A Broader Impact

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 believe that cautious interaction is a necessary feature for the type of deployed algorithmic decision-making systems the RL community envisions, but that technological solutions alone will not suffice.

B Additional Related Works

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), $\Phi(\theta) \coloneqq \mathbb{E}_\pi \left[\mathrm{G}_\mathrm{t} \mid \mathrm{G}_\mathrm{t} \leq \nu_\alpha \right]$, 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.

C Theoretical Results

C.1 Proof of Theorem 3.1

Theorem C.1. *Let,* $\mathrm{G}_\mathrm{t} \coloneqq \sum_{\mathrm{k}=\mathrm{t}+1}^{\mathrm{T}} \gamma^{\mathrm{k}-1-\mathrm{t}} \mathrm{R}_\mathrm{k}$*, denote the discounted return. Let* $q_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[\mathrm{G}_\mathrm{t} \mid \mathbf{S}_\mathrm{t} = \mathbf{s}, \mathbf{A}_\mathrm{t} = \mathbf{a} \right]$*, denote the state-action value function, and* $v_\pi(\mathbf{s}) = \mathbb{E}_\pi \left[\mathrm{G}_\mathrm{t} \mid \mathbf{S}_\mathrm{t} = \mathbf{s} \right]$*, denote the state value function, under policy* $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$*. Let* $\left(x \right)^+ \coloneqq \max(0, x)$*. Assume, without loss of generality, that rewards,* R_t*, are non-negative. Assume that 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,*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}\left[\Big(q_{\pi}(\mathbf{S}_{\mathrm{t}}, \mathbf{A}_{\mathrm{t}}) - v_{\pi}(\mathbf{S}_{\mathrm{t}})\Big)^{+} \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A}_{\mathrm{t}} \mid \mathbf{S}_{\mathrm{t}}, \boldsymbol{\theta})\right], \tag{8}$$

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

$$v_{\pi}^{*}(\mathbf{s}) \leq v_{\pi}(\mathbf{s}) + C_{\pi}(\mathbf{s}), \tag{9}$$

where, $C_{\pi}(\mathbf{s}) = \iint \Big(\gamma v_{\pi}(\mathbf{s}') - v_{\pi}(\mathbf{s})\Big)^{+} d\mathbb{P}(\mathbf{s}' \mid \mathbf{S}_{\mathrm{t}} = \mathbf{s}, \mathbf{A}_{\mathrm{t}} = \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{S}_{\mathrm{t}} = \mathbf{s})$, 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 C.1 shows that the policy-gradient theorem (Sutton et al., 1999) can be expressed in terms of the clipped advantage function,

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

as,

$$\begin{aligned}
\nabla v_{\pi}(\mathbf{s}) = &\int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^{k} \int_{\mathcal{A}} h_{\pi}^{+}(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x})\right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi) \\
&+ \int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^{k} \int_{\mathcal{A}} \mathbb{1}\big(q_{\pi}(\mathbf{x}, \mathbf{a}) > v_{\pi}(\mathbf{x})\big) v_{\pi}(\mathbf{x}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x})\right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi) \\
&+ \int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^{k} \int_{\mathcal{A}} \mathbb{1}\big(q_{\pi}(\mathbf{x}, \mathbf{a}) \leq v_{\pi}(\mathbf{x})\big) q_{\pi}(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x})\right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi),
\end{aligned} \tag{10}$$

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 $\boldsymbol{\theta}$:

$$\nabla v_{\pi}^{*}(\mathbf{s}) \coloneqq \int_{\mathcal{S}} \sum_{k=0}^{\infty} \left[\gamma^{k} \int_{\mathcal{A}} h_{\pi}^{+}(\mathbf{x}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{x})\right] d\mathbb{P}(\mathbf{s} \to \mathbf{x}; k, \pi). \tag{11}$$

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

$$\nabla J(\boldsymbol{\theta}) \coloneqq \nabla v_{\pi}^{*}(\mathbf{s}_0) \propto \iint h_{\pi}^{+}(\mathbf{s}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) d\mathrm{P}(\mathbf{s}).$$

We then arrive at Equation (8) by moving from the all states/actions to single state/action formulation:

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &\coloneqq \nabla v_{\pi}^{*}(\mathbf{s}_0), && \text{by definition} \\
&\propto \iint \Big(q_{\pi}(\mathbf{s}, \mathbf{a}) - v_{\pi}(\mathbf{s})\Big)^{+} \nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) d\mathrm{P}(\mathbf{s}), && \text{Sutton et al. (1999)} \\
&= \mathbb{E}_{\pi}\left[\int \Big(q_{\pi}(\mathbf{S}_{\mathrm{t}}, \mathbf{a}) - v_{\pi}(\mathbf{S}_{\mathrm{t}})\Big)^{+} \nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{S}_{\mathrm{t}}, \boldsymbol{\theta})\right], \\
&= \mathbb{E}_{\pi}\left[\int \Big(q_{\pi}(\mathbf{S}_{\mathrm{t}}, \mathbf{a}) - v_{\pi}(\mathbf{S}_{\mathrm{t}})\Big)^{+} \frac{\nabla_{\boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{S}_{\mathrm{t}}, \boldsymbol{\theta})}{d\Pi(\mathbf{a} \mid \mathbf{S}_{\mathrm{t}}, \boldsymbol{\theta}} d\Pi(\mathbf{a} \mid \mathbf{S}_{\mathrm{t}}, \boldsymbol{\theta}\right], \\
&= \mathbb{E}_{\pi}\left[\int \Big(q_{\pi}(\mathbf{S}_{\mathrm{t}}, \mathbf{A}_{\mathrm{t}}) - v_{\pi}(\mathbf{S}_{\mathrm{t}})\Big)^{+} \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A}_{\mathrm{t}} \mid \mathbf{S}_{\mathrm{t}}, \boldsymbol{\theta})\right].
\end{aligned}$$

Now we need to show that,

$$v^*_\pi(\mathbf{s}) \leq v_\pi(\mathbf{s}) + \iint \Big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\Big)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{S}_\mathrm{t} = \mathbf{s}, \mathbf{A}_\mathrm{t}) d\Pi(\mathbf{a} \mid \mathbf{S}_\mathrm{t} = \mathbf{s}).$$

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 (9) for episodes of arbitrary length by mathematical induction and conclude the proof.

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

$$\begin{aligned}
\nabla v_\pi(\mathbf{s}) &= \int q_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \nabla q_\pi(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), \\
&= \int q_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \left(\nabla \int \mathrm{r} d\mathbb{P}(\mathrm{r} \mid \mathbf{s}, \mathbf{a})\right) d\Pi(\mathbf{a} \mid \mathbf{s}), \\
&= \int q_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}), \\
&= \int h^+_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}) + \int \Big(\mathbb{1}\big(q_\pi > v_\pi\big) v_\pi(\mathbf{s}) + \mathbb{1}\big(q_\pi \leq v_\pi\big) q_\pi(\mathbf{s}, \mathbf{a})\Big) \nabla d\Pi(\mathbf{a} \mid \mathbf{s}).
\end{aligned} \tag{13}$$

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,

$$\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \pi(\mathbf{a} \mid \mathbf{s}, \theta_1), \\ \vdots \\ \frac{\partial}{\partial \theta_k} \pi(\mathbf{a} \mid \mathbf{s}, \theta_k) \end{pmatrix}, \tag{14}$$

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,

$$\begin{aligned}
\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) &= \tilde{\beta}\big(\nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})\big), \\
&= \int_\gamma \nabla \pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta}) d\mathbf{x}, \\
&= \int_\gamma \frac{\partial}{\partial \theta_1} \pi(\mathbf{a} \mid \mathbf{s}, \theta_1) d\theta_1 + \cdots + \frac{\partial}{\partial \theta_k} \pi(\mathbf{a} \mid \mathbf{s}, \theta_k) d\theta_k,
\end{aligned} \tag{15}$$

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,

$$\begin{aligned}
v^*_\pi(\mathbf{s}) &= \tilde{\beta}\bigg(\int h^+_\pi(\mathbf{s}, \mathbf{a}) \nabla d\Pi(\mathbf{a} \mid \mathbf{s})\bigg), \\
&= \int h^+_\pi(\mathbf{s}, \mathbf{a}) \tilde{\beta}\big(\nabla d\Pi(\mathbf{a} \mid \mathbf{s})\big), && \text{linearity} \\
&= \int h^+_\pi(\mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), && \text{Equation (15)} \\
&\leq \iint \Big(\mathrm{r} + \big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\big)^+\Big) d\mathbb{P}(\mathbf{s}', \mathrm{r} \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), && \text{Lemma C.2} \\
&= v_\pi(\mathbf{s}) + \iint \big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\big)^+ d\mathbb{P}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{s}), && \text{—T— = 1}
\end{aligned}$$

which concludes the proof for episodes of length 1.

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

$$\begin{aligned}
\nabla v_\pi(\mathbf{s}) &= \int q_\pi(\mathbf{s},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \int \nabla q_\pi(\mathbf{s},\mathbf{a}) d\Pi(\mathbf{a}\mid\mathbf{s}),\\
&= \int q_\pi(\mathbf{s},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \iiint q_\pi(\mathbf{s}',\mathbf{a}')\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s})\\
&\quad + \iiint \left(\nabla \int \mathrm{r}' d\mathbb{P}(\mathrm{r}'\mid\mathbf{s}',\mathbf{a}')\right) d\Pi(\mathbf{a}'\mid\mathbf{s}'),\\
&= \int q_\pi(\mathbf{s},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \iiint q_\pi(\mathbf{s}',\mathbf{a}')\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s}),\\
&= \int h^+_\pi(\mathbf{s},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \iiint h^+_\pi(\mathbf{s}',\mathbf{a}')\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s})\\
&\quad + \int \Big(\mathbb{1}\big(q_\pi > v_\pi\big) v_\pi(\mathbf{s}) + \mathbb{1}\big(q_\pi \le v_\pi\big) q_\pi(\mathbf{s},\mathbf{a})\Big)\nabla d\Pi(\mathbf{a}\mid\mathbf{s})\\
&\quad + \iiint \Big(\mathbb{1}\big(q_\pi > v_\pi\big) v_\pi(\mathbf{s}') + \mathbb{1}\big(q_\pi \le v_\pi\big) q_\pi(\mathbf{s}',\mathbf{a}')\Big)\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s}).
\end{aligned}$$

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:

$$\begin{aligned}
v^*_\pi(\mathbf{s}) &= \tilde{\beta}\Big(\int h^+_\pi(\mathbf{s},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \iiint h^+_\pi(\mathbf{s}',\mathbf{a}')\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s})\Big),\\
&= \int h^+_\pi(\mathbf{s},\mathbf{a})\tilde{\beta}\Big(\nabla d\Pi(\mathbf{a}\mid\mathbf{s})\Big) + \iiint h^+_\pi(\mathbf{s}',\mathbf{a}')\tilde{\beta}\Big(\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}')\Big) d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s}), && \text{linearity}\\
&= \int h^+_\pi(\mathbf{s},\mathbf{a}) d\Pi(\mathbf{a}\mid\mathbf{s}) + \iiint h^+_\pi(\mathbf{s}',\mathbf{a}') d\Pi(\mathbf{a}'\mid\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s}), && \text{Equation (15)}\\
&\le \iint \mathrm{r} d\mathbb{P}(\mathrm{r}\mid\mathbf{s},\mathbf{a}) d\Pi(\mathbf{a}\mid\mathbf{s}) + \iint \big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\big)^+ d\mathbb{P}(\mathbf{s}'\mid\mathbf{s},\mathbf{a}) d\Pi(\mathbf{a}\mid\mathbf{s})\\
&\quad + \iiint h^+_\pi(\mathbf{s}',\mathbf{a}') d\Pi(\mathbf{a}'\mid\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s}), && \text{Lemma C.2}\\
&\le \iint \mathrm{r} d\mathbb{P}(\mathrm{r}\mid\mathbf{s},\mathbf{a}) d\Pi(\mathbf{a}\mid\mathbf{s}) + \iint \big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\big)^+ d\mathbb{P}(\mathbf{s}'\mid\mathbf{s},\mathbf{a}) d\Pi(\mathbf{a}\mid\mathbf{s})\\
&\quad + \iint \gamma v_\pi(\mathbf{s}') d\mathbb{P}(\mathbf{s}'\mid\mathbf{a},\mathbf{s}) d\Pi(\mathbf{a}\mid\mathbf{s}), && \text{Lemma C.3}\\
&= v_\pi(\mathbf{s}) + \iint \big(\gamma v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})\big)^+ d\mathbb{P}(\mathbf{s}'\mid\mathbf{s},\mathbf{a}) d\Pi(\mathbf{a}\mid\mathbf{s}). && \text{rearranging terms}
\end{aligned}$$

□

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

Proof.

$$\nabla v_\pi(\mathbf{s}) = \nabla\Big[\int q_\pi(\mathbf{s},\mathbf{a})d\Pi(\mathbf{a}\mid\mathbf{s})\Big], \tag{19a}$$

$$= \int q_\pi(\mathbf{s},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \int \nabla q_\pi(\mathbf{s},\mathbf{a})d\Pi(\mathbf{a}\mid\mathbf{s}), \tag{19b}$$

$$= \int \Big(h_\pi^+(\mathbf{s},\mathbf{a}) + \mathbb{1}\big(q_\pi > v_\pi\big)v_\pi(\mathbf{s}) + \mathbb{1}\big(q_\pi \le v_\pi\big)q_\pi(\mathbf{s},\mathbf{a})\Big)\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \int \nabla q_\pi(\mathbf{s},\mathbf{a})d\Pi(\mathbf{a}\mid\mathbf{s}), \tag{19c}$$

$$= \int \Big(h_\pi^+(\mathbf{s},\mathbf{a}) + \mathbb{1}\big(q_\pi > v_\pi\big)v_\pi(\mathbf{s}) + \mathbb{1}\big(q_\pi \le v_\pi\big)q_\pi(\mathbf{s},\mathbf{a})\Big)\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \int \nabla\Big[\int \big(\mathrm{r} + \gamma v_\pi(\mathbf{s}')\big)d\mathbb{P}(\mathbf{s}',\mathrm{r}\mid\mathbf{s},\mathbf{a})\Big]d\Pi(\mathbf{a}\mid\mathbf{s}), \tag{19d}$$

$$= \int \Big(h_\pi^+(\mathbf{s},\mathbf{a}) + \mathbb{1}\big(q_\pi > v_\pi\big)v_\pi(\mathbf{s}) + \mathbb{1}\big(q_\pi \le v_\pi\big)q_\pi(\mathbf{s},\mathbf{a})\Big)\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \gamma\iint \nabla v_\pi(\mathbf{s}')d\mathbb{P}(\mathbf{s}'\mid\mathbf{s},\mathbf{a})d\Pi(\mathbf{a}\mid\mathbf{s}), \tag{19e}$$

$$= \int \Big(h_\pi^+(\mathbf{s},\mathbf{a}) + \mathbb{1}\big(q_\pi > v_\pi\big)v_\pi(\mathbf{s}) + \mathbb{1}\big(q_\pi \le v_\pi\big)q_\pi(\mathbf{s},\mathbf{a})\Big)\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \gamma\iint \Big[\int q_\pi(\mathbf{s}',\mathbf{a}')\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}') + \gamma\int \nabla v_\pi(\mathbf{s}'')d\mathbb{P}(\mathbf{s}''\mid\mathbf{s}',\mathbf{a}')d\Pi(\mathbf{a}'\mid\mathbf{s}')\Big]d\mathbb{P}(\mathbf{s}'\mid\mathbf{s},\mathbf{a})d\Pi(\mathbf{a}\mid\mathbf{s}), \tag{19f}$$

$$= \int \Big(h_\pi^+(\mathbf{s},\mathbf{a}) + \mathbb{1}\big(q_\pi > v_\pi\big)v_\pi(\mathbf{s}) + \mathbb{1}\big(q_\pi \le v_\pi\big)q_\pi(\mathbf{s},\mathbf{a})\Big)\nabla d\Pi(\mathbf{a}\mid\mathbf{s}) + \gamma\iint \Big[\int \Big(h_\pi^+(\mathbf{s}',\mathbf{a}') + \mathbb{1}\big(q_\pi > v_\pi\big)v_\pi(\mathbf{s}') + \mathbb{1}\big(q_\pi \le v_\pi\big)q_\pi(\mathbf{s}',\mathbf{a}')\Big)\nabla d\Pi(\mathbf{a}'\mid\mathbf{s}') + \gamma\int \nabla v_\pi(\mathbf{s}'')d\mathbb{P}(\mathbf{s}''\mid\mathbf{s}',\mathbf{a}')d\Pi(\mathbf{a}'\mid\mathbf{s}')\Big]d\mathbb{P}(\mathbf{s}'\mid\mathbf{s},\mathbf{a})d\Pi(\mathbf{a}\mid\mathbf{s}), \tag{19g}$$

$$= \int_{\mathcal{S}}\sum_{k=0}^{\infty}\Big[\gamma^k\int_{\mathcal{A}} h_\pi^+(\mathbf{x},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{x})\Big]d\mathbb{P}(\mathbf{s}\to\mathbf{x};k,\pi) + \int_{\mathcal{S}}\sum_{k=0}^{\infty}\Big[\gamma^k\int_{\mathcal{A}}\mathbb{1}\big(q_\pi(\mathbf{x},\mathbf{a}) > v_\pi(\mathbf{x})\big)v_\pi(\mathbf{x})\nabla d\Pi(\mathbf{a}\mid\mathbf{x})\Big]d\mathbb{P}(\mathbf{s}\to\mathbf{x};k,\pi) + \int_{\mathcal{S}}\sum_{k=0}^{\infty}\Big[\gamma^k\int_{\mathcal{A}}\mathbb{1}\big(q_\pi(\mathbf{x},\mathbf{a}) \le v_\pi(\mathbf{x})\big)q_\pi(\mathbf{x},\mathbf{a})\nabla d\Pi(\mathbf{a}\mid\mathbf{x})\Big]d\mathbb{P}(\mathbf{s}\to\mathbf{x};k,\pi) \tag{19h}$$

□

Lemma C.2.

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

Proof.

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

□

Lemma C.3. *When, without loss of generality, rewards,* R_t*, are assumed to be non-negative:*

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

Proof.

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

□

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

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

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_\pi \left[\int_{\mathcal{A}} \left(q_\pi(\mathbf{S}_\mathrm{t}, \mathbf{a}) - \int_{\mathcal{A}} \pi(\mathbf{a}' \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta}) q_\pi(\mathbf{S}_\mathrm{t}, \mathbf{a}') d\mathbf{a}' \right)^+ \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta}) d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[\int_{\mathcal{A}} \left(q_\pi(\mathbf{S}_\mathrm{t}, \mathbf{a}) - v_\pi(\mathbf{S}_\mathrm{t}) \right)^+ \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta}) d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[\int_{\mathcal{A}} h_{\pi}^{+}(\mathbf{S}_\mathrm{t}, \mathbf{a}) \nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta}) d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[\int_{\mathcal{A}} \pi(\mathbf{a} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta}) h_{\pi}^{+}(\mathbf{S}_\mathrm{t}, \mathbf{a}) \frac{\nabla_{\boldsymbol{\theta}} \pi(\mathbf{a} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta})}{\pi(\mathbf{a} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta})} d\mathbf{a} \right] \\
&= \mathbb{E}_\pi \left[h_{\pi}^{+}(\mathbf{S}_\mathrm{t}, \mathbf{A}_\mathrm{t}) \frac{\nabla_{\boldsymbol{\theta}} \pi(\mathbf{A}_\mathrm{t} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta})}{\pi(\mathbf{A}_\mathrm{t} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta})} \right] \\
&= \mathbb{E}_\pi \left[h_{\pi}^{+}(\mathbf{S}_\mathrm{t}, \mathbf{A}_\mathrm{t}) \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{A}_\mathrm{t} \mid \mathbf{S}_\mathrm{t}, \boldsymbol{\theta}) \right]
\end{aligned}
$$

Lemma C.4.

$$
\mathit{ReLU}(a) <= |a|
$$

Proof.

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

□

D DISCUSSION

D.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, $\pi(\mathbf{a} \mid \mathbf{s}, \boldsymbol{\theta})$, the roles played by the Lipschitz assumption and the use of critic weight spectral normalization becomes clearer. When we do gradient ascent according to,

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

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,

$$C_\pi(\mathbf{s}) \leq \frac{1}{2} \iint |v_\pi(\mathbf{s}') - v_\pi(\mathbf{s})| \, dP(\mathbf{s}' \mid \mathbf{S}_\text{t} = \mathbf{s}, \mathbf{A}_\text{t} = \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{S}_\text{t} = \mathbf{s}).$$

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

$$C_\pi(\mathbf{s}) \leq \frac{1}{2} \iint K_\pi \, ||\mathbf{s}' - \mathbf{s}|| \, dP(\mathbf{s}' \mid \mathbf{S}_\text{t} = \mathbf{s}, \mathbf{A}_\text{t} = \mathbf{a}) d\Pi(\mathbf{a} \mid \mathbf{S}_\text{t} = \mathbf{s}),$$

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.

D.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 6 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.

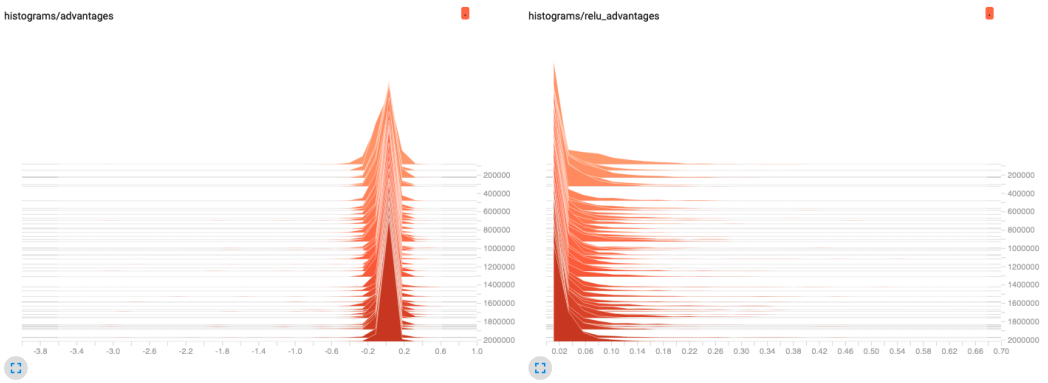


Figure 6: Comparing the histograms of estimated advantages (left) and ReLU'ed advantages (right).

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.3 Concerning Thompson Sampling and approximate Bayesian inference.

Approximate Bayesian inference over the parameters of the policy, θ, yields a distribution over those parameters, $p(\theta \mid \mathcal{D})$. Sampling a policy from this distribution, $\hat{\theta} \sim p(\theta \mid \mathcal{D})$, is as easy as sampling a dropout mask and then running a forward pass of the network, yielding the likelihood, $\pi(\mathbf{a} \mid \mathbf{s}, \hat{\theta})$. Then sampling an action is done by sampling an action from the sampled policy, $\mathbf{a} \sim \pi(\mathbf{a} \mid \mathbf{s}, \hat{\theta})$. This is precisely the procedure descibed by Thompson sampling. We outline this procedure in lines 5-6 of Algorithm 1.

We hypothesize that this is a better state-aware exploration method for two reasons. First, for less frequently visited states the diversity of the sampled paramters 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 that we have less experience of good actions, and less exploration in states where we know what actions lead to good expected returns. Second, this exploration is done around the mode of the policy distribution, so the model is less likely to explore actions that are far from the mode, which could be more likely to lead to failure.

E 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.

E.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

	Gymnasium MuJoCo				
Parameter	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.

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

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

E.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

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

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).

E.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.

F Additional Results

F.1 ProcGen

Figure 7 compares the ProcGen training and test curves of VSOP to PPO.

F.2 Ablation of Mechanisms

Figure 8 compares VSOP training curves to ablated variants.

F.3 Comparison to Baselines

Figure 9 compares VSOP training curves to baseline algorithms.

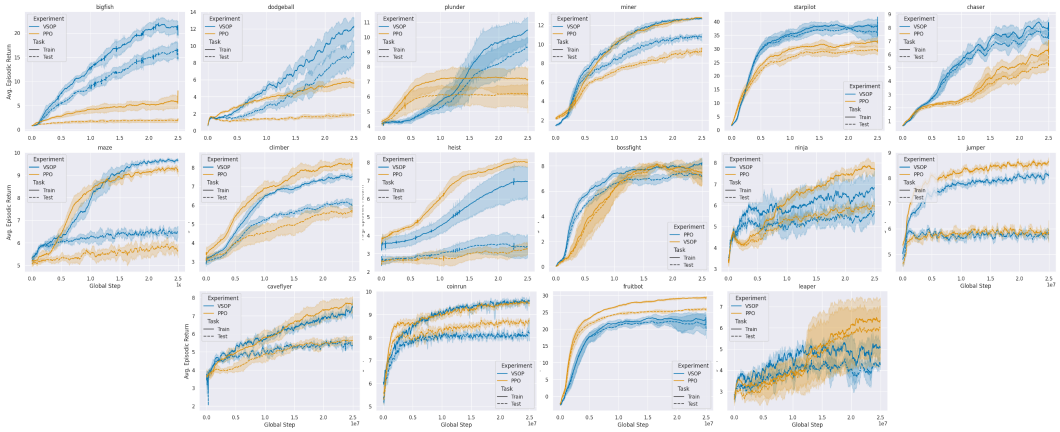
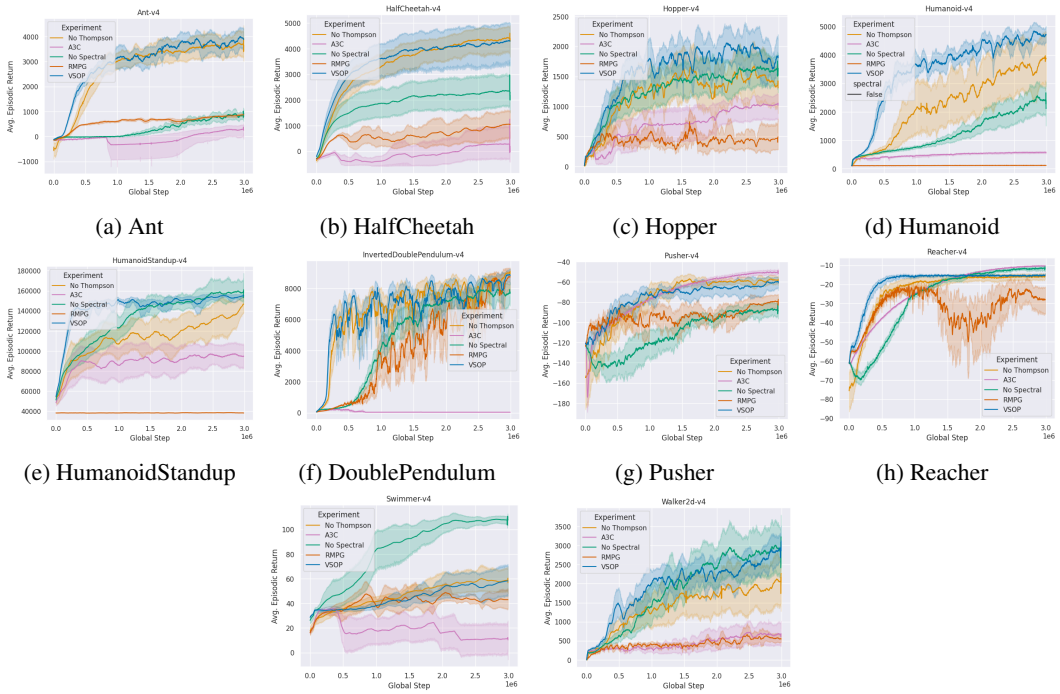


Figure 7: ProcGen training and test curves. We see significant improvement in test set performance on 8 environments, statistical equivalence on 5 environments, and VSOP trails PPO on just 3 environments.



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

(e) HumanoidStandup (f) DoublePendulum (g) Pusher (h) Reacher

(i) Swimmer (j) Walker2d

Figure 8: 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 proposed, optimized method. Orange lines (No Thompson) show VSOP without Thompson sampling. Green lines (No Spectral) show VSOP without spectral normalization. Pink lines (RMPG) show VSOP with "all actions". Red lines (A3C) show VSOP without restricting policy updates to positive advantages.

F.4 Spectral norm and Thompson sampling 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.

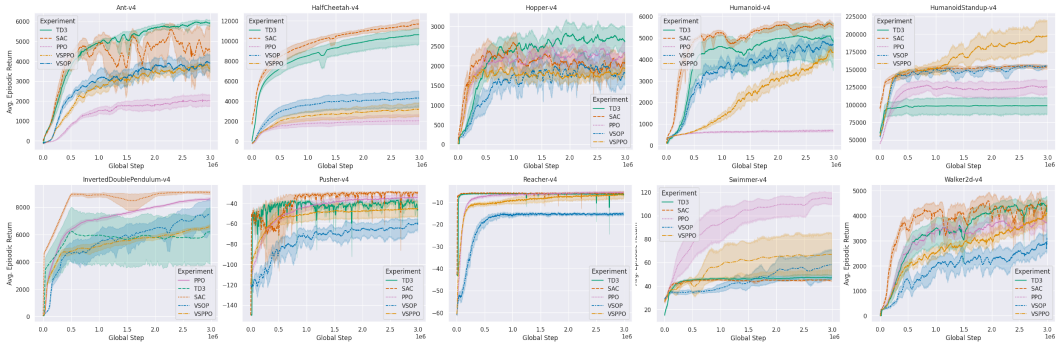


Figure 9: Gymnasium-MuJoCo. Comparing VSOP to baseline algorithms.

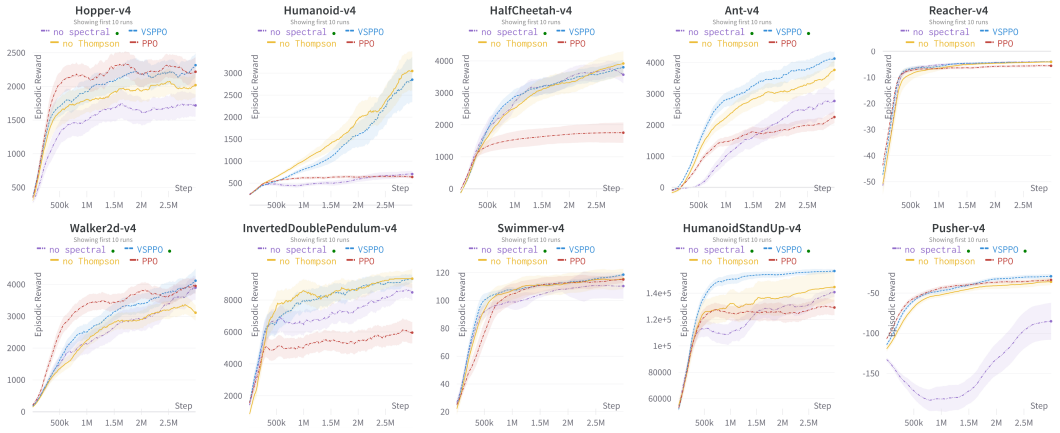


Figure 10: MuJoCo continuous control benchmark examining the effect of Thompson sampling and spectral normalization on PPO.

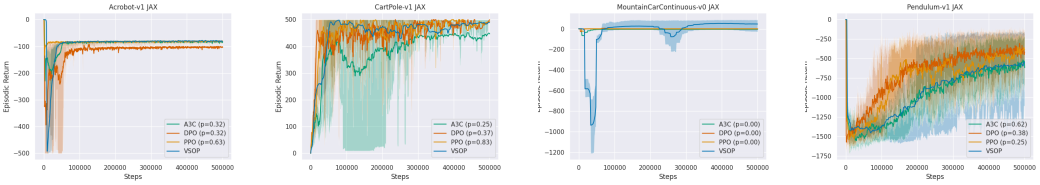
F.5 Gymnax Environments

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 E.2. Table 9 shows the overall ranking of each method. VSOP is competitive with DPO and improves over PPO and A3C.

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

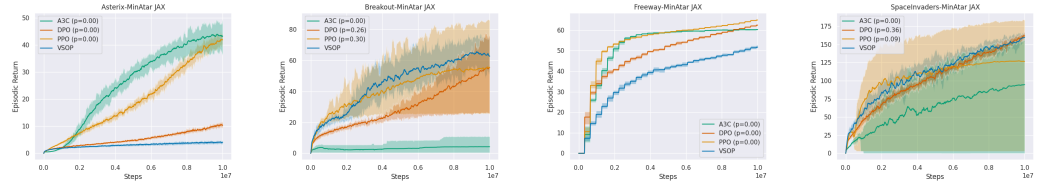
Figure 11 summarize the results for **Classic Control**. Performance of each method is in general statistically equal, but VSOP shows significant gain on MountainCar Continuous.



(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.

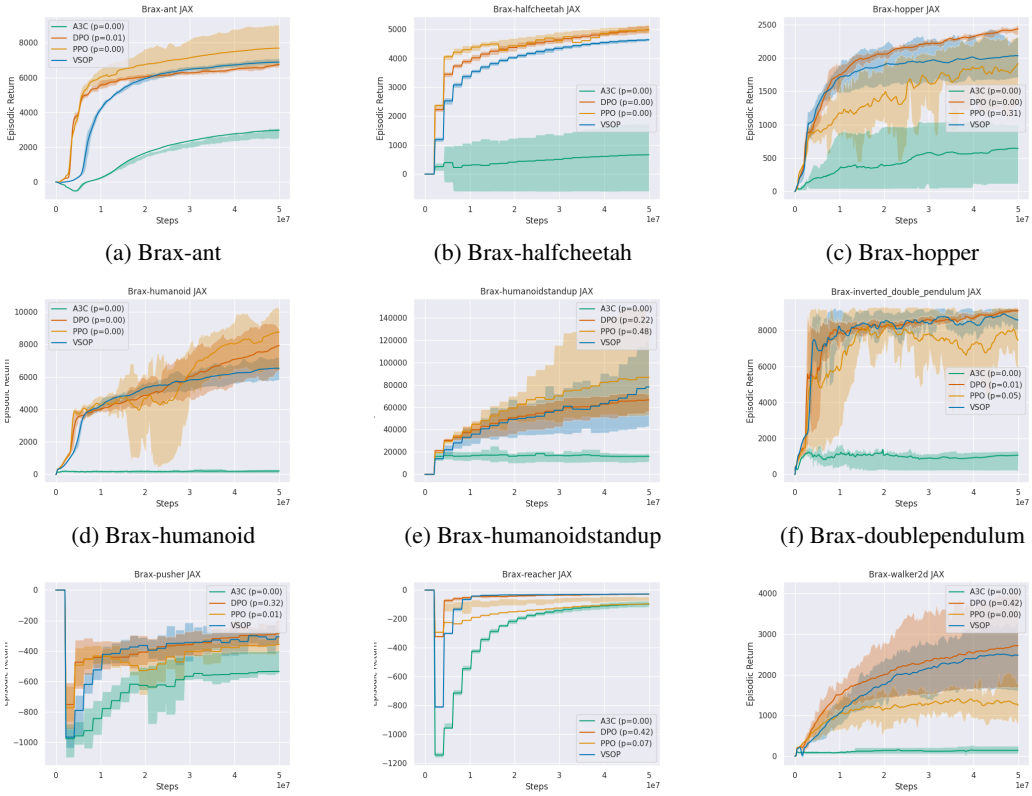
Figure 12 summarize the results for **MinAtar** (Bellemare et al., 2013; Young & Tian, 2019). VSOP shows significant improvement over PPO and A3C in Space Invaders. We see marginal improvement over PPO and DPO in Breakout, with significant improvement over A3C. VSOP trails the baselines significantly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



(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 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.



(a) Brax-ant (b) Brax-halfcheetah (c) Brax-hopper

(d) Brax-humanoid (e) Brax-humanoidstandup (f) Brax-doublependulum

(g) Brax-pusher (h) Brax-reacher (i) Brax-walker2d

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.