] = v_\pi(s) - \sum_{k=t+1}^{T} \gamma^{k-t-1}r_{\min}$$

$$q_\pi^{r_{\min}}(s,a) := E_\pi\left[\sum_{k=t+1}^{T} \gamma^{k-t-1}(R_k - r_{\min}) \mid S_t = s, A_t = a\right] = q_\pi(s,a) - \sum_{k=t+1}^{T} \gamma^{k-t-1} r_{\min}$$

In both episodic $(0 < T < \infty)$ and continuing $(T \to \infty)$ cases, the transformations yield:

$$v_\pi^{r_{\min}}(s) \le v_\pi(s) - \frac{\gamma}{1-\gamma} r_{\min}, \quad q_\pi^{r_{\min}}(s,a) \le q_\pi(s,a) - \frac{\gamma}{1-\gamma} r_{\min}.$$

Crucially, the advantage function remains invariant under this transformation:

$$
\begin{aligned}
h_\pi^{r_{\min}}(s,a) &= q_\pi^{r_{\min}}(s,a) - v_\pi^{r_{\min}}(s) \\
&= q_\pi(s,a) - \sum_{k=t+1}^{T} \gamma^{k-t-1} r_{\min} - \left(v_\pi(s) - \sum_{k=t+1}^{T} \gamma^{k-t-1} r_{\min}\right) \\
&= q_\pi(s,a) - v_\pi(s) \\
&= h_\pi(s,a)
\end{aligned}
\tag{17}
$$

Imagine now that we use the transformed advantage function $h_\pi^{r_{\min}}(s,a)$ to ensure non-negativity in our proof for Theorem 3.1. Then, our objective maximizes:

$$
\begin{aligned}
v_\pi^{*,r_{\min}}(s) &= v_\pi^{r_{\min}}(s) + C_\pi^{r_{\min}}(s) \\
&= v_\pi^{r_{\min}}(s) + \iint \left(\gamma v_\pi^{r_{\min}}(s') - v_\pi^{r_{\min}}(s)\right)^+ dP(s' \mid s, a) d\Pi(a \mid s)
\end{aligned}
\tag{18}
$$

Let's focus on the integrand of the $C_\pi^{r_{\min}}(s)$ term:

$$
\begin{aligned}
\left(\gamma v_\pi^{r_{\min}}(s') - v_\pi^{r_{\min}}(s)\right)^+ &= \left(\gamma v_\pi(s') - v_\pi(s) + \sum_{k=t+1}^{T} \gamma^{k-1-t} r_{\min} - \gamma \sum_{k=t+2}^{T} \gamma^{k-1-t} r_{\min}\right)^+ \\
&= \left(\gamma v_\pi(s') - v_\pi(s) + r_{\min} + \gamma r_{\min} - \gamma^{T-t} r_{\min}\right)^+ \\
&= \left(\gamma v_\pi(s') - v_\pi(s) + c_t\right)^+ : c_t < 0 \\
&= \frac{1}{2}\left(\gamma v_\pi(s') - v_\pi(s) + c_t + |\gamma v_\pi(s') - v_\pi(s) + c_t|\right) \\
&\le \left(\gamma v_\pi(s') - v_\pi(s)\right)^+ + \frac{1}{2}(c_t + |c_t|) \\
&= \left(\gamma v_\pi(s') - v_\pi(s)\right)^+
\end{aligned}
\tag{19}
$$

Therefore, $C_\pi^{r_{\min}}(s) <= C_\pi(s)$, which maintains the direction of the lower bound. So, should the rewards be negative, we optimize

$$v_\pi^{*,r_{\min}}(s) \le v_\pi(s) - \frac{\gamma}{1-\gamma} r_{\min} + C_\pi(s).$$

This expression has an extra non-negative (yet fully determined) constant $-\frac{\gamma}{1-\gamma} r_{\min}$. Importantly, this result has no bearing on the optimization procedure. The invariance of the advantage function ensures that the performance function is also invariant to a shift in rewards:

$$\nabla_\theta J(\theta) := E_\pi\left[\left(q_\pi(S_t, A_t) - v_\pi(S_t)\right)^+ \nabla_\theta \log \pi(A_t \mid S_t, \theta)\right] = E_\pi\left[\left(q_\pi^{r_{\min}}(S_t, A_t) - v_\pi^{r_{\min}}(S_t)\right)^+ \nabla_\theta \log \pi(A_t \mid S_t, \theta)\right],$$

which entails that no special measures need to be taken when rewards can take negative values.

Lemma B.1. $\nabla v_\pi(\mathbf{s})$ *can be written in terms of* $h_\pi^+(\mathbf{s}, \mathbf{a})$.

Proof.

$$\nabla v_\pi(\mathbf{s}) = \nabla \left[\int q_\pi(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) \right], \tag{20a}$$

$$= \int q_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \nabla q_\pi(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), \tag{20b}$$

$$= \int \left(h_\pi^+(\mathbf{s}, \mathbf{a}) + \mathbb{1}(q_\pi > v_\pi) v_\pi(\mathbf{s}) + \mathbb{1}(q_\pi \leq v_\pi) q_\pi(\mathbf{s}, \mathbf{a}) \right) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})$$
$$+ \int \nabla q_\pi(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), \tag{20c}$$

$$= \int \left(h_\pi^+(\mathbf{s}, \mathbf{a}) + \mathbb{1}(q_\pi > v_\pi) v_\pi(\mathbf{s}) + \mathbb{1}(q_\pi \leq v_\pi) q_\pi(\mathbf{s}, \mathbf{a}) \right) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})$$
$$+ \int \nabla \left[\int (\mathbf{r} + \gamma v_\pi(\mathbf{s}')) d\mathbb{P}(\mathbf{s}', \mathbf{r} \mid \mathbf{s}, \mathbf{a}) \right] d\Pi(\mathbf{a} \mid \mathbf{s}), \tag{20d}$$

$$= \int \left(h_\pi^+(\mathbf{s}, \mathbf{a}) + \mathbb{1}(q_\pi > v_\pi) v_\pi(\mathbf{s}) + \mathbb{1}(q_\pi \leq v_\pi) q_\pi(\mathbf{s}, \mathbf{a}) \right) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})$$
$$+ \gamma \iint \nabla v_\pi(\mathbf{s}') d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), \tag{20e}$$

$$= \int \left(h_\pi^+(\mathbf{s}, \mathbf{a}) + \mathbb{1}(q_\pi > v_\pi) v_\pi(\mathbf{s}) + \mathbb{1}(q_\pi \leq v_\pi) q_\pi(\mathbf{s}, \mathbf{a}) \right) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})$$
$$+ \gamma \iint \left[\int q_\pi(\mathbf{s}', \mathbf{a}') \nabla d\Pi(\mathbf{a}' \mid \mathbf{s}') \right.$$
$$\left. + \gamma \int \nabla v_\pi(\mathbf{s}'') d\mathbb{P}(\mathbf{s}'' \mid \mathbf{s}', \mathbf{a}') d\Pi(\mathbf{a}' \mid \mathbf{s}') \right] d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), \tag{20f}$$

$$= \int \left(h_\pi^+(\mathbf{s}, \mathbf{a}) + \mathbb{1}(q_\pi > v_\pi) v_\pi(\mathbf{s}) + \mathbb{1}(q_\pi \leq v_\pi) q_\pi(\mathbf{s}, \mathbf{a}) \right) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})$$
$$+ \gamma \iint \left[\int \left(h_\pi^+(\mathbf{s}', \mathbf{a}') + \mathbb{1}(q_\pi > v_\pi) v_\pi(\mathbf{s}') + \mathbb{1}(q_\pi \leq v_\pi) q_\pi(\mathbf{s}', \mathbf{a}') \right) \nabla d\Pi(\mathbf{a}' \mid \mathbf{s}') \right.$$
$$\left. + \gamma \int \nabla v_\pi(\mathbf{s}'') d\mathbb{P}(\mathbf{s}'' \mid \mathbf{s}', \mathbf{a}') d\Pi(\mathbf{a}' \mid \mathbf{s}') \right] d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), \tag{20g}$$

$$= \int_\mathcal{S} \sum_{k=0}^\infty \left[\gamma^k \int_\mathcal{A} h_\pi^+(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x}) \right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi)$$
$$+ \int_\mathcal{S} \sum_{k=0}^\infty \left[\gamma^k \int_\mathcal{A} \mathbb{1}(q_\pi(\mathbf{x}, \mathbf{a}) > v_\pi(\mathbf{x})) v_\pi(\mathbf{x}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x}) \right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi)$$
$$+ \int_\mathcal{S} \sum_{k=0}^\infty \left[\gamma^k \int_\mathcal{A} \mathbb{1}(q_\pi(\mathbf{x}, \mathbf{a}) \leq v_\pi(\mathbf{x})) q_\pi(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x}) \right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi) \tag{20h}$$

$\square$

Lemma B.2.

$$\underline{v}_\pi^{v_\pi}(\mathbf{s}) \leq \iint \mathbf{r} \, d\mathbb{P}(\mathbf{r} \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) + \iint \left(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) \right)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s})$$

Proof.

$$
\begin{aligned}
\underline{v}_\pi^{v_\pi}(\mathbf{s}) &:= \int h_\pi^+(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) \\
&= \frac{1}{2} \int \Big(q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi + \big| q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi \big| \Big) d\Pi(\mathbf{a} \mid \mathbf{s}) && (2\max(0, a) = a + |a|) \\
&= \frac{1}{2} \int \bigg(\Big(\int (\mathrm{r} + \gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})) d\mathbb{P}(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a}) \\
&\qquad + \Big| \int (\mathrm{r} + \gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})) d\mathbb{P}(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a}) \Big| \bigg) d\Pi(\mathbf{a} \mid \mathbf{s}) \\
&\leq \frac{1}{2} \iint \Big(\mathrm{r} + \gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) + \big| \mathrm{r} + \gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) \big| \Big) && \text{(Jensen's inequality)} \\
&\qquad d\mathbb{P}(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) \\
&\leq \frac{1}{2} \iint \Big(2\mathrm{r} + \gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) + \big| \gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) \big| \Big) && \text{(triangle inequality)} \\
&\qquad d\mathbb{P}(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) \\
&= \iint \Big(\mathrm{r} + \big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s}) \big)^+ \Big) d\mathbb{P}(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) && (2\max(0, a) = a + |a|)
\end{aligned}
$$

$\square$

Lemma B.3. *When, without loss of generality, rewards, $\mathrm{R_t}$, are assumed to be non-negative:*

$$
\underline{v}_\pi^{v_\pi}(\mathbf{s}) := \int h_\pi^+(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) \leq v_\pi(\mathbf{s})
$$

Proof.

$$
\begin{aligned}
\int h_\pi^+(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) &= \frac{1}{2} \int \Big(q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi + \big| q_\pi(\mathbf{s}, \mathbf{a}) - v_\pi \big| \Big) d\Pi(\mathbf{a} \mid \mathbf{s}) && (\, 2\max(0, a) = a + |a| \,) \\
&\leq \int q_\pi(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}) && \text{(triangle inequality)} \\
&= v_\pi(\mathbf{s})
\end{aligned}
$$

$\square$

B.2. Relation to Regret Matching Policy Gradient (RMPG)

Here we provide a derivation starting from RMPG and arriving at our method.

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_\pi \left[\int_\mathcal{A} \Big(q_\pi(\mathbf{S_t}, \mathbf{a}) - \int_\mathcal{A} \pi(\mathbf{a}' \mid \mathbf{S_t}, \boldsymbol{\theta}) q_\pi(\mathbf{S_t}, \mathbf{a}') d\mathbf{a}' \Big)^+ \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta}) d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[\int_\mathcal{A} (q_\pi(\mathbf{S_t}, \mathbf{a}) - v_\pi(\mathbf{S_t}))^+ \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta}) d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[\int_\mathcal{A} h_\pi^+(\mathbf{S_t}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta}) d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[\int_\mathcal{A} \pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta}) h_\pi^+(\mathbf{S_t}, \mathbf{a}) \frac{\nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta})}{\pi(\mathbf{a} \mid \mathbf{S_t}, \boldsymbol{\theta})} d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[h_\pi^+(\mathbf{S_t}, \mathbf{A_t}) \frac{\nabla_{\boldsymbol{\theta}} \pi(\mathbf{A_t} \mid \mathbf{S_t}, \boldsymbol{\theta})}{\pi(\mathbf{A_t} \mid \mathbf{S_t}, \boldsymbol{\theta})} \right] \\
&= \mathbb{E}_\pi \left[h_\pi^+(\mathbf{S_t}, \mathbf{A_t}) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A_t} \mid \mathbf{S_t}, \boldsymbol{\theta}) \right]
\end{aligned}
$$

Lemma B.4.

$$ReLU(a) <= |a|$$

Proof.

$$
\begin{aligned}
\text{ReLU}(a) &= \max(0, a) \\
&= \frac{1}{2}a + \frac{1}{2}|a| \\
&= \begin{cases} a & a \geq 0 \\ 0 & a < 0 \end{cases} \\
&\leq \begin{cases} a & a \geq 0 \\ -a & a < 0 \quad (-a > 0) \end{cases} \\
&= |a|
\end{aligned}
$$

$\square$

C. Commentary

The derivation in Equation (3) assumes access to the policy precision parameter, $\tau = 1/\sigma^2$, and samples of H_t. In practice, we fit τ using maximum likelihood estimation and use clipped GAEs to obtain samples of H_t. Moreover, it is only valid for continuous action spaces. We evaluate discrete action spaces below but leave theoretical grounding for future work.

Note that while we show in Equation (3) that approximate Bayesian inference of $\boldsymbol{\theta}$ under an assumed policy that scales actor precision, τ, by clipped advantages, h^+, yields an equivalent likelihood objective, we do not implement a policy, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$, that includes this scaling. We leave this exploration to future work as it requires joint inference over τ and $\boldsymbol{\theta}$ and an appropriate state conditional advantage estimator.

Finally, the conservative vector field assumption of Theorem 3.1 assumes that the actor implements a smooth function. This assumption is often broken in practice as non-smooth ReLU activation functions see use in the baselines we compare to. We leave the investigation of using smooth activation functions to future work.

C.1. Concerning $C_\pi(\mathbf{s})$, K_π-Lipschitz Continuity, and Spectral Normalization

In light of the dependence of the Lipschitz constant, K_π, on the policy, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$, we offer insight into the roles played by the Lipschitz assumption and the use of critic weight spectral normalization. When we do gradient ascent according to,

$$
\mathbb{E}_\pi \left[(q_\pi(\mathbf{S}_t, \mathbf{A}_t) - v_\pi(\mathbf{S}_t))^+ \nabla_\theta \log \pi(\mathbf{A}_t \mid \mathbf{S}_t, \theta), \right]
$$

we show that we maximize

$$
v_\pi^*(\mathbf{s}) \leq v_\pi(\mathbf{s}) + C_\pi(\mathbf{s}).
$$

We want this optimization to lead to a policy π that maximizes value, v_π, but perhaps it could lead to an undesirable policy that instead maximizes C_π. We show that,

$$
C_\pi(\mathbf{s}) \leq \frac{1}{2} \iint |v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})| \, dP(\mathbf{s}' \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{S}_t = \mathbf{s}).
$$

In theory, a policy that leads to large fluctuations in value, v_π, as the agent transitions from state, $\mathbf{s}$, to state, $\mathbf{s}'$, could maximize this objective.

Assuming that the value function, $v_\pi(\mathbf{s})$, is K_π-Lipschitz continuous allows us to express this bound as

$$
C_\pi(\mathbf{s}) \leq \frac{1}{2} \iint K_\pi \|\mathbf{s}' - \mathbf{s}\| \, dP(\mathbf{s}' \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{S}_t = \mathbf{s}),
$$

but this does not solve the problem in itself: it could still be possible to learn a policy that merely maximizes K_π instead of $v_\pi(\mathbf{s})$.

Hence, when we use spectral normalization of the critic weights, we regularize K_π to be 1. We find this regularization provides increased performance in most experiments run thus far. But empirically, it does not seem like the pathological behavior of maximizing $C_\pi(\mathbf{s})$ is happening to a significant extent even when we do not use spectral normalization. For example, we can see in Figure 1 that the performance of VSOP without spectral normalization is about equal to that of PPO on MuJoCo.

Next, we believe this analysis gives us further insight into understanding how we observe spectral normalization detrimental in highly parallel settings. In the single-threaded setting, a single agent collects data. This specific experience from a single initialization, coupled with the flexibility of Neural Networks, could result in the objective maximizing a policy that encourages spuriously high-frequency (rather than high-value) value functions when the data is sparse early in training. In this case, regularization from spectral normalization would be beneficial. Conversely, the algorithm collects data from many agents with unique initializations in the highly parallel setting. Thus, with more diverse and less sparse data, we can expect more robust value function estimates, less likely to be spuriously high-frequency between state transitions. Then, the $K_\pi = 1$ assumption induced by spectral normalization may be too strong and lead to over-regularization.

C.2. Concerning the Normal-Gamma Assumption

Is the Normal-Gamma Assumption Necessary? The gamma-normal assumption allows us to interpret adding dropout and weight-decay regularization as sensible approximate Bayesian inference without adding complex computational overhead to the original A3C optimization algorithm. As such, this assumption primarily serves to ground Thompson sampling through approximate Bayesian inference and is not requisite for Theorem 3.1. As with the original result of the policy gradient theorem, the results in Equations (5-6) do not make any distributional assumptions on π and should hold for all policies with differentiable probability densities/distributions.

Are the Clipped Advantages Gamma Distributed? The intuition behind assuming a gamma distribution for the clipped advantages is that advantages ideally have zero mean by construction (we subtract the state-action value by its expected state value over actions), so clipping at zero will result in a heavy-tailed distribution. Gamma distributions are sensible hypotheses for heavy-tailed distributions. In Figure 8 we plot the marginal histograms for the advantages (left) and the clipped advantages (right) over each training update for a training run of Humanoid-v4.
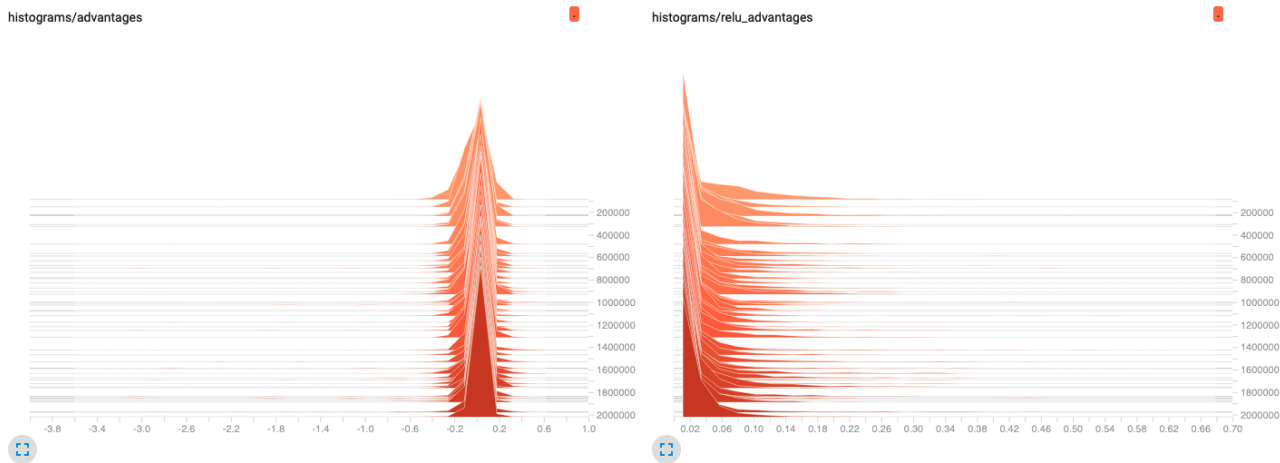

Figure 8: Comparing the histograms of estimated advantages (left) and ReLU'ed advantages (right).

The clipped advantage histogram on the right lends evidence to the gamma assumption (at least for the marginal distribution). We may expect multi-modality at the state-action level, which integration over actions may marginalize out at the state level; however, we would still expect a heavy tail in both cases.

D. Implementation Details

We have attached the code that replicates the reported results in the folder "vsop-main" and will release a public github repo after the review process.

D.1. Gymansium

We build off of Huang et al. (2022)'s CleanRL package which provides reproducible, user-friendly implementations of state-of-the-art reinforcement learning algorithms using PyTorch (Paszke et al., 2019), Gymnasium (Brockman et al., 2016; Todorov et al., 2012), and Weights & Biases (Biases, 2018). Several code-level optimizations (Engstrom et al., 2020; Andrychowicz et al., 2021) key to PPO reproducibility are superfluous for our method. We omit advantage normalization, value loss clipping (Schulman et al., 2017), gradient clipping, and modification of the default Adam (Kingma & Ba, 2014) epsilon parameter as they either do not lead to an appreciable difference in performance or have a slightly negative effect. However, we find that orthogonal weight initialization, learning rate annealing, reward scaling/clipping, and observation normalization/clipping remain to have non-negligible positive effects on performance (Engstrom et al., 2020; Andrychowicz et al., 2021). In addition to adding dropout, weight decay regularization, and spectral normalization, we also look at model architecture modifications not present in the CleanRL implementation: layer width, number of hidden layers, layer activation, layer normalization (Ba et al., 2016), and residual connections. We find that ReLU activation functions (Nair & Hinton, 2010), increasing layer width to 256, and a dropout rate of 0.01-0.04 are beneficial. We find that network depth and residual connections are benign overall. In contrast to recent findings in the context of offline data for off-policy reinforcement learning (Ball et al., 2023), layer normalization — whether applied to the actor, the critic, or both — is detrimental to performance.

Table 2: Hyper-parameters for ablation of mechinism study. VSOP, no-spectral, no-Thompson, all-actions, and no ReLU Advantage variants across Gymnasium MuJoCo environments

| Parameter | Gymnasium MuJoCo | | | | |
	VSOP	no-Spectral	all-actions	no-ReLU Adv.	no-Thompson
timesteps	3e6	3e6	3e6	3e6	3e6
num. envs	1	1	1	1	1
num. steps	2048	2048	2048	2048	2048
learning rate	2e-4	5.5e-4	2e-4	7.5e-4	2.5e-4
anneal lr	True	True	True	True	True
optim. ϵ.	1e-8	1e-8	1e-8	1e-8	1e-8
GAE γ	0.99	0.99	0.99	0.99	0.99
GAE λ	0.61	0.93	0.60	0.99	0.76
num. minibatch	32	2	4	1	32
update epochs	9	6	10	5	8
clip v-loss	False	False	False	False	False
v-loss coef.	0.5	0.5	0.5	0.5	0.5
max grad. norm.	7.1	8.5	6.4	8.5	7.2
norm. obs.	True	True	True	True	True
norm. reward	True	True	True	True	True
width	256	256	256	256	256
activation	relu	relu	relu	relu	relu
weight decay	2.4e-4	2.4e-4	2.4e-4	2.4e-4	2.4e-4
dropout	0.025	0.005	0.0	0.025	0.05

In Table 2 we present the hyperparameters used in the ablation of mechanisms study. In Table 3, we present the hyperparameters used for the VSOP, VSPPO, RMPG, A3C, and PPO algorithms when trained on Gymnasium MuJoCo environments. The table lists hyperparameters such as the number of timesteps, thread number, and learning rate, among others. Each algorithm may have a unique set of optimal hyperparameters. Please note that some hyperparameters: 'clip ϵ', 'norm. adv.', and 'clip v-loss' may not apply to all algorithms, as these are specific to certain policy optimization methods. The 'width' and 'activation' fields correspond to the architecture of the neural network used by the policy, and the 'weight decay' and 'dropout' fields pertain to the regularization techniques applied during training. In general, tuning these hyperparameters is crucial to achieving optimal performance. Note that Adam optimization (Kingma & Ba, 2014) is used for all algorithms except for A3C where RMSProp (Hinton et al., 2012) is used.

We report mean values and 95% confidence intervals over ten random seeds.

Table 3: Hyper-parameters for PPO, VSOP, RMPG, A3C, and VSPPO algorithms across Gymnasium MuJoCo environments

Parameter	Gymnasium MuJoCo				
	VSOP	VSPPO	RMPG	A3C	PPO
timesteps	3e6	3e6	3e6	3e6	3e6
num. envs	1	1	1	1	1
num. steps	2048	2048	2048	5	2048
learning rate	2e-4	2.5e-4	2e-4	7e-4	3e-4
anneal lr	True	True	True	True	True
optim. ϵ.	1e-8	1e-8	1e-8	3e-6	1e-5
GAE γ	0.99	0.99	0.99	0.99	0.99
GAE λ	0.61	0.89	0.60	1.0	0.95
num. minibatch	32	64	4	1	32
update epochs	9	9	10	1	10
norm. adv.	False	False	False	False	True
clip ϵ	N/A	N/A	N/A	N/A	0.2
clip v-loss	False	False	False	False	True
ent. coef.	0.0	0.0	0.0	0.0	0.0
v-loss coef.	0.5	0.5	0.5	0.5	0.5
max grad. norm.	7.1	2.1	6.4	0.5	0.5
norm. obs.	True	True	True	True	True
norm. reward	True	True	True	True	True
width	256	256	256	64	64
activation	relu	relu	relu	tanh	tanh
weight decay	2.4e-4	2.4e-4	2.4e-4	0.0	0.0
dropout	0.025	0.035	0.0	0.0	0.0

D.2. Gymnax

Hyperparameter	Range	Transformation	Transformed Range
num. envs	[2, 8]	2^x where x is int	{4, 8, 16, 32, 64, 128, 256}
num. steps	[2, 8]	2^x where x is int	{4, 8, 16, 32, 64, 128, 256}
λ	[0.0, 1.0]	round to multiple of 0.002	{0.0, 0.002, . . . , 1.0}
learning rate	[1e-4, 1e-3]	round to multiple of 0.00005	{1e-4, 1.5e-5, . . . , 1e-3}
max grad. norm.	[0.2, 5.0]	round to multiple of 0.1	{0.2, 0.3, . . . , 5.0}
num. minibatch	[0, 6]	2^x where x is int	{1, 2, 4, 8, 16, 32, 64}
update epochs	[1, 10]	round to int	{1, 2, 3, ..., 10}
width	[6, 10]	2^x where x is int	{64, 128, 256, 512, 1024}

Table 4: Hyperparameter search space with transformations

We optimize the hyper-parameters for each algorithm for each set of environments using a Bayesian optimization search strategy (Snoek et al., 2012). Each algorithm has a budget of 100 search steps. We use NVIDIA A100 GPUs. The hyperparameters we search over include learning rate, number of steps, number of environments, GAE λ, update epochs, number of minibatches, and the maximum gradient norm. We also search over the hidden layer width for Brax-MuJoCo and MinAtar environments. Each hyperparameter has a specific search space and transformation applied during the search. We summarize the search sapce in Table 4.

For the MinAtar environments, the hyper-parameters search spaces are: the number of steps in $[2, 8]$ (transformed to 2^x where x is the integer part of the sample), GAE λ in $[0.0, 1.0]$ (rounded to the nearest multiple of 0.002), learning rate in $[1e - 4, 1e - 3]$ (rounded to the nearest multiple of 0.00005), update epochs in $[1, 10]$ (rounded to the nearest integer), maximum gradient norm in $[0.0, 5.0]$ (rounded to the nearest multiple of 0.1), number of minibatches in $[0, 6]$ (transformed to 2^x), update epochs in $[1, 10]$ (rounded to the nearest integer), and number of minibatches in $[0, 7]$ (transformed to 2^x),

and hidden layer width in $[6, 10]$ (transformed to 2^x). We set the γ and number of environments to fixed values at 0.99 and 64, respectively.

For MuJoCo-Brax, we do not search over the number of environments or steps. Instead we set them to fixed values at 0.99, 2048, and either 10 or 5, respectively. The search space for the remaining hyper-parameters the same ranges as for the MinAtar environments. Further, we only optimize over the Humanoid, Hopper, and Reacher environments for 20 million steps. We test for each environment for 50 million steps.

Finally, for Classic Control environments, we employ the same hyperparameter search as for MinAtar, except that we search over the number of environments in $[2, 8]$ (transformed to 2^x where x is the integer part of the sample) and we do not search over the hidden layer width, instead setting it to a fixed value of 64.

This strategy allows us to thoroughly explore the hyperparameter space and find values that generalize well across a variety of different tasks. Further it allows us to fairly compare each algorithm. Tables 5 to 7 report the final hyper-parameter values for PPO, VSOP, and A3C.

Table 5: PPO, VSOP, A3C, and DPO Hyper-parameters for MinAtar environments.

Parameter	PPO	VSOP	A3C	DPO
learning rate	9e-4	7.5e-4	7e-4	1e-3
num. envs	128	128	128	128
num. steps	64	32	4	16
GAE γ	0.99	0.99	0.99	0.99
GAE λ	0.70	0.82	0.87	0.70
num. minibatch	8	16	2	8
update epochs	10	9	1	6
max grad. norm.	1.9	2.8	1.3	0.4
width	512	512	512	256
activation	relu	relu	relu	relu
clip ϵ	0.2	N/A	N/A	0.2
ent. coef.	0.01	0.01	0.01	0.01

Table 6: Hyper-parameters for PPO, VSOP, A3C, and DPO algorithms across Brax-MuJoCo environments

Parameter	PPO	VSOP	A3C	DPO
learning rate	4.5e-4	1e-4	7e-4	2e-4
num. envs	2048	2048	2048	2048
num. steps	10	10	5	10
GAE γ	0.99	0.99	0.99	0.99
GAE λ	0.714	1.0	0.97	0.942
num. minibatch	32	64	2	32
update epochs	3	2	1	6
max grad. norm.	3.3	3.7	1.0	0.4
width	512	512	128	512
activation	relu	relu	relu	relu
clip ϵ	0.2	N/A	N/A	0.2
ent. coef.	0.0	0.0	0.0	0.0

All reported results for MinAtar, Classic Control, and MuJoCo-Brax respectively are given by mean values and 68% confidence intervals over 20 random seeds. During tuning we use 2 random seeds and for testing we use a different set of 20 random seeds, as per the guidance of Eimer et al. (2023).

Table 7: Hyper-parameters for PPO, VSOP, A3C, and DPO algorithms across Classic Control environments

Parameter	PPO	VSOP	A3C	DPO
learning rate	1e-3	8.5e-4	5.5e-4	1e-3
num. envs	8	16	8	4
num. steps	8	64	4	4
GAE γ	0.99	0.99	0.99	0.99
GAE λ	0.54	0.58	0.13	1.0
num. minibatch	8	16	8	1
update epochs	3	8	1	10
max grad. norm.	3.4	1.9	3.8	5.0
width	64	64	64	64
activation	tanh	tanh	tanh	tanh
clip ϵ	0.2	N/A	N/A	0.2
ent. coef.	0.01	0.01	0.01	0.01

Method	lr	GAE λ	num. minibatch	update epochs	dropout	ent. coef.
VSOP	4.5e-4	0.88	8	3	0.075	1e-5
PPO	5.0e-4	0.95	8	3	0.000	1e-2

Table 8: Final ProcGen hyperparameters for VSOP

D.3. ProcGen

ProcGen (Cobbe et al., 2020) is a set of 16 environments where game levels are procedurally generated, creating a virually unlimited set of unique levels. We follow the "easy" generalization protocol where, for a given environment, models are trained on 200 levels for 25 million time steps and evaluated on the full distribution of environments. We use the same architecture as PPO in the CleanRL library (Huang et al., 2022), and do a Bayesian optimization hyper-parameter search (Snoek et al., 2012) using the bossfight environment. We search over the learning rate, GAE λ, number of minibatches per epoch, number of epochs per rollout, the dropout rate, and the entropy regularization coefficient. We report the final VSOP hyperparamters in Table 8 and include the relevant PPO hyperparameters for comparison. Note also that, VSOP does not make use of advantage normalization or value loss clipping.

E. Additional Results

E.1. Comparison to Baselines

Figure 9 compares VSOP training curves to baseline algorithms.

E.2. Spectral Normalization and Thompson Sampling May Improve PPO

Interestingly, we see this same trend when applying spectral normalization and dropout to PPO. In Figure 10 we compare how Thompson sampling and spectral norm effect PPO.

E.3. Gymnax Environments

PureJaxRL (Lu et al., 2022) uses Gymnax (Lange, 2022) and Jax (Bradbury et al., 2018) to enable vectorization, which facilitates principled hyper-parameter tuning. Using it, we explore several environments and compare VSOP, PPO, A3C, and DPO. We use Bayesian hyper-parameter optimization (Snoek et al., 2012) and give each algorithm a search budget of 100 steps. We search over hyper-parameters such as the learning rate, number of update epochs, number of mini-batches in an update epoch, the GAE λ parameter, the max gradient norm, and the width of the network. We give full implementation details in Appendix D.2. Table 9 shows the overall ranking of each method. VSOP is competitive with DPO and improves over PPO and A3C.
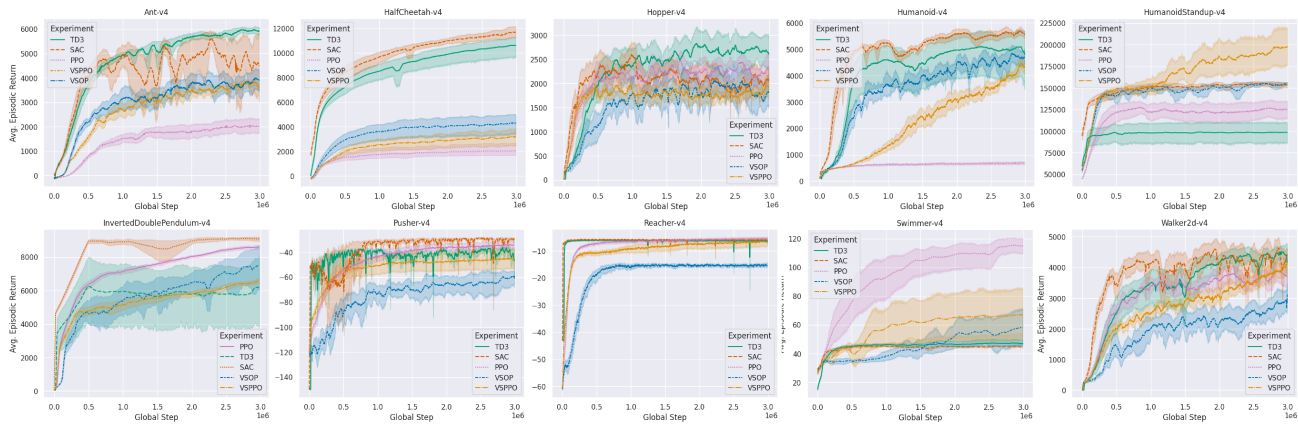
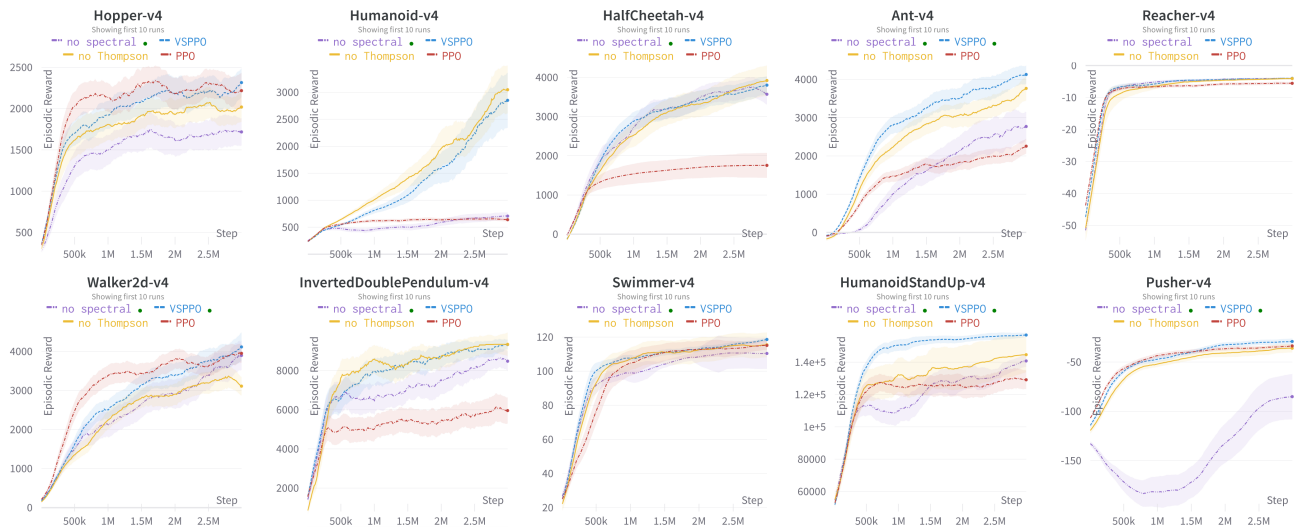Figure 9: Gymnasium-MuJoCo. Comparing VSOP to baseline algorithms.

Figure 10: MuJoCo continuous control benchmark examining the effect of Thompson sampling and spectral normalization on PPO.

Figure 11 summarize the results for **Classic Control**. Performance of each method is in general statistically equal, but VSOP shows significant gain on MountainCar Continuous.

Figure 12 summarize the results for **MinAtar** (Bellemare et al., 2013; Young & Tian, 2019). VSOP shows significant improvement over PPO and A3C in Space Invaders. We see marginal improvement over PPO and DPO in Breakout, with significant improvement over A3C. VSOP trails the baselines significantly in Asterix and Freeway.

Figure 13 summarize the results for **Brax MuJoCo** (Todorov et al., 2012; Freeman et al., 2021). We perform paired t-tests for the last episode between each method and VSOP. We threshold at a p-value of 0.1 to indicate significance. VSOP significantly outperforms A3C in all environments. VSOP significantly outperforms PPO in four of nine environments (InvertedDoublePendulum, Pusher, Reacher, and Walker2d), is statistically equivalent in two environments (Hopper and HumanoidStandUp), and is significantly less effective in three environments (Ant, HalfCheetah, and Humanoid). VSOP outperforms DPO on Ant, is statistically equivalent in four environments (HumanoidStandUp, Pusher, Reacher, and Walker2d), but is significantly less effective in four environments (HalfCheetah, Hopper, Humanoid, and InvertedDoublePendulum). Overall, VSOP outperforms A3C and PPO and is competitive with DPO.

Table 9: Rank scores (lower is better) for VSOP, DPO, PPO, and A3C on Brax-MuJoCo, MinAtar, and Classic Control. Methods are ranked from 1 to 4 based on statistically significant differences (paired t-test with p-value 0.1) between mean last episode returns. Ties are given the same rank, and the proceeding score will be the last rank plus the number of additional methods.

Method	Brax-MuJoCo	MinAtar	Classic Control	Avg. Rank
DPO	**1.33**	**1.75**	1.25	1.44
VSOP (Ours)	1.78	2.50	**1.00**	1.76
PPO	2.00	2.25	1.25	1.83
A3C	4.00	2.25	1.25	2.50

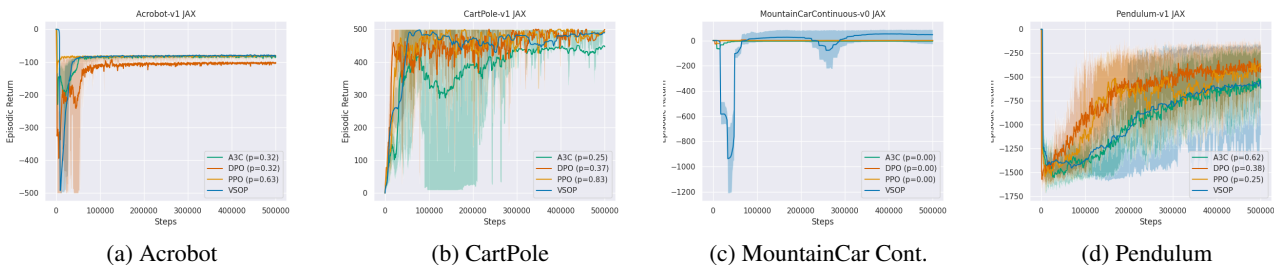| (a) Acrobot | (b) CartPole | (c) MountainCar Cont. | (d) Pendulum |

Figure 11: Classic Control Environments (Lange, 2022). Mean episodic return and 68% CI over 20 random seeds are shown for VSOP (Blue), PPO (Orange), A3C (Green), and DPO (Red). Each method is hyper-parameter tuned using Bayesian Optimization with 100 search steps. Paired t-test p-values for last episode with respect to VSOP shown in brackets. Significant improvement is seen for VSOP compared to all other methods on MountainCar Continuous.

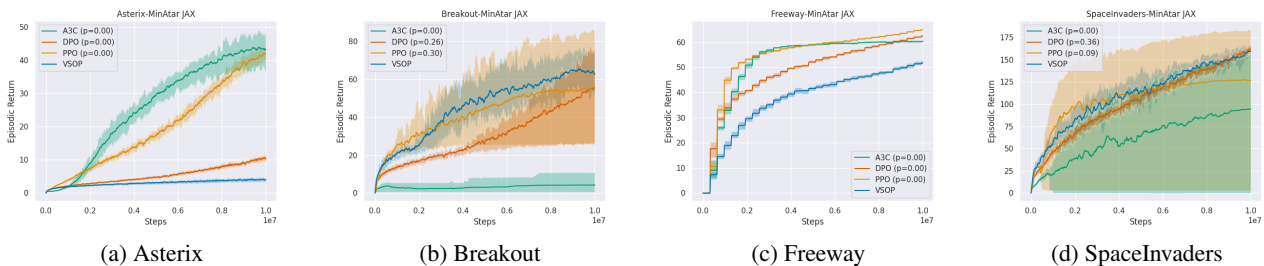| (a) Asterix | (b) Breakout | (c) Freeway | (d) SpaceInvaders |

Figure 12: MinAtar Environments (Young & Tian, 2019). Mean episodic return and 68% CI over 20 random seeds are shown for VSOP (Blue), PPO (Orange), A3C (Green), and DPO (Red). Methods are hyper-parameter tuned using Bayesian Optimization with 100 search steps. p-values for last episode with respect to VSOP shown in brackets. VSOP performs well on Breakout and SpaceInvaders.
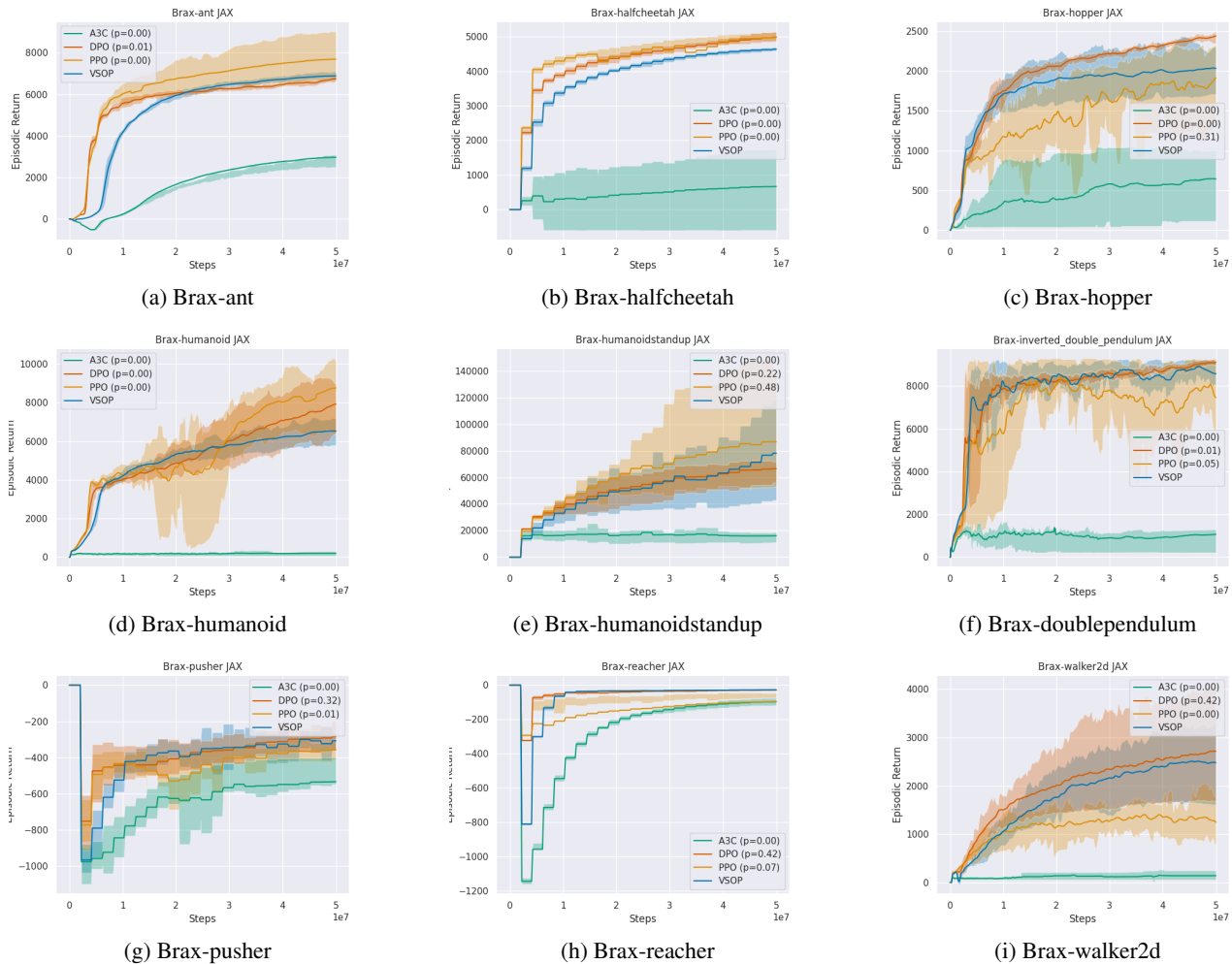
Figure 13: Brax-MuJoCo Environments (Freeman et al., 2021; Todorov et al., 2012). Mean episodic return and 68% CI over 20 random seeds are shown for VSOP (Blue), PPO (Orange), A3C (Green), and DPO (red). Each method is hyper-parameter tuned using Bayesian Optimization (Snoek et al., 2012) with a budget of 100 search steps. Paired t-test p-values for last episode with respect to VSOP shown in brackets. VSOP generally out performs PPO and A3C and is competitive with DPO.