Policy Gradient with Tree Expansion

Gal Dalal [*1] **Assaf Hallak** [*1] **Gugan Thoppe** [2] **Shie Mannor** [1 3] **Gal Chechik** [1 4]

Abstract

Policy gradient methods are notorious for having a large variance and high sample complexity. To mitigate this, we introduce SoftTreeMax—a generalization of softmax that employs planning. In SoftTreeMax, we extend the traditional logits with the multi-step discounted cumulative reward, topped with the logits of future states. We analyze SoftTreeMax and explain how tree expansion helps to reduce its gradient variance. We prove that the variance depends on the chosen tree-expansion policy. Specifically, we show that the closer the induced transitions are to being state-independent, the stronger the variance decay. With approximate forward models, we prove that the resulting gradient bias diminishes with the approximation error while retaining the same variance reduction. Ours is the first result to bound the gradient bias for an approximate model. In a practical implementation of SoftTreeMax, we utilize a parallel GPU-based simulator for fast and efficient tree expansion. Using this implementation in Atari, we show that SoftTreeMax reduces the gradient variance by three orders of magnitude. This leads to better sample complexity and improved performance compared to distributed PPO.

1. Introduction

Policy Gradient (PG) methods (Sutton et al., 1999) for Reinforcement Learning (RL) are often the first choice for environments that allow numerous interactions at a fast pace (Schulman et al., 2017). Their success is attributed to several factors: they are easy to distribute to multiple workers, require no assumptions on the underlying value function, and have both on-policy and off-policy variants.

Despite these positive features, PG algorithms are also notoriously unstable due to the high variance of the gradients computed over entire trajectories (Liu et al., 2020; Xu et al., 2020). As a result, PG algorithms tend to be highly inefficient in terms of sample complexity. Several solutions have been proposed to mitigate the high variance issue, including baseline subtraction (Greensmith et al., 2004; Thomas & Brunskill, 2017; Wu et al., 2018), anchor-point averaging (Papini et al., 2018), and other variance reduction techniques (Zhang et al., 2021; Shen et al., 2019; Pham et al., 2020).

A second family of algorithms that achieved state-of-the-art results in several domains is based on planning. Planning is exercised primarily in the context of value-based RL and is usually implemented using a Tree Search (TS) (Silver et al., 2016; Schrittwieser et al., 2020). In this work, we combine PG with TS by introducing a parameterized differentiable policy that incorporates tree expansion. Namely, our SoftTreeMax policy replaces the standard policy logits of a state and action, with the expected value of trajectories that originate from these state and action. We consider two variants of SoftTreeMax, one for cumulative reward and one for exponentiated reward.

Unlike approaches that incorporate multi-step returns for value function estimation (e.g., n-step TD methods), our work explicitly integrates planning into the policy parameterization itself. This distinction is crucial—while n-step returns serve as a Monte Carlo estimation technique for the advantage function A in the gradient estimator $E[(\nabla \log \pi)A]$, our SoftTreeMax affects the policy π directly. This enables us to obtain fundamentally different variance reduction properties while keeping the policy gradient framework intact.

Combining TS and PG should be done with care given the biggest downside of PG—its high gradient variance. This raises questions that were ignored until this work: (i) How to design a PG method based on tree-expansion that is stable and performs well in practice? and (ii) How does the tree-expansion policy affect the PG variance? Here, we analyze SoftTreeMax, and provide a practical methodology to choose the expansion policy to minimize the resulting variance. Our main result shows that a desirable expansion policy is one, under which the induced transition probabilities are similar for each starting state. More generally, we show that the gradient variance of SoftTreeMax decays at a

[*]Equal contribution [1]NVIDIA Research [2]Indian Institute of Science [3]Technion University [4]Bar-Ilan University. Correspondence to: Gal dalal <gdalal@nvidia.com>, Assaf Hallak <ahallak@nvidia.com>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

rate of $|\lambda_2|^d$, where d is the depth of the tree and λ_2 is the second eigenvalue of the transition matrix induced by the tree expansion policy. This work is the first to prove such a relation between PG variance and tree expansion policy. In addition, we prove that with an approximate forward model, the bias of the gradient is bounded proportionally to the approximation error of the model.

To verify our results, we implemented a practical version of SoftTreeMax that exhaustively searches the entire tree and applies a neural network on its leaves. We test our algorithm on a parallelized Atari GPU simulator (Dalton et al., 2020). To enable a tractable deep search, up to depth eight, we also introduce a pruning technique that limits the width of the tree. We do so by sampling only the most promising nodes at each level. We integrate our SoftTreeMax GPU implementation into the popular PPO (Schulman et al., 2017) and compare it to the flat distributed variant of PPO. This allows us to demonstrate the potential benefit of utilizing learned models while isolating the fundamental properties of TS without added noise. In all tested Atari games, our results outperform the baseline and obtain up to 5x more reward. We further show in Section 6 that the associated gradient variance is smaller by three orders of magnitude in all games, demonstrating the relation between low gradient variance and high reward.

We summarize our key contributions. (i) We show how to combine two families of SoTA approaches: PG and TS by **introducing SoftTreeMax:** a novel parametric policy that generalizes softmax to planning. Specifically, we propose two variants based on cumulative and exponentiated rewards. (ii) We **prove that the gradient variance of SoftTreeMax in its two variants decays** with its tree depth. Our analysis sheds new light on the choice of tree expansion policy. It raises the question of optimality in terms of variance versus the traditional regret; e.g., in UCT (Kocsis & Szepesvári, 2006). (iii) We prove that with an approximate forward model, the **gradient bias is proportional to the approximation error**, while retaining the variance decay. This quantifies the accuracy required from a learned forward model. (iv) We **implement a differentiable deep version of SoftTreeMax** that employs a parallelized GPU tree expansion. We demonstrate how its gradient variance is reduced by three orders of magnitude over PPO while obtaining up to 5x reward.

2. Preliminaries

Let Δ_U denote simplex over the set U. Throughout, we consider a discounted Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \nu)$, where $\mathcal{S}$ is a finite state space of size S, $\mathcal{A}$ is a finite action space of size A, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ is the transition function, $\gamma \in (0, 1)$ is the discount factor, and $\nu \in \mathbb{R}^S$ is the initial state distribution. We denote the transition matrix starting from state s by $P_s \in [0, 1]^{A \times S}$, i.e., $[P_s]_{a,s'} = P(s'|a, s)$. Similarly, let $R_s = r(s, \cdot) \in \mathbb{R}^A$ denote the corresponding reward vector. Separately, let $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ be a stationary policy. Let P^π and R_π be the induced transition matrix and reward function, respectively, i.e., $P^\pi(s'|s) = \sum_a \pi(a|s) \Pr(s'|s, a)$ and $R_\pi(s) = \sum_a \pi(a|s) r(s, a)$. Denote the stationary distribution of P^π by $\mu_\pi \in \mathbb{R}^S$ s.t. $\mu_\pi^\top P^\pi = P^\pi$, and the discounted state visitation frequency by d_π so that $d_\pi^\top = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \nu^\top (P^\pi)^t$. Also, let $V^\pi \in \mathbb{R}^S$ be the value function of π defined by $V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, \pi(s_t)\right) \mid s_0 = s\right]$, and let $Q^\pi \in \mathbb{R}^{S \times A}$ be the Q-function such that $Q^\pi(s, a) = \mathbb{E}^\pi \left[r(s, a) + \gamma V^\pi(s')\right]$. Our goal is to find an optimal policy $\pi^\star$ such that $V^\star(s) \equiv V^{\pi^\star}(s) = \max_\pi V^\pi(s)$, $\forall s \in \mathcal{S}$.

For the analysis in Section 4, we introduce the following notation. Denote by $\Theta \in \mathbb{R}^S$ the vector representation of $\theta(s)$ $\forall s \in \mathcal{S}$. For a vector u, denote by $\exp(u)$ the coordinate-wise exponent of u and by $D(u)$ the diagonal square matrix with u in its diagonal. For a matrix A, denote its i-th eigenvalue by $\lambda_i(A)$. Denote the k-dimensional identity matrix and all-ones vector by I_k and $\mathbf{1}_k$, respectively. Also, denote the trace operator by Tr . Finally, we treat all vectors as column vectors.

2.1. Policy Gradient

PG schemes seek to maximize the cumulative reward as a function of the policy $\pi_\theta(a|s)$ by performing gradient steps on θ. The celebrated Policy Gradient Theorem (Sutton et al., 1999) states that

$$\frac{\partial}{\partial \theta} \nu^\top V^{\pi_\theta} = \mathbb{E}_{s \sim d_{\pi_\theta}, a \sim \pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)\right],$$

where ν and $d_{\pi_\theta}^\top$ are as defined above. The variance of the gradient is thus

$$\mathrm{Var}_{s \sim d_{\pi_\theta}, a \sim \pi_\theta(\cdot|s)} \left(\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)\right). \quad (1)$$

In the notation above, we denote the variance of a vector random variable X by

$$\mathrm{Var}_x\left(X\right) = \mathrm{Tr}\left[\mathbb{E}_x\left[\left(X - \mathbb{E}_x X\right)^\top \left(X - \mathbb{E}_x X\right)\right]\right],$$

similarly as in (Greensmith et al., 2004). From now on, we drop the subscript from Var in (1) for brevity. When the action space is discrete, a commonly used parameterized policy is softmax: $\pi_\theta(a|s) \propto \exp\left(\theta(s, a)\right)$, where $\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a state-action parameterization.

3. SoftTreeMax: Exponent of trajectories

We introduce a new family of policies called SoftTreeMax, which are a model-based generalization of the popular softmax. We propose two variants: Cumulative

(C-SoftTreeMax) and Exponentiated (E-SoftTreeMax). In both variants, we replace the generic softmax logits $\theta(s,a)$ with the score of a trajectory of horizon d starting from (s,a), generated by applying a behavior policy π_b. In C-SoftTreeMax, we exponentiate the expectation of the logits. In E-SoftTreeMax, we first exponentiate the logits and then only compute their expectation.

Logits. We define the SoftTreeMax logit $\ell_{s,a}(d;\theta)$ to be the random variable depicting the score of a trajectory of horizon d starting from (s,a) and following the policy π_b:

$$\ell_{s,a}(d;\theta) = \gamma^{-d}\left[\sum_{t=0}^{d-1}\gamma^t r_t + \gamma^d\theta(s_d)\right]. \tag{2}$$

In the above expression, note that $s_0 = s,\ a_0 = a,\ a_t \sim \pi_b(\cdot|s_t)\ \forall t \geq 1$, and $r_t \equiv r(s_t,a_t)$. For brevity of the analysis, we let the parametric score θ in (2) be state-based, similarly to a value function. Instead, one could use a state-action input analogous to a Q-function. Thus, SoftTreeMax can be integrated into the two types of implementation of RL algorithms in standard packages. Lastly, the preceding γ^{-d} scales the θ parametrization to correspond to its softmax counerpart.

C-SoftTreeMax. Given an inverse temperature parameter β, we let C-SoftTreeMax be

$$\pi^{\text{C}}_{d,\theta}(a|s) \propto \exp\left[\beta\mathbb{E}^{\pi_b}\ell_{s,a}(d;\theta)\right]. \tag{3}$$

C-SoftTreeMax gives higher weight to actions that result in higher expected returns. While standard softmax relies entirely on parametrization θ, C-SoftTreeMax also interpolates a Monte-Carlo portion of the reward.

E-SoftTreeMax. The second operator we propose is E-SoftTreeMax:

$$\pi^{\text{E}}_{d,\theta}(a|s) \propto \mathbb{E}^{\pi_b}\exp\left[(\beta\ell_{s,a}(d;\theta))\right]; \tag{4}$$

here, the expectation is taken outside the exponent. This objective corresponds to the exponentiated reward objective which is often used for risk-sensitive RL (Howard & Matheson, 1972; Fei et al., 2021; Noorani & Baras, 2021). The common risk-sensitive objective is of the form $\log\mathbb{E}[\exp(\delta R)]$, where δ is the risk parameter and R is the cumulative reward. Similarly to that literature, the exponent in (4) emphasizes the most promising trajectories.

SoftTreeMax properties. SoftTreeMax is a natural model-based generalization of softmax. For $d=0$, both variants above coincide since (2) becomes deterministic. In that case, for a state-action parametrization, they reduce to standard softmax. When $\beta \to 0$, both variants again coincide and sample actions uniformly (exploration). When $\beta \to \infty$, the policies become deterministic and greedily optimize for the best trajectory (exploitation). For C-SoftTreeMax, the best trajectory is defined in expectation, while for E-SoftTreeMax it is defined in terms of the best sample path.

SoftTreeMax behavior policy selection. The choice of behavior policy π_b plays a crucial role in the performance of SoftTreeMax. As we show in Section 4, the gradient variance is minimized when the transitions induced by π_b result in similar distributions across states, which is achieved when the second eigenvalue of P^{π_b} is small. Without specific assumptions on the MDP, a uniform policy that smoothens transition probabilities across all actions serves as a reasonable approximation to this goal. In practice, this leads to better mixing properties in the associated Markov chain.

SoftTreeMax convergence. Under regularity conditions, for any parametric policy, PG converges to local optima (Bhatnagar et al., 2009), and thus also SoftTreeMax. For softmax PG, asymptotic (Agarwal et al., 2021) and rate results (Mei et al., 2020b) were recently obtained, by showing that the gradient is strictly positive everywhere (Mei et al., 2020b, Lemmas 8-9). We conjecture that SoftTreeMax satisfies the same property, being a generalization of softmax, but formally proving it is subject to future work.

SoftTreeMax gradient. The two variants of SoftTreeMax involve an expectation taken over S^d many trajectories from the root state s and weighted according to their probability. Thus, during the PG training process, the gradient $\nabla_\theta \log \pi_\theta$ is calculated using a weighted sum of gradients over all reachable states starting from s. Our method exploits the exponential number of trajectories to reduce the variance while improving performance. Indeed, in the next section we prove that the gradient variance of SoftTreeMax decays exponentially fast as a function of the behavior policy π_b and trajectory length d. In the experiments in Section 6, we also show how the practical version of SoftTreeMax achieves a significant reduction in the noise of the PG process and leads to faster convergence and higher reward.

4. Theoretical Analysis

In this section, we first bound the variance of PG when using the SoftTreeMax policy. Later, we discuss how the gradient bias resulting due to approximate forward models diminishes as a function of the approximation error, while retaining the same variance decay.

We show that the variance decreases with the tree depth, and the rate is determined by the second eigenvalue of the transition kernel induced by π_b. Specifically, we bound the same expression for variance as appears in (Greensmith et al., 2004, Sec. 3.5) and (Wu et al., 2018, Sec. A, Eq. (21)). Other types of analysis could instead have focused on the estimation aspect in the context of sampling (Zhang et al., 2021; Shen et al., 2019; Pham et al., 2020). Indeed, in

our implementation in Section 5, we manage to avoid sampling and directly compute the expectations in Eqs. (3) and (4). As we show later, we do so by leveraging efficient parallel simulation on the GPU in feasible run-time. In our application, due to the nature of the finite action space and quasi-deterministic Atari dynamics (Bellemare et al., 2013), our expectation estimator is noiseless. We encourage future work to account for the finite-sample variance component. We defer all the proofs to Appendix A.

We begin with a general variance bound that holds for *any* parametric policy.

Lemma 4.1 (Bound on the policy gradient variance)**.** *Let* $\nabla_\theta \log \pi_\theta(\cdot|s) \in \mathbb{R}^{A\times \dim(\theta)}$ *be a matrix whose* a*-th row is* $\nabla_\theta \log \pi_\theta(a|s)^\top$*. For any parametric policy* π_θ *and function* $Q^{\pi_\theta} : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$,

$$\mathrm{Var}\left(\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right) \leq \max_{s,a}\left[Q^{\pi_\theta}(s,a)\right]^2 \max_s \|\nabla_\theta \log \pi_\theta(\cdot|s)\|_F^2.$$

Hence, to bound (1), it is sufficient to bound the Frobenius norm $\|\nabla_\theta \log \pi_\theta(\cdot|s)\|_F$ for any s.

Note that SoftTreeMax does not reduce the gradient uniformly, which would have been equivalent to a trivial change in the learning rate. While the gradient norm shrinks, the gradient itself scales differently along the different coordinates. This scaling occurs along different eigenvectors, as a function of problem parameters (P, θ) and our choice of behavior policy (π_b), as can be seen in the proof of the upcoming Theorem 4.4. This allows SoftTreeMax to learn a good "shrinkage" that, while reducing the overall gradient, still updates the policy quickly enough. This reduction in norm and variance resembles the idea of gradient clipping (Zhang et al., 2019), where the gradient is scaled to reduce its variance, thus increasing stability and improving overall performance.

A common assumption in the RL literature (Szepesvári, 2010) that we adopt for the remainder of the section is that the transition matrix P^{π_b}, induced by the behavior policy π_b, is irreducible and aperiodic. Consequently, its second highest eigenvalue satisfies $|\lambda_2(P^{\pi_b})| < 1$.

From now on, we divide the variance results for the two variants of SoftTreeMax into two subsections. For C-SoftTreeMax, the analysis is simpler and we provide an exact bound. The case of E-SoftTreeMax is more involved and we provide for it a more general result. In both cases, we show that the variance decays exponentially with the planning horizon.

4.1. Variance of C-SoftTreeMax

We express C-SoftTreeMax in vector form as follows.

Lemma 4.2 (Vector form of C-SoftTreeMax)**.** *For* $d \geq 1$, (3) *is given by*

$$\pi^C_{d,\theta}(\cdot|s) = \frac{\exp\left[\beta\left(C_{s,d} + P_s\left(P^{\pi_b}\right)^{d-1}\Theta\right)\right]}{\boldsymbol{1}_A^\top \exp\left[\beta\left(C_{s,d} + P_s\left(P^{\pi_b}\right)^{d-1}\Theta\right)\right]}, \quad (5)$$

where

$$C_{s,d} = \gamma^{-d} R_s + P_s\left[\sum_{h=1}^{d-1}\gamma^{h-d}\left(P^{\pi_b}\right)^{h-1}\right] R_{\pi_b}.$$

The vector $C_{s,d} \in \mathbb{R}^A$ represents the cumulative discounted reward in expectation along the trajectory of horizon d. This trajectory starts at state s, involves an initial reward dictated by R_s and an initial transition as per P_s. Thereafter, it involves rewards and transitions specified by R_{π_b} and P^{π_b}, respectively. Once the trajectory reaches depth d, the score function $\theta(s_d)$ is applied,.

Lemma 4.3 (Gradient of C-SoftTreeMax)**.** *The C-SoftTreeMax gradient is given by*

$$\nabla_\theta \log \pi^C_{d,\theta} = \beta\left[I_A - \boldsymbol{1}_A(\pi^C_{d,\theta})^\top\right] P_s\left(P^{\pi_b}\right)^{d-1},$$

in $\mathbb{R}^{A\times S}$*, where for brevity, we drop the* s *index in the policy above, i.e.,* $\pi^C_{d,\theta} \equiv \pi^C_{d,\theta}(\cdot|s)$.

We are now ready to present our first main result:

Theorem 4.4 (Variance decay of C-SoftTreeMax)**.** *For every* $Q : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$*, the C-SoftTreeMax policy gradient variance is bounded by*

$$\mathrm{Var}\left(\nabla_\theta \log \pi^C_{d,\theta}(a|s) Q(s,a)\right) \leq 2\frac{A^2 S^2 \beta^2}{(1-\gamma)^2}|\lambda_2(P^{\pi_b})|^{2(d-1)}.$$

We provide the full proof in Appendix A.4, and briefly outline its essence here.

Proof outline. Lemma 4.1 allows us to bound the variance using a direct bound on the gradient norm. The gradient is given in Lemma 4.3 as a product of three matrices, which we now study from right to left. The matrix P^{π_b} is a row-stochastic matrix. Because the associated Markov chain is irreducible and aperiodic, it has a unique stationary distribution. This implies that P^{π_b} has one and only one eigenvalue equal to 1; all others have magnitude strictly less than 1. Let us suppose that all these other eigenvalues have multiplicity 1 (the general case with repeated eigenvalues can be handled via Jordan decompositions as in (Pelletier, 1998, Lemma1)). Then, P^{π_b} has the spectral decomposition $P^{\pi_b} = \mathbf{1}_S \mu_{\pi_b}^\top + \sum_{i=2}^S \lambda_i v_i u_i^\top$, where λ_i is the i-th

eigenvalue of P^{π_b} (ordered in descending order according to their magnitude) and u_i and v_i are the corresponding left and right eigenvectors, respectively, and therefore $(P^{\pi_b})^{d-1} = \mathbf{1}_S \mu_{\pi_b}^\top + \sum_{i=2}^{S} \lambda_i^{d-1} v_i u_i^\top$. The second matrix in the gradient relation in Lemma 4.3, P_s, is a rectangular transition matrix that translates the vector of all ones from dimension S to A : $P_s \mathbf{1}_S = \mathbf{1}_A$. Lastly, the first matrix $\left[I_A - \mathbf{1}_A(\pi^{\mathrm{C}}_{d,\theta})^\top\right]$ is a projection whose null-space includes the vector $\mathbf{1}_A$, i.e., $\left[I_A - \mathbf{1}_A(\pi^{\mathrm{C}}_{d,\theta})^\top\right]\mathbf{1}_A = 0$. Combining the three properties above when multiplying the three matrices of the gradient, it is easy to see that the first term in the expression for $(P^{\pi_b})^{d-1}$ gets canceled, and we are left with bounded summands scaled by $\lambda_i(P^{\pi_b})^{d-1}$. Recalling that $|\lambda_i(P^{\pi_b})| < 1$ and that $|\lambda_2| \geq |\lambda_3| \geq \dots$ for $i = 2, \dots, S$, we obtain the desired result. □

It's important to note that SoftTreeMax does not reduce the gradient uniformly, which would be equivalent to simply decreasing the learning rate. Rather, the gradient is scaled differently along different eigenvectors, with scaling factors that depend on the MDP structure and behavior policy π_b. This non-uniform scaling allows the policy to continue learning effectively while reducing harmful variance. The empirical results in Section 6 demonstrate that this variance reduction leads to faster, more stable convergence rather than slowing it down.

Theorem 4.4 guarantees that the variance of the gradient decays with d. More importantly, it also provides a novel insight for choosing the behavior policy π_b as the policy that minimizes the absolute second eigenvalue of the P^{π_b}. Indeed, the second eigenvalue of a Markov chain relates to its connectivity and its rate of convergence to the stationary distribution (Levin & Peres, 2017).

Optimal variance decay. For the strongest reduction in variance, the behavior policy π_b should be chosen to achieve an induced Markov chain whose transitions are state-independent. In that case, P^{π_b} is a rank one matrix of the form $\mathbf{1}_S \mu_{\pi_b}^\top$, and $\lambda_2(P^{\pi_b}) = 0$. Then, $\mathrm{Var}\left(\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right) = 0$. Naturally, this can only be done for pathological MDPs; see Appendix C.1 for a more detailed discussion. Nevertheless, as we show in Section 5, we choose our tree expansion policy to reduce the variance as best as possible.

Worst-case variance decay. In contrast, and somewhat surprisingly, when π_b is chosen so that the dynamics is deterministic, there is no guarantee that it will decay exponentially fast. For example, if P^{π_b} is a permutation matrix, then $\lambda_2(P^{\pi_b}) = 1$, and advancing the tree amounts to only updating the gradient of one state for every action, as in the basic softmax.

4.2. Variance of E-SoftTreeMax

The proof of the variance bound for E-SoftTreeMax is similar to that of C-SoftTreeMax, but more involved. It also requires the assumption that the reward depends only on the state, i.e. $r(s,a) \equiv r(s)$. This is indeed the case in most standard RL environments such as Atari and Mujoco.

Lemma 4.5 (Vector form of E-SoftTreeMax)**.** *For $d \geq 1$, (4) is given by*

$$\pi^{E}_{d,\theta}(\cdot|s) = \frac{E_{s,d}\exp(\beta\Theta)}{1_A^\top E_{s,d}\exp(\beta\Theta)}, \tag{6}$$

where

$$E_{s,d} = P_s \prod_{h=1}^{d-1} \left(D\left(\exp(\beta\gamma^{h-d}R)\right) P^{\pi_b}\right).$$

The vector R above is the S-dimensional vector whose s-th coordinate is $r(s)$.

The matrix $E_{s,d} \in \mathbb{R}^{A\times S}$ has a similar role to $C_{s,d}$ from (5), but it represents the exponentiated cumulative discounted reward. Accordingly, it is a product of d matrices as opposed to a sum. It captures the expected reward sequence starting from s and then iteratively following P^{π_b}. After d steps, we apply the score function on the last state as in (6).

Lemma 4.6 (Gradient of E-SoftTreeMax)**.** *The E-SoftTreeMax gradient is given by*

$$\nabla_\theta \log \pi^{E}_{d,\theta} = \beta\left[I_A - \mathbf{1}_A(\pi^{E}_{d,\theta})^\top\right] \times \frac{D\left(\pi^{E}_{d,\theta}\right)^{-1} E_{s,d} D(\exp(\beta\Theta))}{\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta)} \in \mathbb{R}^{A\times S},$$

where for brevity, we drop the s index in the policy above, i.e., $\pi^{E}_{d,\theta} \equiv \pi^{E}_{d,\theta}(\cdot|s)$.

This gradient structure is harder to handle than that of C-SoftTreeMax in Lemma 4.3, but here we also can bound the decay of the variance nonetheless.

Theorem 4.7 (Variance decay of E-SoftTreeMax)**.** *There exists $\alpha \in (0,1)$ such that,*

$$\mathrm{Var}\left(\nabla_\theta \log \pi^{E}_{d,\theta}(a|s) Q(s,a)\right) \in \mathcal{O}\left(\beta^2 \alpha^{2d}\right),$$

for every Q. Further, if P^{π_b} is reversible or if the reward is constant, then $\alpha = |\lambda_2(P^{\pi_b})|$.

Theory versus Practice. We demonstrate the above result in simulation. We draw a random finite MDP, parameter vector $\Theta \in \mathbb{R}_+^S$, and behavior policy π_b. We then empirically compute the PG variance of E-SoftTreeMax as given in (1) and compare it to $|\lambda_2(P^{\pi_b})|^d$. We repeat this experiment three times for different P^{π_b} : (i) close to uniform, (ii) drawn

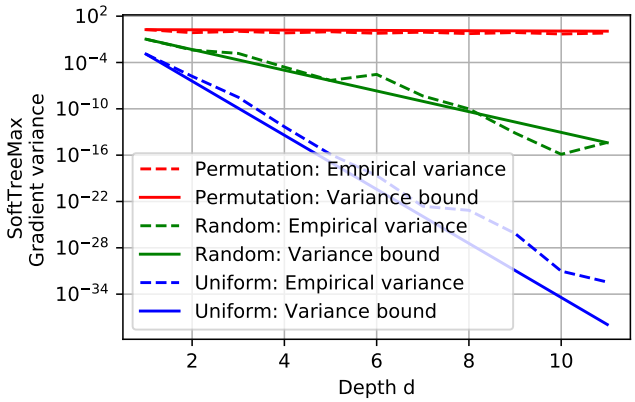

Figure 1. A comparison of the empirical PG variance and our bound for E-SoftTreeMax on randomly drawn MDPs. We present three cases for P^{π_b} : (i) close to uniform, (ii) drawn randomly, and (iii) close to a permutation matrix. This experiment verifies the optimal and worse-case rate decay cases. The variance bounds here are taken from Theorem 4.7 where we substitute $\alpha = |\lambda_2(P^{\pi_b})|$. To account for the constants, we match the values for the first point in $d = 1$.

randomly, and (iii) close to a permutation matrix. As seen in Figure 1, the empirical variance and our bound match almost identically. This also suggests that $\alpha = |\lambda_2(P^{\pi_b})|$ in the general case and not only when P^{π_b} is reversible or when the reward is constant.

4.3. Bias with an Approximate Forward Model

The definition of the two SoftTreeMax variants involves the knowledge of the underlying environment, in particular the value of P and r. However, in practice, we often can only learn approximations of the dynamics from interactions, e.g., using NNs (Ha & Schmidhuber, 2018; Schrittwieser et al., 2020). Let $\hat{P}$ and $\hat{r}$ denote the approximate kernel and reward functions, respectively. In this section, we study the consequences of the approximation error on the C-SoftTreeMax gradient.

Let $\hat{\pi}^{\text{C}}_{d,\theta}$ be the C-SoftTreeMax policy defined given the approximate forward model introduced above. That is, let $\hat{\pi}^{\text{C}}_{d,\theta}$ be defined exactly as in (5), but using $\hat{R}_s, \hat{P}_s, \hat{R}_{\pi_b}$ and $\hat{P}^{\pi_b}$, instead of their unperturbed counterparts from Section 2. Then, the variance of the corresponding gradient again decays exponentially with a decay rate of $\lambda_2(\hat{P}^{\pi_b})$. However, a gradient bias is introduced. In the following, we bound this bias in terms of the approximation error and other problem parameters. The proof is provided in Appendix A.9.

Theorem 4.8. *Let ϵ be the maximal model mis-specification, i.e., let $\max\{\|P - \hat{P}\|, \|r - \hat{r}\|\} = \epsilon$. Then the policy gradient bias due to $\hat{\pi}^{\text{C}}_{d,\theta}$ satisfies*

$$\left\| \frac{\partial}{\partial\theta}\left(\nu^\top V^{\pi^{\text{C}}_{d,\theta}}\right) - \frac{\partial}{\partial\theta}\left(\nu^\top V^{\hat{\pi}^{\text{C}}_{d,\theta}}\right) \right\| = \mathcal{O}\left(\frac{1}{(1-\gamma)^2\gamma^d} S\beta^2 d\epsilon\right). \tag{7}$$

Proof outline. First, we prove that $\max\{\|R_s - \hat{R}_s\|, \|P_s - \hat{P}_s\|, \|R_{\pi_b} - \hat{R}_{\pi_b}\|, \|P^{\pi_b} - \hat{P}^{\pi_b}\|\} = \mathcal{O}(\epsilon)$. This follows from the fact that the differences above are suitable convex combinations of either the rows of $P - \hat{P}$ or $r - \hat{r}$. We use the above observation along with the definitions of $\pi^{\text{C}}_{d,\theta}$ and $\hat{\pi}^{\text{C}}_{d,\theta}$ given in (5) to show that $\|\pi^{\text{C}}_{d,\theta} - \hat{\pi}^{\text{C}}_{d,\theta}\| = O(\beta d\epsilon)$. The proof for the latter builds upon two key facts: (a) $\|(P^{\pi_b})^k - (\hat{P}^{\pi_b})^k\| \leq \sum_{h=1}^{k} \|\hat{P}^{\pi_b}\|^{h-1}\|\hat{P}^{\pi_b} - P^{\pi_b}\|\|p^{\pi_b}\|^{k-h} = O(k\epsilon)$ for any $k \geq 0$, and (b) $|e^x - 1| = O(x)$ as $x \to 0$. Next, we decompose the LHS of (7) to get

$$\sum_s \left(\prod_{i=1}^{4} X_i(s) - \prod_{i=1}^{4} \hat{X}_i(s)\right) = \sum_s \sum_{i=1}^{4} \hat{X}_1(s)\cdots\hat{X}_{i-1}(s)\left(X_i(s) - \hat{X}_i(s)\right) \times X_{i+1}(s)\cdots X_4(s),$$

where $X_1(s) = d_{\pi^{\text{C}}_{d,\theta}}(s) \in \mathbb{R}$, $X_2(s) = (\nabla_\theta \log \pi^{\text{C}}_{d,\theta}(\cdot|s))^\top \in \mathbb{R}^{S\times A}$, $X_3(s) = D(\pi^{\text{C}}_{d,\theta}(\cdot|s)) \in \mathbb{R}^{A\times A}$, $X_4(s) = Q^{\pi^{\text{C}}_{d,\theta}}(s,\cdot) \in \mathbb{R}^{A\times A}$, and $\hat{X}_1(s), \ldots, \hat{X}_4(s)$ are similarly defined with $\pi^{\text{C}}_{d,\theta}$ replaced by $\hat{\pi}^{\text{C}}_{d,\theta}$. Then, we show that, for $i = 1, \ldots, 4$, (i) $\|X_i(s) - \hat{X}_i(s)\| = O(\epsilon)$ and (ii) $\max\{\|X_i\|, \|\hat{X}_i\|\}$ is bounded by problem parameters. From this, the desired result follows. □

To the best of our knowledge, Theorem 4.8 is the first result that bounds the bias of the gradient of a parametric policy due to an approximate model. This theorem reveals an intriguing trade-off in SoftTreeMax: while the variance decays exponentially with tree depth d as $O(|\lambda_2(P^{\pi_b})|^{d-1})$ according to Theorem 4.4, the gradient bias with an approximate model grows as $O(d\gamma^{-d})$. Since γ^{-d} grows exponentially with d, there exists an optimal tree depth that balances these opposing effects.

The bias also depends on the temperature parameter β, with higher temperature (lower β) reducing bias at the expense of more exploratory policies. In the extreme case of $\beta = 0$, the policy becomes uniform with no bias. Additionally, the error scales linearly with d because the policy suffers from cumulative error as it relies on further-looking states in the approximate model.

These results suggest that if the learned model is accurate enough, we can expect similar convergence properties for

C-SoftTreeMax as we would obtain with the true dynamics. In practice, one should adjust both d and β based on the estimated accuracy of the forward model to achieve the best performance. This is particularly important for practitioners implementing SoftTreeMax with learned forward models.

5. SoftTreeMax: Deep Parallel Implementation

Following impressive successes of deep RL (Mnih et al., 2015; Silver et al., 2016), using deep NNs in RL is standard practice. Depending on the RL algorithm, a loss function is defined and gradients on the network weights can be calculated. In PG methods, the scoring function used in the softmax is commonly replaced by a neural network W_θ: $\pi_\theta(a|s) \propto \exp\left(W_\theta(s,a)\right)$. Similarly, we implement SoftTreeMax by replacing $\theta(s)$ in (2) with a neural network $W_\theta(s)$. Although both variants of SoftTreeMax from Section 3 involve computing an expectation, this can be hard in general. One approach to handle it is with sampling, though these introduce estimation variance into the process. We leave the question of sample-based theory and algorithmic implementations for future work.

Instead, in finite action space environments such as Atari, we compute the exact expectation in SoftTreeMax with an exhaustive TS of depth d. Despite the exponential computational cost of spanning the entire tree, recent advancements in parallel GPU-based simulation allow efficient expansion of all nodes at the same depth simultaneously (Dalal et al., 2021; Rosenberg et al., 2022). This is possible when a simulator is implemented on GPU (Dalton et al., 2020; Makoviychuk et al., 2021; Freeman et al., 2021), or when a forward model is learned (Kim et al., 2020; Ha & Schmidhuber, 2018). To reduce the complexity to be linear in depth, we apply tree pruning to a limited width in all levels. We do so by sub-sampling only the most promising branches at each level. Limiting the width drastically improves runtime, and enables respecting GPU memory limits, with only a small sacrifice in performance.

To summarize, in the practical SoftTreeMax algorithm we perform an exhaustive tree expansion with pruning to obtain trajectories up to depth d. We expand the tree with equal weight to all actions, which corresponds to a uniform tree expansion policy π_b. We apply a neural network on the leaf states, and accumulate the result with the rewards along each trajectory to obtain the logits in (2). Finally, we aggregate the results using C-SoftTreeMax. We leave experiments E-SoftTreeMax for future work on risk-averse RL.

For a detailed illustration of our GPU-based tree expansion implementation, see Appendix B.2. In addition, we provide the psudeocode for C-SoftTreeMax in Algorithm 1 and its integration with PPO in Algorithm 2 in Appendix B.3.

During training, the gradient propagates to the NN weights of W_θ. When the gradient $\nabla_\theta \log \pi_{d,\theta}$ is calculated at each time step, it updates W_θ for all leaf states, similarly to Siamese networks (Bertinetto et al., 2016). An illustration of the policy is given in Figure 2.

6. Experiments

We conduct our experiments on multiple games from the Atari simulation suite (Bellemare et al., 2013). As a baseline, we train a PPO (Schulman et al., 2017) agent with 256 GPU workers in parallel (Dalton et al., 2020). For the tree expansion, we employ a GPU breadth-first as in (Dalal et al., 2021). We then train C-SoftTreeMax [1] for depths $d = 1 \ldots 8$, with a single worker. For depths $d \geq 3$, we limited the tree to a maximum width of 1024 nodes and pruned trajectories with low estimated weights. Since the distributed PPO baseline advances significantly faster in terms of environment steps, for a fair comparison, we ran all experiments for one week on the same machine. For more details see Appendix B.

In Figure 3, we plot the reward and variance of SoftTreeMax for each game, as a function of depth. The dashed lines are the results for PPO. Each value is taken after convergence, i.e., the average over the last 20% of the run. The numbers represent the average over five seeds per game. The plot conveys three intriguing conclusions. First, in all games, SoftTreeMax achieves significantly higher reward than PPO. Its gradient variance is also orders of magnitude lower than that of PPO. Second, the reward and variance are negatively correlated and mirror each other in almost all games. This phenomenon demonstrates the necessity of reducing the variance of PG for improving performance. Lastly, each game has a different sweet spot in terms of optimal tree depth. Recall that we limit the run-time in all experiments to one week The deeper the tree, the slower each step and the run consists of less steps. This explains the non-monotone behavior as a function of depth. For a more thorough discussion on the sweet spot of different games, see Appendix B.5.

7. Related Work

Reducing variance in PG estimates is essential for improving efficiency and stability. Approaches include baseline subtraction (Sutton et al., 1999; Weaver & Tao, 2001), action-dependent baselines (Wu et al., 2018), sub-sampling techniques like SVRPG (Papini et al., 2018), and natural policy

[1] We also experimented with E-SoftTreeMax and the results were almost identical. This is due to the quasi-deterministic nature of Atari, which causes the trajectory logits (2) to have almost no variability. We encourage future work on E-SoftTreeMax using probabilistic environments that are risk-sensitive.

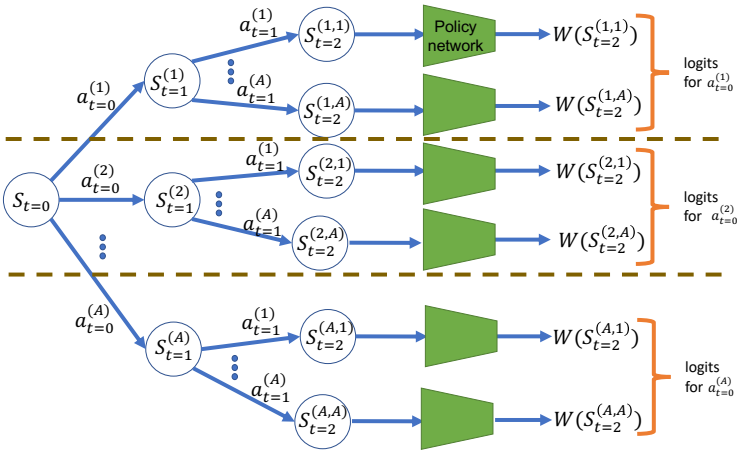

Figure 2. **SoftTreeMax policy**. Our exhaustive parallel tree expansion iterates on all actions at each state up to depth d ($= 2$ here). The leaf state of every trajectory is used as input to the policy network. The output is then added to the trajectory's cumulative reward as described in (2). I.e., instead of the standard softmax logits, we add the cumulative discounted reward to the policy network output. This policy is differentiable and can be easily integrated into any PG algorithm. In this work, we build on PPO and use its loss function to train the policy network.

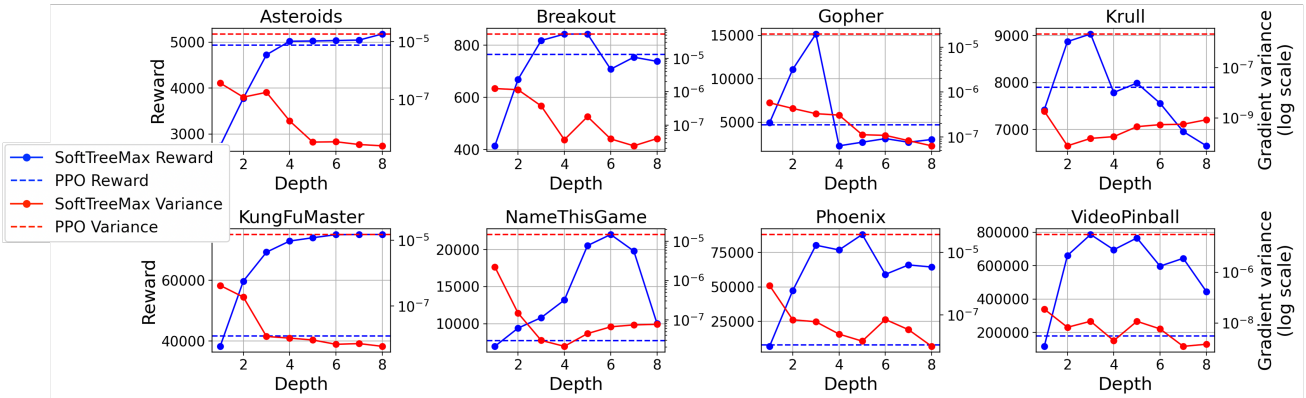

Figure 3. **Reward and Gradient variance: GPU SoftTreeMax (single worker) vs PPO (256 GPU workers).** The blue reward plots show the average of 50 evaluation episodes. The red variance plots show the average gradient variance of the corresponding training runs, averaged over five seeds. The dashed lines represent the same for PPO. Note that the variance y-axis is in log-scale.

gradient (Kakade, 2001). Multi-step returns for value estimation, such as n-step TD (Sutton et al., 1998) and GAE (Schulman et al., 2015), primarily affect learning targets rather than policy parameterization. Our approach differs by integrating planning directly into the policy structure, making it fundamentally different from both variance reduction techniques and multi-step return methods.

Softmax Operator. The softmax policy became a canonical part of PG to the point where theoretical results of PG focus specifically on it (Zhang et al., 2021; Mei et al., 2020b; Li et al., 2021; Ding et al., 2022). Even though we focus on a tree extension to the softmax policy, our methodology is general and can be easily applied to other discrete or continuous parameterized policies as in (Mei et al., 2020a; Miahi et al., 2021; Silva et al., 2019). It's important to distinguish between tree search and our approach of tree expansion. Traditional tree search methods like MCTS identify the best trajectory through a selection process, while our approach explores all possible trajectories up to a certain depth to compute an improved policy. Recent work by Efroni et al. (2018) showed that multi-step greedy policies improve convergence rates, and Protopapas & Barakat (2024) combined policy mirror descent with lookahead planning, though using different mechanisms than our SoftTreeMax policy.

Parallel Environments. In this work we used accurate parallel models that are becoming more common with the increasing popularity of GPU-based simulation (Makoviychuk et al., 2021; Dalton et al., 2020; Freeman et al., 2021).

Alternatively, in relation to Theorem 4.8, one can rely on recent works that learn the underlying model (Ha & Schmidhuber, 2018; Schrittwieser et al., 2020) and use an approximation of the true dynamics. **Risk Aversion.** Previous work considered exponential utility functions for risk aversion (Chen et al., 2007; Garcıa & Fernández, 2015; Fei et al., 2021). This utility function is the same as E-SoftTreeMax formulation from (4), but we have it directly in the policy instead of the objective. **Reward-free RL.** We showed that the gradient variance is minimized when the transitions induced by the behavior policy π_b are uniform. This is expressed by the second eigenvalue of the transition matrix P^{π_b}. This notion of uniform exploration is common to the reward-free RL setup (Jin et al., 2020). Several such works also considered the second eigenvalue in their analysis (Liu & Brunskill, 2018; Tarbouriech & Lazaric, 2019).

8. Discussion and Future Work

In this work, we introduced for the first time a differentiable parametric policy that combines TS with PG. We proved that SoftTreeMax is essentially a variance reduction technique and explained how to choose the expansion policy to minimize the gradient variance. It is an open question whether optimal variance reduction corresponds to the appealing regret properties the were put forward by UCT (Kocsis & Szepesvári, 2006). We believe that this can be answered by analyzing the convergence rate of SoftTreeMax, relying on the bias and variance results we obtained here.

As the learning process continues, the norm of the gradient and the variance *both* become smaller. On the face of it, one can ask if the gradient becomes small as fast as the variance or even faster can there be any meaningful learning? As we showed in the experiments, learning happens because the variance reduces fast enough (a variance of 0 represents deterministic learning, which is fastest).

Our work can be extended to infinite action spaces, where the theoretical analysis would involve transition kernels rather than matrices while preserving the same non-expansive operator properties. For implementation, the tree of continuous actions can be expanded by maintaining a parametric distribution over actions depending on θ, similar to a tree adaptation of MPPI (Williams et al., 2017) with a value function. This would significantly broaden the applicability of SoftTreeMax to important domains like robotics and continuous control.

Further important extensions include learning the forward model from interactions (Ha & Schmidhuber, 2018; Schrittwieser et al., 2020) rather than using the true dynamics. Our analysis in Theorem 4.8 already provides theoretical guidance on the resulting gradient bias, but empirical validation with learned models would bridge the gap between model-based approaches like MuZero and policy gradient methods. Additionally, adapting the behavior policy π_b over time as learning progresses could lead to more relevant exploration, though this requires careful management to maintain variance reduction benefits. Finally, the E-SoftTreeMax variant offers potential applications in risk-sensitive RL by naturally emphasizing exceptional trajectories through its reward exponentiation structure.

Acknowledgements

Gugan Thoppe's research is supported by the Walmart Centre for Tech Excellence at IISc, the Indo-French Centre for the Promotion of Advanced Research—CEFIPRA (7102-1), DST SERB's Core Research Grant CRG/2021/008330, and the Pratiksha Trust Young Investigator Award.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Reproducibility and Limitations

The code for our implementation is available at `https://github.com/NVlabs/SoftTreeMax`. We provide a docker file for setting up the environment and a README file with instructions on how to run both training and evaluation. The environment engine is an extension of Atari-CuLE (Dalton et al., 2020), a CUDA-based Atari emulator that runs on GPU. Our usage of a GPU environment is both a novelty and a current limitation of our work.

There are additional limitations to consider. First, our analysis focuses on MDPs with finite state and action spaces, and while we discuss theoretical extensions to continuous domains, practical implementations would require careful design choices to manage the sampling process efficiently. Second, while SoftTreeMax can be applied to any environment, it provides most benefit in quasi-deterministic environments where accurate forward planning is possible. In highly stochastic environments, the exponential growth of the tree width with depth would present challenges even with pruning.

References

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *J. Mach. Learn. Res.*, 22(98):1–76, 2021.

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.

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pp. 850–865. Springer, 2016.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. Natural actor–critic algorithms. *Automatica*, 45(11): 2471–2482, 2009.

Chatterjee, S. and Seneta, E. Towards consensus: Some convergence theorems on repeated averaging. *Journal of Applied Probability*, 14(1):89–97, 1977.

Chen, X., Sim, M., Simchi-Levi, D., and Sun, P. Risk aversion in inventory management. *Operations Research*, 55(5):828–842, 2007.

Dalal, G., Hallak, A., Dalton, S., Mannor, S., Chechik, G., et al. Improve agents without retraining: Parallel tree search with off-policy correction. *Advances in Neural Information Processing Systems*, 34:5518–5530, 2021.

Dalton, S. et al. Accelerating reinforcement learning through gpu atari emulation. *Advances in Neural Information Processing Systems*, 33:19773–19782, 2020.

Ding, Y., Zhang, J., and Lavaei, J. On the global optimum convergence of momentum-based policy gradient. In *International Conference on Artificial Intelligence and Statistics*, pp. 1910–1934. PMLR, 2022.

Efroni, Y., Dalal, G., Scherrer, B., and Mannor, S. Beyond the one-step greedy approach in reinforcement learning. In *International Conference on Machine Learning*, pp. 1387–1396. PMLR, 2018.

Fei, Y., Yang, Z., Chen, Y., and Wang, Z. Exponential bellman equation and improved regret bounds for risk-sensitive reinforcement learning. *Advances in Neural Information Processing Systems*, 34:20436–20446, 2021.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Greensmith, E., Bartlett, P. L., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Howard, R. A. and Matheson, J. E. Risk-sensitive markov decision processes. *Management science*, 18(7):356–369, 1972.

Jin, C., Krishnamurthy, A., Simchowitz, M., and Yu, T. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pp. 4870–4879. PMLR, 2020.

Kakade, S. M. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Kim, S. W., Zhou, Y., Philion, J., Torralba, A., and Fidler, S. Learning to simulate dynamic environments with gamegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1231–1240, 2020.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Levin, D. A. and Peres, Y. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Li, G., Wei, Y., Chi, Y., Gu, Y., and Chen, Y. Softmax policy gradient methods can take exponential time to converge. In *Conference on Learning Theory*, pp. 3107–3110. PMLR, 2021.

Liu, Y. and Brunskill, E. When simple exploration is sample efficient: Identifying sufficient conditions for random exploration to yield pac rl algorithms. *arXiv preprint arXiv:1805.09045*, 2018.

Liu, Y., Zhang, K., Basar, T., and Yin, W. An improved analysis of (variance-reduced) policy gradient and natural policy gradient methods. *Advances in Neural Information Processing Systems*, 33:7624–7636, 2020.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Mathkar, A. S. and Borkar, V. S. Nonlinear gossip. *SIAM Journal on Control and Optimization*, 54(3):1535–1557, 2016.

Mei, J., Xiao, C., Dai, B., Li, L., Szepesvári, C., and Schuurmans, D. Escaping the gravitational pull of softmax. *Advances in Neural Information Processing Systems*, 33: 21130–21140, 2020a.

Mei, J., Xiao, C., Szepesvari, C., and Schuurmans, D. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pp. 6820–6829. PMLR, 2020b.

Miahi, E., MacQueen, R., Ayoub, A., Masoumzadeh, A., and White, M. Resmax: An alternative soft-greedy operator for reinforcement learning. 2021.

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.

Noorani, E. and Baras, J. S. Risk-sensitive reinforce: A monte carlo policy gradient algorithm for exponential performance criteria. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 1522–1527. IEEE, 2021.

Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. Stochastic variance-reduced policy gradient. In *International conference on machine learning*, pp. 4026–4035. PMLR, 2018.

Pelletier, M. On the almost sure asymptotic behaviour of stochastic algorithms. *Stochastic processes and their applications*, 78(2):217–244, 1998.

Pham, N., Nguyen, L., Phan, D., Nguyen, P. H., Dijk, M., and Tran-Dinh, Q. A hybrid stochastic policy gradient algorithm for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 374–385. PMLR, 2020.

Protopapas, K. and Barakat, A. Policy mirror descent with lookahead. *Advances in Neural Information Processing Systems*, 37:26443–26481, 2024.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. Stable baselines3, 2019.

Rosenberg, A., Hallak, A., Mannor, S., Chechik, G., and Dalal, G. Planning and learning with adaptive lookahead. *arXiv preprint arXiv:2201.12403*, 2022.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

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

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

Shen, Z., Ribeiro, A., Hassani, H., Qian, H., and Mi, C. Hessian aided policy gradient. In *International conference on machine learning*, pp. 5729–5738. PMLR, 2019.

Silva, A., Killian, T., Rodriguez, I. D. J., Son, S.-H., and Gombolay, M. Optimization methods for interpretable differentiable decision trees in reinforcement learning. *arXiv preprint arXiv:1903.09338*, 2019.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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.

Szepesvári, C. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

Tarbouriech, J. and Lazaric, A. Active exploration in markov decision processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 974–982. PMLR, 2019.

Thomas, P. S. and Brunskill, E. Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines. *arXiv preprint arXiv:1706.06643*, 2017.

Weaver, L. and Tao, N. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 538–545, 2001.

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J. M., Boots, B., and Theodorou, E. A. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721. IEEE, 2017.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A. M., Kakade, S., Mordatch, I., and Abbeel, P. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018.

Xu, P., Gao, F., and Gu, Q. An improved convergence analysis of stochastic variance-reduced policy gradient. In *Uncertainty in Artificial Intelligence*, pp. 541–551. PMLR, 2020.

Zhang, J., He, T., Sra, S., and Jadbabaie, A. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*, 2019.

Zhang, J., Ni, C., Szepesvari, C., Wang, M., et al. On the convergence and sample efficiency of variance-reduced policy gradient method. *Advances in Neural Information Processing Systems*, 34:2228–2240, 2021.

Appendix

A. Proofs

A.1. Proof of Lemma 4.1 – Bound on the policy gradient variance

For any parametric policy π_θ and function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\mathrm{Var}\left(\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right) \leq \max_{s,a} \left[Q(s,a)\right]^2 \max_s \|\nabla_\theta \log \pi_\theta(\cdot|s)\|_F^2,$$

where $\nabla_\theta \log \pi_\theta(\cdot|s) \in \mathbb{R}^{A \times \dim(\theta)}$ is a matrix whose a-th row is $\nabla_\theta \log \pi_\theta(a|s)^\top$.

Proof. The variance for a parametric policy π_θ is given as follows:

$$\begin{aligned}\mathrm{Var}\left(\nabla_\theta \log \pi_\theta(a|s) Q(a,s)\right) =& \mathbb{E}_{s\sim d_{\pi_\theta}, a\sim\pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s)^\top \nabla_\theta \log \pi_\theta(a|s) Q(s,a)^2\right] - \\ & \mathbb{E}_{s\sim d_{\pi_\theta}, a\sim\pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right]^\top \mathbb{E}_{s\sim d_{\pi_\theta}, a\sim\pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right],\end{aligned}$$

where $Q(s,a)$ is the currently estimated Q-function and d_{π_θ} is the discounted state visitation frequency induced by the policy π_θ. Since the second term we subtract is always positive (it is of quadratic form $v^\top v$) we can bound the variance by the first term:

$$\begin{aligned}\mathrm{Var}\left(\nabla_\theta \log \pi_\theta(a|s) Q(a,s)\right) \leq& \mathbb{E}_{s\sim d_{\pi_\theta}, a\sim\pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s)^\top \nabla_\theta \log \pi_\theta(a|s) Q(s,a)^2\right] \\ =& \sum_s d_{\pi_\theta}(s) \sum_a \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top \nabla_\theta \log \pi_\theta(a|s) Q(s,a)^2 \\ \leq& \max_{s,a} \left[\left[Q(s,a)\right]^2 \pi_\theta(a|s)\right] \sum_s d_{\pi_\theta}(s) \sum_a \nabla_\theta \log \pi_\theta(a|s)^\top \nabla_\theta \log \pi_\theta(a|s) \\ \leq& \max_{s,a} \left[Q(s,a)\right]^2 \max_s \sum_a \nabla_\theta \log \pi_\theta(a|s)^\top \nabla_\theta \log \pi_\theta(a|s) \\ =& \max_{s,a} \left[Q(s,a)\right]^2 \max_s \|\nabla_\theta \log \pi_\theta(\cdot|s)\|_F^2.\end{aligned}$$

□

A.2. Proof of Lemma 4.2 – Vector form of C-SoftTreeMax

In vector form, (3) is given by

$$\pi^{\mathrm{C}}_{d,\theta}(\cdot|s) = \frac{\exp\left[\beta\left(C_{s,d} + P_s \left(P^{\pi_b}\right)^{d-1} \Theta\right)\right]}{\mathbf{1}_A^\top \exp\left[\beta\left(C_{s,d} + P_s \left(P^{\pi_b}\right)^{d-1} \Theta\right)\right]}, \tag{8}$$

where

$$C_{s,d} = \gamma^{-d} R_s + P_s \left[\sum_{h=1}^{d-1} \gamma^{h-d} \left(P^{\pi_b}\right)^{h-1}\right] R_{\pi_b}. \tag{9}$$

Proof. Consider the vector $\ell_{s,\cdot} \in \mathbb{R}^{|\mathcal{A}|}$. Its expectation satisfies

$$\begin{aligned}\mathbb{E}^{\pi_b} \ell_{s,\cdot}(d;\theta) &= \mathbb{E}^{\pi_b}\left[\sum_{t=0}^{d-1} \gamma^{t-d} r_t + \theta(s_d)\right] \\ &= \gamma^{-d} R_s + \sum_{t=1}^{d-1} \gamma^{t-d} P_s (P^{\pi_b})^{t-1} R_{\pi_b} + P_s (P^{\pi_b})^{d-1} \Theta.\end{aligned}$$

As required. □

A.3. Proof of Lemma 4.3 – Gradient of C-SoftTreeMax

The C-SoftTreeMax gradient of dimension $A \times S$ is given by

$$\nabla_\theta \log \pi^{\text{C}}_{d,\theta} = \beta \left[I_A - \mathbf{1}_A (\pi^{\text{C}}_{d,\theta})^\top \right] P_s \left(P^{\pi_b} \right)^{d-1},$$

where for brevity, we drop the s index in the policy above, i.e., $\pi^{\text{C}}_{d,\theta} \equiv \pi^{\text{C}}_{d,\theta}(\cdot|s)$.

Proof. The (j,k)-th entry of $\nabla_\theta \log \pi^{\text{C}}_{d,\theta}$ satisifes

$$\begin{aligned}
[\nabla_\theta \log \pi^{\text{C}}_{d,\theta}]_{j,k} &= \frac{\partial \log(\pi^{\text{C}}_{d,\theta}(a^j|s))}{\partial \theta(s^k)} \\
&= \beta [P_s (P^{\pi_b})^{d-1}]_{j,k} - \frac{\sum_a \left[\exp\left[\beta \left(C_{s,d} + P_s \left(P^{\pi_b} \right)^{d-1} \Theta \right) \right] \right]_a \beta \left[P_s (P^{\pi_b})^{d-1} \right]_{a,k}}{\mathbf{1}_A^\top \exp\left[\beta \left(C_{s,d} + P_s \left(P^{\pi_b} \right)^{d-1} \Theta \right) \right]} \\
&= \beta [P_s (P^{\pi_b})^{d-1}]_{j,k} - \beta \sum_a \pi^{\text{C}}_{d,\theta}(a|s) \left[P_s (P^{\pi_b})^{d-1} \right]_{a,k} \\
&= \beta [P_s (P^{\pi_b})^{d-1}]_{j,k} - \beta \left[(\pi^{\text{C}}_{d,\theta})^\top P_s (P^{\pi_b})^{d-1} \right]_k \\
&= \beta [P_s (P^{\pi_b})^{d-1}]_{j,k} - \beta \left[\mathbf{1}_A (\pi^{\text{C}}_{d,\theta})^\top P_s (P^{\pi_b})^{d-1} \right]_{j,k}.
\end{aligned}$$

Moving back to matrix form, we obtain the stated result. □

A.4. Proof of Theorem 4.4 – Exponential variance decay of C-SoftTreeMax

The C-SoftTreeMax policy gradient is bounded by

$$\text{Var}\left(\nabla_\theta \log \pi^{\text{C}}_{d,\theta}(a|s) Q(s,a) \right) \leq 2 \frac{A^2 S^2 \beta^2}{(1-\gamma)^2} |\lambda_2(P^{\pi_b})|^{2(d-1)}.$$

Proof. We use Lemma 4.1 directly. First of all, it is know that when the reward is bounded in $[0,1]$, the maximal value of the Q-function is $\frac{1}{1-\gamma}$ as the sum as infinite discounted rewards. Next, we bound the Frobenius norm of the term achieved in Lemma 4.3, by applying the eigen-decomposition on P^{π_b}:

$$P^{\pi_b} = \mathbf{1}_S \mu^\top + \sum_{i=2}^{S} \lambda_i u_i v_i^\top, \tag{10}$$

where μ is the stationary distribution of P^{π_b}, and u_i and v_i are left and right eigenvectors correspondingly.

$$\begin{aligned}
\|\beta \left(I_{A,A} - \mathbf{1}_A \pi^\top \right) P_s (P^{\pi_b})^{d-1} \|_F &= \beta \| \left(I_{A,A} - \mathbf{1}_A \pi^\top \right) P_s \left(\mathbf{1}_S \mu^\top + \sum_{i=2}^{S} \lambda_i^{d-1} u_i v_i^\top \right) \|_F \\
\textit{(}P_s\textit{ is stochastic)} \quad &= \beta \| \left(I_{A,A} - \mathbf{1}_A \pi^\top \right) \left(\mathbf{1}_A \mu^\top + \sum_{i=2}^{S} \lambda_i^{d-1} P_s u_i v_i^\top \right) \|_F \\
\textit{(projection nullifies } \mathbf{1}_A \mu^\top\textit{)} \quad &= \beta \| \left(I_{A,A} - \mathbf{1}_A \pi^\top \right) \left(\sum_{i=2}^{S} \lambda_i^{d-1} P_s u_i v_i^\top \right) \|_F \\
\textit{(triangle inequality)} \quad &\leq \beta \sum_{i=2}^{S} \| \left(I_{A,A} - \mathbf{1}_A \pi^\top \right) \left(\lambda_i^{d-1} P_s u_i v_i^\top \right) \|_F \\
\textit{(matrix norm sub-multiplicativity)} \quad &\leq \beta |\lambda_2^{d-1}| \sum_{i=2}^{S} \| I_{A,A} - \mathbf{1}_A \pi^\top \|_F \| P_s \|_F \| u_i v_i^\top \|_F \\
&= \beta |\lambda_2^{d-1}| (S-1) \| I_{A,A} - \mathbf{1}_A \pi^\top \|_F \| P_s \|_F.
\end{aligned}$$

Now, we can bound the norm $\|I_{A,A} - \mathbf{1}_A \pi^\top\|_F$ by direct calculation:

$$\begin{aligned}
\|I_{A,A} - \mathbf{1}_A \pi^\top\|_F^2 &= \mathrm{Tr}\left[\left(I_{A,A} - \mathbf{1}_A \pi^\top\right)\left(I_{A,A} - \mathbf{1}_A \pi^\top\right)^\top\right] && (11)\\
&= \mathrm{Tr}\left[I_{A,A} - \mathbf{1}_A \pi^\top - \pi \mathbf{1}_A^\top + \pi^\top \pi \mathbf{1}_A \mathbf{1}_A^\top\right] && (12)\\
&= A - 1 - 1 + A\pi^\top \pi && (13)\\
&\leq 2A. && (14)
\end{aligned}$$

From the Cauchy-Schwartz inequality,

$$\|P_s\|_F^2 = \sum_a \sum_s \left[[P_s]_{a,s}\right]^2 = \sum_a \|[P_s]_{a,\cdot}\|_2^2 \leq \sum_a \|[P_s]_{a,\cdot}\|_1 \|[P_s]_{a,\cdot}\|_\infty \leq A.$$

So,

$$\begin{aligned}
\mathrm{Var}\left(\nabla_\theta \log \pi^{\mathrm{C}}_{d,\theta}(a|s) Q(s,a)\right) &\leq \max_{s,a} \left[Q(s,a)\right]^2 \max_s \|\nabla_\theta \log \pi^{\mathrm{C}}_{d,\theta}(\cdot|s)\|_F^2\\
&\leq \frac{1}{(1-\gamma)^2} \|\beta\left(I_{A,A} - \mathbf{1}_A \pi^\top\right) P_s (P^{\pi_b})^{d-1}\|_F^2\\
&\leq \frac{1}{(1-\gamma)^2} \beta^2 |\lambda_2(P^{\pi_b})|^{2(d-1)} S^2 (2A^2),
\end{aligned}$$

which obtains the desired bound. □

A.5. A lower bound on C-SoftTreeMax gradient (result not in the paper)

For completeness we also supply a lower bound on the Frobenius norm of the gradient. Note that this result does not translate to the a lower bound on the variance since we have no lower bound equivalence of Lemma 4.1.

Lemma A.1. *The Frobenius norm on the gradient of the policy is lower-bounded by:*

$$\|\nabla_\theta \log \pi^{C}_{d,\theta}(\cdot|s)\|_F \geq C \cdot \beta |\lambda_2(P^{\pi_b})|^{(d-1)}. \tag{15}$$

Proof. We begin by moving to the induced l_2 norm by norm-equivalence:

$$\|\beta\left(I_{A,A} - \mathbf{1}_A \pi^\top\right) P_s (P^{\pi_b})^{d-1}\|_F \geq \|\beta\left(I_{A,A} - \mathbf{1}_A \pi^\top\right) P_s (P^{\pi_b})^{d-1}\|_2.$$

Now, taking the vector u to be the eigenvector of the second eigenvalue of P^{π_b}:

$$\begin{aligned}
\|\beta\left(I_{A,A} - \mathbf{1}_A \pi^\top\right) P_s (P^{\pi_b})^{d-1}\|_2 &\geq \|\beta\left(I_{A,A} - \mathbf{1}_A \pi^\top\right) P_s (P^{\pi_b})^{d-1} u\|_2\\
&= \beta \|\left(I_{A,A} - \mathbf{1}_A \pi^\top\right) P_s u\|_2\\
&= \beta |\lambda_2(P^{\pi_b})|^{(d-1)} \|\left(I_{A,A} - \mathbf{1}_A \pi^\top\right) P_s u\|_2.
\end{aligned}$$

Note that even though $P_s u$ can be 0, that is not the common case since we can freely change π_b (and therefore the eigenvectors of P^{π_b}). □

A.6. Proof of Lemma 4.5 – Vector form of E-SoftTreeMax

For $d \geq 1$, (4) is given by

$$\pi^{\mathrm{E}}_{d,\theta}(\cdot|s) = \frac{E_{s,d} \exp(\beta\Theta)}{1_A^\top E_{s,d} \exp(\beta\Theta)}, \tag{16}$$

where

$$E_{s,d} = P_s \prod_{h=1}^{d-1} \left(D\left(\exp[\beta\gamma^{h-d} R]\right) P^{\pi_b}\right) \tag{17}$$

with R being the $|S|$-dimensional vector whose s-th coordinate is $r(s)$.

Proof. Recall that

$$\ell_{s,a}(d;\theta) = \gamma^{-d}\left[r(s) + \sum_{t=1}^{d-1}\gamma^t r(s_t) + \gamma^d \theta(s_d)\right]. \tag{18}$$

and, hence,

$$\exp[\beta \ell_{s,a}(d;\theta)] = \exp\left[\beta\gamma^{-d}\left(r(s) + \sum_{t=1}^{d-1}\gamma^t r(s_t) + \gamma^d \theta(s_d)\right)\right]. \tag{19}$$

Therefore,

$$\mathbb{E}[\exp \beta \ell_{s,a}(d;\theta)] = \mathbb{E}\left[\exp\left[\beta\gamma^{-d}\left(r(s) + \sum_{t=1}^{d-1}\gamma^t r(s_t)\right)\right]\mathbb{E}\left[\exp\left[\beta\left(\theta(s_d)\right)\right]|s_1,\dots,s_{d-1}\right]\right] \tag{20}$$

$$= \mathbb{E}\left[\exp\left[\beta\gamma^{-d}\left(r(s) + \sum_{t=1}^{d-1}\gamma^t r(s_t)\right)\right]P^{\pi_b}(\cdot|s_{d-1})\right]\exp(\beta\Theta) \tag{21}$$

$$= \mathbb{E}\left[\exp\left[\beta\gamma^{-d}\left(r(s) + \sum_{t=1}^{d-2}\gamma^t r(s_t)\right)\right]\exp[\beta\gamma^{-1}r(s_{d-1})]P^{\pi_b}(\cdot|s_{d-1})\right]\exp(\beta\Theta). \tag{22}$$

By repeatedly using iterative conditioning as above, the desired result follows. Note that $\exp(\beta\gamma^{-d}r(s))$ does not depend on the action and is therefore cancelled out with the denominator. □

A.7. Proof of Lemma 4.6 – Gradient of E-SoftTreeMax

The E-SoftTreeMax gradient of dimension $A \times S$ is given by

$$\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} = \beta\left[I_A - \mathbf{1}_A(\pi^{\mathrm{E}}_{d,\theta})^\top\right]\frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1}E_{s,d}D(\exp(\beta\Theta))}{\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta)},$$

where for brevity, we drop the s index in the policy above, i.e., $\pi^{\mathrm{E}}_{d,\theta} \equiv \pi^{\mathrm{E}}_{d,\theta}(\cdot|s)$.

Proof. The (j,k)-th entry of $\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}$ satisfies

$$\begin{aligned}
[\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}]_{j,k} &= \frac{\partial \log(\pi^{\mathrm{E}}_{d,\theta}(a^j|s))}{\partial\theta(s^k)} \\
&= \frac{\partial}{\partial\theta(s^k)}\left(\log[(E_{s,d})_j^\top \exp(\beta\Theta)] - \log[\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta)]\right) \\
&= \frac{\beta(E_{s,d})_{j,k}\exp(\beta\theta(s^k))}{(E_{s,d})_j^\top\exp(\beta\Theta)} - \frac{\beta\mathbf{1}_A^\top E_{s,d}e_k\exp(\beta\theta(s^k))}{\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta)} \\
&= \frac{\beta(E_{s,d}e_k\exp(\beta\theta(s^k)))_j}{(E_{s,d})_j^\top\exp(\beta\Theta)} - \frac{\beta\mathbf{1}_A^\top E_{s,d}e_k\exp(\beta\theta(s^k))}{\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta)} \\
&= \beta\left[\frac{e_j^\top}{e_j^\top E_{s,d}\exp(\beta\Theta)} - \frac{\mathbf{1}_A^\top}{\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta)}\right]E_{s,d}e_k\exp(\beta\theta(s^k)).
\end{aligned}$$

Hence,

$$[\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}]_{\cdot,k} = \beta\left[D(E_{s,d}\exp(\beta\Theta))^{-1} - (\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta))^{-1}\mathbf{1}_A\mathbf{1}_A^\top\right]E_{s,d}e_k\exp(\beta\theta(s^k))$$

From this, it follows that

$$\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} = \beta\left[D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1} - \mathbf{1}_A\mathbf{1}_A^\top\right]\frac{E_{s,d}D(\exp(\beta\Theta))}{\mathbf{1}_A^\top E_{s,d}\exp(\beta\Theta)}. \tag{23}$$

The desired result is now easy to see. □

A.8. Proof of Theorem 4.7 — Exponential variance decay of E-SoftTreeMax

There exists $\alpha \in (0,1)$ such that, for any function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\mathrm{Var}\left(\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}(a|s) Q(s,a)\right) \in \mathcal{O}\left(\beta^2 \alpha^{2d}\right).$$

If all rewards are equal ($r \equiv$ const), then $\alpha = |\lambda_2(P^{\pi_b})|$.

Proof outline. Recall that thanks to Lemma 4.1, we can bound the PG variance using a direct bound on the gradient norm. The definition of the induced norm is

$$\|\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}\| = \max_{z:\|z\|=1} \|\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} z\|,$$

with $\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}$ given in Lemma 4.6. Let $z \in \mathbb{R}^S$ be an arbitrary vector such that $\|z\| = 1$. Then, $z = \sum_{i=1}^S c_i z_i$, where c_i are scalar coefficients and z_i are vectors spanning the S-dimensional space. In the full proof, we show our specific choice of z_i and prove they are linearly independent given that choice. We do note that $z_1 = \mathbf{1}_S$.

The first part of the proof relies on the fact that $(\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}) z_1 = 0$. This is easy to verify using Lemma 4.6 together with (6), and because $\left[I_A - \mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top\right]$ is a projection matrix whose null-space is spanned by $\mathbf{1}_S$. Thus,

$$\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} z = \nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} \sum_{i=2}^S c_i z_i.$$

In the second part of the proof, we focus on $E_{s,d}$ from (6), which appears within $\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta}$. Notice that $E_{s,d}$ consists of the product $\prod_{h=1}^{d-1} \left(D\left(\exp(\beta\gamma^{h-d} R\right) P^{\pi_b}\right)$. Even though the elements in this product are not stochastic matrices, in the full proof we show how to normalize each of them to a stochastic matrix B_h. We thus obtain that

$$E_{s,d} = P_s D(M_1) \prod_{h=1}^{d-1} B_h,$$

where $M_1 \in \mathbb{R}^S$ is some strictly positive vector. Then, we can apply a result by Mathkar & Borkar (2016), which itself builds on (Chatterjee & Seneta, 1977). The result states that the product of stochastic matrices $\prod_{h=1}^{d-1} B_h$ of our particular form converges exponentially fast to a matrix of the form $\mathbf{1}_S \mu^\top$ s.t. $\|\mathbf{1}_S \mu^\top - \prod_{h=1}^{d-1} B_h\| \le C\alpha^d$ for some constant C.

Lastly, $\mathbf{1}_S \mu_{\pi_b}^\top$ gets canceled due to our choice of $z_i,\ i = 2, \dots, S$. This observation along with the above fact that the remainder decays then shows that $\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} \sum_{i=2}^S z_i = \mathcal{O}(\alpha^d)$, which gives the desired result. □

Full technical proof. Let $d \ge 2$. Recall that

$$E_{s,d} = P_s \prod_{h=1}^{d-1} \left(D\left(\exp[\beta\gamma^{h-d} R]\right) P^{\pi_b}\right), \tag{24}$$

and that R refers to the S-dimensional vector whose s-th coordinate is $r(s)$.

Define

$$B_i = \begin{cases} P^{\pi_b} & \text{if } i = d-1, \\ D^{-1}(P^{\pi_b} M_{i+1}) P^{\pi_b} D(M_{i+1}) & \text{if } i = 1, \dots, d-2, \end{cases} \tag{25}$$

and the vector

$$M_i = \begin{cases} \exp(\beta\gamma^{-1} R) & \text{if } i = d-1, \\ \exp(\beta\gamma^{i-d} R) \circ P^{\pi_b} M_{i+1} & \text{if } i = 1, \dots, d-2, \end{cases} \tag{26}$$

where $\circ$ denotes the element-wise product. Then,

$$E_{s,d} = P_s D(M_1) \prod_{i=1}^{d-1} B_i. \tag{27}$$

It is easy to see that each B_i is a row-stochastic matrix, i.e., all entries are non-negative and $B_i \mathbf{1}_S = \mathbf{1}_S$.

Next, we prove that all non-zeros entries of B_i are bounded away from 0 by a constant. This is necessary to apply the next result from (Chatterjee & Seneta, 1977). The j-th coordinate of M_i satisfies

$$(M_i)_j = \exp[\beta\gamma^{i-d}R_j] \sum_k [P^{\pi_b}]_{j,k}(M_{i+1})_k \leq \| \exp[\beta\gamma^{i-d}R]\|_\infty \|M_{i+1}\|_\infty. \tag{28}$$

Separately, observe that $\|M_{d-1}\|_\infty \leq \| \exp(\beta\gamma^{-1}R)\|_\infty$. Plugging these relations in (26) gives

$$\|M_1\|_\infty \leq \prod_{h=1}^{d-1} \| \exp[\beta\gamma^{h-d}R]\|_\infty = \prod_{h=1}^{d-1} \| \exp[\beta\gamma^{-d}R]\|_\infty^{\gamma^h} = \| \exp[\beta\gamma^{-d}R]\|_\infty^{\sum_{h=1}^{d-1}\gamma^h} \leq \| \exp[\beta\gamma^{-d}R]\|_\infty^{\frac{1}{1-\gamma}}. \tag{29}$$

Similarly, for every $1 \leq i \leq d-1$, we have that

$$\|M_i\|_\infty \leq \prod_{h=i}^{d-1} \| \exp[\beta\gamma^{-d}R]\|_\infty^{\gamma^h} \leq \| \exp[\beta\gamma^{-d}R]\|_\infty^{\frac{1}{1-\gamma}}. \tag{30}$$

The jk-th entry of $B_i = D^{-1}(P^{\pi_b}M_{i+1})P^{\pi_b}D(M_{i+1})$ is

$$(B_i)_{jk} = \frac{P^{\pi_b}_{jk}[M_{i+1}]_k}{\sum_{\ell=1}^{|S|} P^{\pi_b}_{j\ell}[M_{i+1}]_\ell} \geq \frac{P^{\pi_b}_{jk}}{\sum_{\ell=1}^{|S|} P^{\pi_b}_{j\ell}[M_{i+1}]_\ell} \geq \frac{P^{\pi_b}_{jk}}{\| \exp[\beta\gamma^{-d}R]\|_\infty^{\frac{1}{1-\gamma}}}. \tag{31}$$

Hence, for non-zero $P^{\pi_b}_{jk}$, the entries are bounded away from zero by the same. We can now proceed with applying the following result.

Now, by (Chatterjee & Seneta, 1977, Theorem 5) (see also (14) in (Mathkar & Borkar, 2016)), $\lim_{d\to\infty} \prod_{i=1}^{d-1} B_i$ exists and is of the form $\mathbf{1}_S\mu^\top$ for some probability vector μ. Furthermore, there is some $\alpha \in (0,1)$ such that $\varepsilon(d) := \left(\prod_{i=1}^{d-1} B_i\right) - \mathbf{1}_S\,\mu^\top$ satisfies

$$\|\varepsilon(d)\| = O(\alpha^d). \tag{32}$$

Pick linearly independent vectors $w_2, \ldots, w_S$ such that

$$\mu^\top w_i = 0 \text{ for } i = 2, \ldots, d. \tag{33}$$

Since $\sum_{i=2}^S \alpha_i w_i$ is perpendicular to μ for any $\alpha_2, \ldots \alpha_S$ and because $\mu^\top \exp(\beta\Theta) > 0$, there exists no choice of $\alpha_2, \ldots, \alpha_S$ such that $\sum_{i=2}^S \alpha_i w_i = \exp(\beta\Theta)$. Hence, if we let $z_1 = \mathbf{1}_S$ and $z_i = D(\exp(\beta\Theta))^{-1}w_i$ for $i = 2, \ldots, S$, then it follows that $\{z_1, \ldots, z_S\}$ is linearly independent. In particular, it implies that $\{z_1, \ldots, z_S\}$ spans $\mathbb{R}^S$.

Now consider an arbitrary unit norm vector $z := \sum_{i=1}^{S} c_i z_i \in \mathbb{R}^S$ s.t. $\|z\|_2 = 1$. Then,

$$\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} z = \nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} \sum_{i=2}^{S} c_i z_i \tag{34}$$

$$= \beta \left[I_A - \mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top \right] \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1} E_{s,d} D(\exp(\beta\Theta))}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} \sum_{i=2}^{S} c_i z_i \tag{35}$$

$$= \beta \left[I_A - \mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top \right] \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1} E_{s,d}}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} \sum_{i=2}^{S} c_i w_i \tag{36}$$

$$= \beta \left[I_A - \mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top \right] \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1} \left[\mathbf{1}_S \mu^\top + \varepsilon(d)\right]}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} \sum_{i=2}^{S} c_i w_i \tag{37}$$

$$= \beta \left[I_A - \mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top \right] \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1} \varepsilon(d)}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} \sum_{i=2}^{S} c_i w_i \tag{38}$$

$$= \beta \left[I_A - \mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top \right] \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1} \varepsilon(d) D(\exp(\beta\Theta))}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} (z - c_1 \mathbf{1}_S), \tag{39}$$

where (34) follows from the fact that $\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} z_1 = \nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} \mathbf{1}_S = 0$, (35) follows from Lemma 4.6, (36) holds since $z_i = D(\exp(\beta\Theta))^{-1} w_i$, (38) because μ is perpendicular w_i for each i, while (39) follows by reusing $z_i = D(\exp(\beta\Theta))^{-1} w_i$ relation along with the fact that $z_1 = \mathbf{1}_S$.

From (39), it follows that

$$\|\nabla_\theta \log \pi^{\mathrm{E}}_{d,\theta} z\| \leq \beta \|\varepsilon(d)\| \left\| \left[I_A - \mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top \right] \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1}}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} \right\| \|D(\exp(\beta\Theta))\| \, \|z - c_1 \mathbf{1}_S\| \tag{40}$$

$$\leq \beta \alpha^d (\|I_A\| + \|\mathbf{1}_A (\pi^{\mathrm{E}}_{d,\theta})^\top\|) \left\| \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1}}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} \right\| \exp(\beta \max_s \theta(s)) \|z - c_1 \mathbf{1}_S\| \tag{41}$$

$$\leq \beta \alpha^d (1 + \sqrt{A}) \left\| \frac{D\left(\pi^{\mathrm{E}}_{d,\theta}\right)^{-1}}{\mathbf{1}_A^\top E_{s,d} \exp(\beta\Theta)} \right\| \exp(\beta \max_s \theta(s)) \|z - c_1 \mathbf{1}_S\| \tag{42}$$

$$\leq \beta \alpha^d (1 + \sqrt{A}) \left\| D^{-1}(E_{s,d} \exp(\beta\Theta)) \right\| \exp(\beta \max_s \theta(s)) \|z - c_1 \mathbf{1}_S\| \tag{43}$$

$$\leq \beta \alpha^d (1 + \sqrt{A}) \frac{1}{\min_s [E_{s,d} \exp(\beta\Theta]_s} \exp(\beta \max_s \theta(s)) \|z - c_1 \mathbf{1}_S\| \tag{44}$$

$$\leq \beta \alpha^d (1 + \sqrt{A}) \frac{\exp(\beta \max_s \theta(s))}{\exp(\beta \min_s \theta(s)) \min_s |M_1|} \|z - c_1 \mathbf{1}_S\| \tag{45}$$

$$\leq \beta \alpha^d (1 + \sqrt{A}) \frac{\exp(\beta \max_s \theta(s))}{\exp(\beta \min_s \theta(s)) \exp(\beta \min_s r(s))} \|z - c_1 \mathbf{1}_S\| \tag{46}$$

$$\leq \beta \alpha^d (1 + \sqrt{A}) \exp(\beta [\max_s \theta(s) - \min_s \theta(s) - \min_s r(s)]) \|z - c_1 \mathbf{1}_S\|. \tag{47}$$

Lastly, we prove that $\|z - c_1 \mathbf{1}_S\|$ is bounded independently of d. First, denote by $c = (c_1, \ldots, c_S)^\top$ and $\tilde{c} = (0, c_2, \ldots, c_S)^\top$.

Also, denote by Z the matrix with z_i as its i-th column. Now,

$$\|z - c_1\mathbf{1}_S\| = \|\sum_{i=2}^{S} c_i z_i\| \tag{48}$$
$$= \|Z\tilde{c}\| \tag{49}$$
$$\leq \|Z\|\|\tilde{c}\| \tag{50}$$
$$\leq \|Z\|\|c\| \tag{51}$$
$$= \|Z\|\|Z^{-1}z\| \tag{52}$$
$$\leq \|Z\|\|Z^{-1}\|, \tag{53}$$

where the last relation is due to z being a unit vector. All matrix norms here are l_2-induced norms.

Next, denote by W the matrix with w_i in its i-th column. Recall that in (33) we only defined $w_2, \ldots, w_S$. We now set $w_1 = \exp(\beta\Theta)$. Note that w_1 is linearly independent of $\{w_2, \ldots, w_S\}$ because of (33) together with the fact that $\mu^\top w_1 > 0$. We can now express the relation between Z and W by $Z = D^{-1}(\exp(\beta\Theta))W$. Substituting this in (53), we have

$$\|z - c_1\mathbf{1}_S\| \leq \|D^{-1}(\exp(\beta\Theta))W\|\|W^{-1}D(\exp(\beta\Theta))\| \tag{54}$$
$$\leq \|W\|\|W^{-1}\|\|D(\exp(\beta\Theta))\|\|D^{-1}(\exp(\beta\Theta))\|. \tag{55}$$

It further holds that

$$\|D(\exp(\beta\Theta))\| \leq \max_s \exp\left(\beta\theta(s)\right) \leq \max\{1, \exp[\beta \max_s \theta(s)])\}, \tag{56}$$

where the last relation equals 1 if $\theta(s) < 0$ for all s. Similarly,

$$\|D^{-1}(\exp(\beta\Theta))\| \leq \frac{1}{\min_s \exp\left(\beta\theta(s)\right)} \leq \frac{1}{\min\{1, \exp[\beta \min_s \theta(s)])\}}. \tag{57}$$

Furthermore, by the properties of the l_2-induced norm,

$$\|W\|_2 \leq \sqrt{S}\|W\|_1 \tag{58}$$
$$= \sqrt{S} \max_{1\leq i\leq S} \|w_i\|_1 \tag{59}$$
$$= \sqrt{S} \max\{\exp(\beta\Theta), \max_{2\leq i\leq S} \|w_i\|_1\} \tag{60}$$
$$\leq \sqrt{S} \max\{1, \exp[\beta \max_s \theta(s)], \max_{2\leq i\leq S} \|w_i\|_1)\}. \tag{61}$$

Lastly,

$$\|W^{-1}\| = \frac{1}{\sigma_{\min}(W)} \tag{62}$$
$$\leq \left(\prod_{i=1}^{S-1} \frac{\sigma_{\max}(W)}{\sigma_i(W)}\right) \frac{1}{\sigma_{\min}(W)} \tag{63}$$
$$= \frac{(\sigma_{\max}(W))^{S-1}}{\prod_{i=1}^{S} \sigma_i(W)} \tag{64}$$
$$= \frac{\|W\|^{S-1}}{|\det(W)|}. \tag{65}$$

The determinant of W is a sum of products involving its entries. To upper bound (65) independently of d, we lower bound its denominator by upper and lower bounds on the entries $[W]_{i,1}$ that are independent of d, depending on their sign:

$$\min\{1, \exp[\beta \min_s \theta(s)])\} \leq [W]_{i,1} \leq \max\{1, \exp[\beta \max_s \theta(s)])\}. \tag{66}$$

Using this, together with (53), (55), (56), (57), and (61), we showed that $\|z - c_1\mathbf{1}_S\|$ is upper bounded by a constant independent of d. This concludes the proof. □

A.9. Bias Estimates

Lemma A.2. *For any matrix A and $\hat{A}$,*

$$\hat{A}^k - A^k = \sum_{h=1}^{k} \hat{A}^{h-1}(\hat{A} - A)A^{k-h}.$$

Proof. The proof follows from first principles:

$$\sum_{h=1}^{k} \hat{A}^{h-1}(\hat{A} - A)A^{k-h} = \sum_{h=1}^{k} \hat{A}^{h-1}\hat{A}A^{k-h} - \sum_{h=1}^{k} \hat{A}^{h-1}AA^{k-h} \tag{67}$$

$$= \sum_{h=1}^{k} \hat{A}^{h}A^{k-h} - \sum_{h=1}^{k} \hat{A}^{h-1}A^{k-h+1} \tag{68}$$

$$= \hat{A}^k - A^k + \sum_{h=1}^{k-1} \hat{A}^{h}A^{k-h} - \sum_{h=2}^{k} \hat{A}^{h-1}A^{k-h+1} \tag{69}$$

$$= \hat{A}^k - A^k. \tag{70}$$

□

Henceforth, $\|\cdot\|$ will refer to $\|\cdot\|_\infty$, i.e. the induced infinity norm. Also, for brevity, we denote $\pi^{\mathrm{C}}_{d,\theta}$ and $\hat{\pi}^{\mathrm{C}}_{d,\theta}$ by π_θ and $\hat{\pi}_\theta$, respectively. Similarly, we use d_{π_θ} and $d_{\hat{\pi}_\theta}$ to denote $d_{\pi^{\mathrm{C}}_{d,\theta}}$ and $d_{\hat{\pi}^{\mathrm{C}}_{d,\theta}}$. As for the induced norm of the matrix P and its perturbed counterpart $\hat{P}$, which are of size $S \times A \times S$, we slightly abuse notation and denote $\|P - \hat{P}\| = \max_s\{\|P_s - \hat{P}_s\|\}$, where P_s is as defined in Section 2.

Definition A.3. Let ϵ be the maximal model mis-specification, i.e., $\max\{\|P - \hat{P}\|, \|r - \hat{r}\|\} = \epsilon$.

Lemma A.4. *Recall the definitions of R_s, P_s, R_{π_b} and P^{π_b} from Section 2, and respectively denote their perturbed counterparts by $\hat{R}_s, \hat{P}_s, \hat{R}_{\pi_b}$ and $\hat{P}^{\pi_b}$. Then, for ϵ defined in Definition A.3,*

$$\max\{\|R_s - \hat{R}_s\|, \|P_s - \hat{P}_s\|, \|R_{\pi_b} - \hat{R}_{\pi_b}\|, \|P^{\pi_b} - \hat{P}^{\pi_b}\|\} = O(\epsilon). \tag{71}$$

Proof. The proof follows easily from the fact that the differences above are convex combinations of $P - \hat{P}$ and $r - \hat{r}$. □

Lemma A.5. *Let π_θ be as in (5), and let $\hat{\pi}_\theta$ also be defined as in (5), but with R_s, P_s, P^{π_b} replaced by their perturbed counterparts $\hat{R}_s, \hat{P}_s, \hat{P}^{\pi_b}$ throughout. Then,*

$$\|\pi^{C}_{d,\theta} - \hat{\pi}^{C}_{d,\theta}\| = O(\beta d \gamma^{-d}\epsilon). \tag{72}$$

Proof. To prove the desired result, we work with (5) to bound the error between $R_s, P_s, P^{\pi_b}, R_{\pi_b}$ and their perturbed versions.

First, we apply Lemma A.2 together with Lemma A.4 to obtain that $\|(P^{\pi_b})^k - (\hat{P}^{\pi_b})^k\| = O(k\epsilon)$. Next, denote by M the argument in the exponent in (5), i.e.

$$M := \beta[C_{s,d} + P_s(P^{\pi_b})^{d-1}\Theta].$$

Similarly, let $\hat{M}$ be the corresponding perturbed sum that relies on $\hat{P}$ and $\hat{r}$. Combining the bounds from Lemma A.4, and using the triangle inequality, we have that $\|\hat{M} - M\| = O(\beta d \gamma^{-d}\epsilon)$. The factor γ^{-d} appears because $C_{s,d}$ includes the term γ^{-d} as shown in Lemma 4.2.

Eq. (5) states that the C-SoftTreeMax policy in the true environment is $\pi_\theta = \exp(M)/(1^\top \exp(M))$. Similarly define $\hat{\pi}_\theta$ using $\hat{M}$ for the approximate model. Then,

$$\hat{\pi}_\theta = (\pi_\theta \circ \exp(\hat{M} - M))1^\top \exp(M)/(1^\top \exp(\hat{M})),$$

where $\circ$ denotes element-wise multiplication. Using the above relation, we have that $\|\hat{\pi}_\theta - \pi_\theta\| = \|\pi_\theta\| \|\frac{\exp(\hat{M}-M)1^\top \exp(M)}{1^\top \exp(\hat{M})} - 1\|$. Using the relation $|e^x - 1| = O(x)$ as $x \to 0$, the desired result follows.

□

Theorem A.6. *Let ϵ be as in Definition A.3. Further let $\hat{\pi}^C_{d,\theta}$ being the corresponding approximate policy as given in Lemma 4.2. Then, the policy gradient bias is bounded by*

$$\left\| \frac{\partial}{\partial \theta} \left(\nu^\top V^{\pi_\theta}\right) - \frac{\partial}{\partial \theta} \left(\nu^\top V^{\hat{\pi}_\theta}\right) \right\| = \mathcal{O}\left(\frac{1}{(1-\gamma)^2} S \beta^2 d \gamma^{-d} \epsilon \right). \tag{73}$$

We first provide a proof outline for conciseness, and only after it the complete proof.

Proof outline. First, we prove that $\max\{\|R_s - \hat{R}_s\|, \|P_s - \hat{P}_s\|, \|R_{\pi_b} - \hat{R}_{\pi_b}\|, \|P^{\pi_b} - \hat{P}^{\pi_b}\|\} = \mathcal{O}(\epsilon)$. This follows from the fact that the differences above are suitable convex combinations of either the rows of $P - \hat{P}$ or $r - \hat{r}$. We use the above observation along with the definitions of $\pi^{\text{C}}_{d,\theta}$ and $\hat{\pi}^{\text{C}}_{d,\theta}$ given in (5) to show that $\|\pi^{\text{C}}_{d,\theta} - \hat{\pi}^{\text{C}}_{d,\theta}\| = O(\beta d \gamma^{-d} \epsilon)$. The proof for the latter builds upon two key facts: (a) $\|(P^{\pi_b})^k - (\hat{P}^{\pi_b})^k\| \leq \sum_{h=1}^k \|\hat{P}^{\pi_b}\|^{h-1} \|\hat{P}^{\pi_b} - P^{\pi_b}\| \|p^{\pi_b}\|^{k-h} = O(k\epsilon)$ for any $k \geq 0$, and (b) $|e^x - 1| = O(x)$ as $x \to 0$. Next, we decompose the LHS of (7) to get

$$\sum_s \left(\prod_{i=1}^4 X_i(s) - \prod_{i=1}^4 \hat{X}_i(s) \right) = \sum_s \sum_{i=1}^4 \hat{X}_1(s) \cdots \hat{X}_{i-1}(s) \left(X_i(s) - \hat{X}_i(s) \right) \times X_{i+1}(s) \cdots X_4(s),$$

where $X_1(s) = d_{\pi^{\text{C}}_{d,\theta}}(s) \in \mathbb{R}$, $X_2(s) = (\nabla_\theta \log \pi^{\text{C}}_{d,\theta}(\cdot|s))^\top \in \mathbb{R}^{S \times A}$, $X_3(s) = D(\pi^{\text{C}}_{d,\theta}(\cdot|s)) \in \mathbb{R}^{A \times A}$, $X_4(s) = Q^{\pi^{\text{C}}_{d,\theta}}(s, \cdot) \in \mathbb{R}^{A \times A}$, and $\hat{X}_1(s), \ldots, \hat{X}_4(s)$ are similarly defined with $\pi^{\text{C}}_{d,\theta}$ replaced by $\hat{\pi}^{\text{C}}_{d,\theta}$. Then, we show that, for $i = 1, \ldots, 4$, (i) $\|X_i(s) - \hat{X}_i(s)\| = O(\gamma^{-d}\epsilon)$ and (ii) $\max\{\|X_i\|, \|\hat{X}_i\|\}$ is bounded by problem parameters. From this, the desired result follows. □

Proof. We have

$$\begin{aligned}
&\frac{\partial}{\partial \theta} \left(\nu^\top V^{\pi_\theta}\right) - \frac{\partial}{\partial \theta} \left(\nu^\top V^{\pi'_\theta}\right) && (74)\\
&= \mathbb{E}_{s \sim d_{\pi_\theta}, a \sim \pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right] - \mathbb{E}_{s \sim d_{\hat{\pi}_\theta}, a \sim \hat{\pi}_\theta(\cdot|s)} \left[\nabla_\theta \log \hat{\pi}_\theta(a|s) Q^{\hat{\pi}_\theta}(s,a)\right] && (75)\\
&= \sum_{s,a} \left(d_{\pi_\theta}(s) \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) - d_{\hat{\pi}_\theta}(s) \hat{\pi}_\theta(a|s) \nabla_\theta \log \hat{\pi}_\theta(a|s) Q^{\hat{\pi}_\theta}(s,a) \right) && (76)\\
&= \sum_s \Big(d_{\pi_\theta}(s) (\nabla_\theta \log \pi_\theta(\cdot|s))^\top D(\pi_\theta(\cdot|s)) Q^{\pi_\theta}(s,\cdot) && (77)\\
&\quad - d_{\hat{\pi}_\theta}(s) (\nabla_\theta \log \hat{\pi}_\theta(\cdot|s))^\top D(\hat{\pi}_\theta(\cdot|s)) Q^{\hat{\pi}_\theta}(s,\cdot) \Big) && (78)\\
&= \sum_s \left(\prod_{i=1}^4 X_i(s) - \prod_{i=1}^4 \hat{X}_i(s) \right) && (79)\\
&= \sum_s \sum_{i=1}^4 \hat{X}_1(s) \cdots \hat{X}_{i-1}(s) \left(X_i(s) - \hat{X}_i(s) \right) X_{i+1}(s) \cdots X_4(s), && (80)
\end{aligned}$$

where $X_1(s) = d_{\pi_\theta}(s) \in \mathbb{R}$, $X_2(s) = (\nabla_\theta \log \pi_\theta(\cdot|s))^\top \in \mathbb{R}^{S \times A}$, $X_3(s) = D(\pi_\theta(\cdot|s)) \in \mathbb{R}^{A \times A}$, $X_4(s) = Q^{\pi_\theta}(s, \cdot) \in \mathbb{R}^{A \times A}$, and $\hat{X}_1(s), \ldots, \hat{X}_4(s)$ are similarly defined with π_θ replaced by $\hat{\pi}_\theta$.

Therefore,

$$\left\| \frac{\partial}{\partial \theta} \left(\nu^\top V^{\pi_\theta}\right) - \frac{\partial}{\partial \theta} \left(\nu^\top V^{\pi'_\theta}\right) \right\| \leq \left(\max_s \Gamma(s) \right) S, \tag{81}$$

where

$$\Gamma(s) = \| \sum_{i=1}^4 \hat{X}_1(s) \cdots \hat{X}_{i-1}(s) \left(X_i(s) - \hat{X}_i(s) \right) X_{i+1}(s) \cdots X_4(s) \|. \tag{82}$$

Next, since $d_{\pi_\theta}, d_{\hat{\pi}_\theta}, \pi_\theta$, and $\hat{\pi}_\theta$ are all distributions, we have

$$\max\{|X_1(s)|, |\hat{X}_1(s)|, |X_3(s,a)|, |\hat{X}_3(s,a)|\} \leq 1. \tag{83}$$

Separately, using Lemma 4.3, we have

$$\|X_2\| = \|\nabla_\theta \log \pi_\theta(a|s)\| \leq \beta(\|I_A\| + \|\mathbf{1}_A \pi_\theta^\top\|)\|P_s\|\|(P^{\pi_b})^{d-1}\|. \tag{84}$$

Since all rows of the above matrices have non-negative entries that add up to 1, we get

$$\|Y\| \leq 2\beta. \tag{85}$$

In the rest of the proof, we bound each of $\|X_1 - \hat{X}_1\|, \ldots, \|X_4 - \hat{X}_4\|$.

Finally,

$$\|X_4\| \leq \frac{1}{1-\gamma}. \tag{86}$$

Similarly, the same bounds hold for $\hat{X}_1, \hat{X}_2, \hat{X}_3$ and $\hat{X}_4$.

From, we have

$$\begin{aligned}
\|X_1 - \hat{X}_1\| &\leq (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \|\nu^\top (P^{\pi_\theta})^t - \nu^\top (P^{\hat{\pi}_\theta})^t\| && (87)\\
&\leq (1-\gamma)\|\nu\| \sum_{t=0} \gamma^t t d\epsilon && (88)\\
&\leq (1-\gamma) d\epsilon \sum_{t=0}^{\infty} \gamma^t t && (89)\\
&= \frac{\gamma d\epsilon}{1-\gamma}. && (90)
\end{aligned}$$

The last relation follows from the fact that $(1-\gamma)^{-1} = \sum_{t=0}^{\infty} \gamma^t$, which in turn implies

$$\gamma \frac{\partial}{\partial \gamma}\left(\frac{1}{1-\gamma}\right) = \sum_{t=0}^{\infty} t\gamma^t. \tag{91}$$

From Lemma A.5, it follows that

$$\|X_3 - \hat{X}_3\| = O(\beta d\epsilon). \tag{92}$$

Next, recall that from Lemma 4.3 that

$$X_2(s,\cdot) = \beta\left[I_A - \mathbf{1}_A(\pi_\theta)^\top\right] P_s \left(P^{\pi_b}\right)^{d-1}.$$

Then,

$$\begin{aligned}
\|X_2(s,\cdot) - \hat{X}_2(s,\cdot)\| \leq &\|\beta\left[I_A - \mathbf{1}_A(\pi_\theta)^\top\right] P_s\| \|\left(P^{\pi_b}\right)^{d-1} - \left(\hat{P}^{\pi_b}\right)^{d-1}\| && (93)\\
&+ \|\beta\left[I_A - \mathbf{1}_A(\pi_\theta)^\top\right]\| \|P_s - \hat{P}_s\| \|\left(\hat{P}^{\pi_b}\right)^{d-1}\| && (94)\\
&+ \beta\|\mathbf{1}_A(\pi_\theta)^\top - \mathbf{1}_A(\hat{\pi}_\theta)^\top\| \|\hat{P}_s\left(\hat{P}^{\pi_b}\right)^{d-1}\|. && (95)
\end{aligned}$$

Following the same argument as in (85) and applying Lemma A.2, we have that (93) is $O(\beta d\epsilon)$. Similarly, from the argument of (85), Eq. (94) is $O(\beta\epsilon)$. Lastly, (95) is $O(\beta d\epsilon)$ due to Lemma A.5. Putting the above three terms together, we have that

$$\|X_2(s,\cdot) - \hat{X}_2(s,\cdot)\| = O(\beta d\epsilon). \tag{96}$$

Since the state-action value function satisfies the Bellman equation, we have

$$Q^{\pi_\theta} = r + \gamma P Q^{\pi_\theta} \tag{97}$$

and

$$Q^{\hat{\pi}_\theta} = \hat{r} + \gamma \hat{P} Q^{\hat{\pi}_\theta}. \tag{98}$$

Consequently,

$$\|Q^{\pi_\theta} - Q^{\hat{\pi}_\theta}\| \leq \|r - \hat{r}\| + \gamma \|PQ^{\pi_\theta} - PQ^{\hat{\pi}_\theta}\| + \gamma \|PQ^{\hat{\pi}_\theta} - \hat{P}Q^{\hat{\pi}_\theta}\| \tag{99}$$

$$\leq \epsilon + \gamma \|P\| \|Q^{\pi_\theta} - Q^{\hat{\pi}_\theta}\| + \gamma \|P - \hat{P}\| \|Q^{\hat{\pi}_\theta}\| \tag{100}$$

$$\leq \epsilon + \gamma \|Q^{\pi_\theta} - Q^{\hat{\pi}_\theta}\| + \frac{\gamma}{1-\gamma}\epsilon, \tag{101}$$

which finally shows that

$$\|X_4 - \hat{X}_4\| = \|Q^{\pi_\theta} - Q^{\hat{\pi}_\theta}\| \leq \frac{\epsilon}{(1-\gamma)^2}. \tag{102}$$

□

B. Experiments

B.1. Implementation Details

The environment engine is the highly efficient Atari-CuLE (Dalton et al., 2020), a CUDA-based version of Atari that runs on GPU. Similarly, we use Atari-CuLE for the GPU-based breadth-first TS as done in (Dalal et al., 2021).

We train SoftTreeMax for depths $d = 1 \ldots 8$, with a single worker. We use five seeds for each experiment.

For the implementation, we extend Stable-Baselines3 (Raffin et al., 2019) with all parameters taken as default from the original PPO paper (Schulman et al., 2017). For depths $d \geq 3$, we limited the tree to a maximum width of 1024 nodes and pruned non-promising trajectories in terms of estimated weights. Since the distributed PPO baseline advances significantly faster in terms of environment steps, for a fair comparison, we ran all experiments for one week on the same machine and use the wall-clock time as the x-axis. We use Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz equipped with one NVIDIA Tesla V100 32GB.

B.2. GPU-Based Tree Expansion Implementation

Figure 4 illustrates the GPU-based tree expansion mechanism used in our implementation. We achieve efficient parallelization by duplicating and concatenating all states in the current level of the tree with each possible action, then advancing them simultaneously with a single forward pass through the simulator.

The implementation follows these steps:

1. Start with a root state s_0 and expand it across all actions $a \in \mathcal{A}$.
2. For each depth level t from 1 to d:
   - Collect all states s_t from the previous level.
   - Duplicate each state for each action to create a batch of state-action pairs.
   - Submit the entire batch to the GPU simulator in a single forward pass.
   - Collect the resulting next states and corresponding rewards.
   - If $t < d$ and pruning is enabled, select the top-k most promising branches based on accumulated rewards.
3. For the leaf states (at depth d), compute the neural network outputs $W_\theta(s_d)$.
4. Combine the accumulated rewards along each trajectory with the corresponding leaf state values.
5. Compute the final policy logits according to Equation (2) and apply the softmax operation.

This parallel implementation allows us to efficiently explore a larger number of trajectories compared to sequential tree expansion, making deeper tree depths practically feasible.

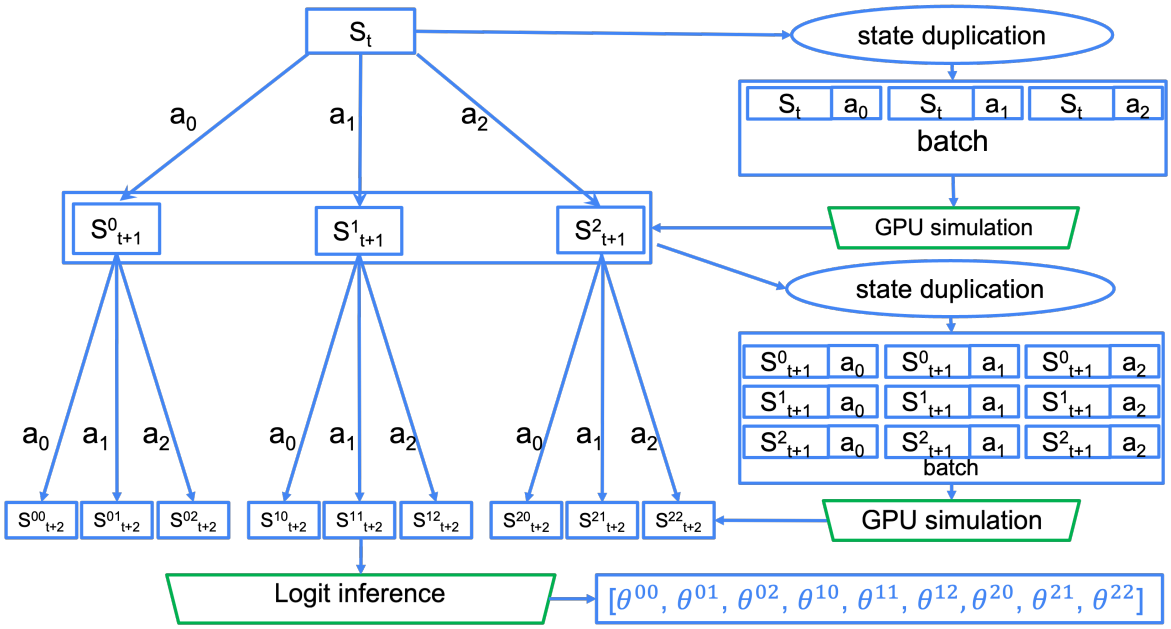

Figure 4. A diagram of the tree expansion used by SoftTreeMax. In every step, the states in the current level of the tree are duplicated and concatenated with each possible action. The resulting state-action pairs are then fed as a batch to the GPU simulator to generate the next level of states. Finally, the states of the last level d are inserted into the neural network W_θ and the logits are computed using the corresponding rewards along each trajectory.

Algorithm 1 C-SoftTreeMax

Input: GPU environment $\mathcal{G}$, network θ, depth d
Init tensors: state $\bar{S} = [s]$, action $\bar{A}_0 = [0, 1, 2, .., A-1]$, reward $\bar{R} = [0]$
for $i_d = 0$ **to** $d-1$ **do**
  $\bar{S} \leftarrow \bar{S} \times A$, $\bar{R} \leftarrow \bar{R} \times A$ {Replicate state and reward tensors A times}
  $\bar{r}, \bar{S}' = \mathcal{G}([\bar{S}, \bar{A}])$ {Feed $[\bar{S}, \bar{A}]$ to simulator and advance}
  $\bar{R} \leftarrow \bar{R} + \gamma^{i_d}\bar{r}$, $\bar{S} \leftarrow \bar{S}'$ {Accumulate discounted reward }
  $\bar{A} \leftarrow \bar{A} \times A$ {Replicate action tensor A times}
end for
$l_{s,a} \leftarrow \text{Average}^{\pi_b}_{A_0=a}(\bar{R} + \gamma^d\theta(\bar{S}))/\gamma^d$ {Weighted average induced by π_b}
Return $\pi(a|s_0) \propto \exp[\beta l_{s,a}(d;\theta)]$ {Return optimal action at the root}

B.3. Algorithms

This section provides the pseudocode for our SoftTreeMax implementation. Algorithm 1 details the C-SoftTreeMax policy computation, which efficiently utilizes GPU parallelization to perform tree expansion. Algorithm 2 shows how SoftTreeMax integrates with the PPO algorithm, distinguishing the usage of our new policy in red.

Note that in Algorithm 2, we use Generalized Advantage Estimation (GAE) with $\lambda = 0.95$ for calculating advantage estimates, which is the standard configuration in the stable-baselines3 PPO implementation that we build upon.

B.4. Time-Based Training Curves

We provide the training curves in Figure 5. For brevity, we exclude a few of the depths from the plots. As seen, there is a clear benefit for SoftTreeMax over distributed PPO with the standard softmax policy. In most games, PPO with the SoftTreeMax policy shows very high sample efficiency: it achieves higher episodic reward although it observes much less episodes, for the same running time.

Algorithm 2 SoftTreeMax-PPO

Initialize policy parameters θ_0
Initialize value function parameters ϕ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy π_{d,θ_k} from Algorithm 1
  Compute rewards-to-go $\hat{R}_t$
  Compute advantage estimates $\hat{A}_t$ using GAE with $\lambda = 0.95$
  for each epoch **do**
    for each minibatch **do**
      Compute policy ratio $r_t(\theta) = \frac{\pi_{d,\theta}(a_t|s_t)}{\pi_{d,\theta_k}(a_t|s_t)}$
      Compute clipped surrogate objective:
      $L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$
      Update θ with gradient step on $L^{CLIP}(\theta)$
      Compute value function loss: $L^{VF}(\phi) = (V_\phi(s_t) - \hat{R}_t)^2$
      Update ϕ with gradient step on $L^{VF}(\phi)$
    end for
  end for
end for

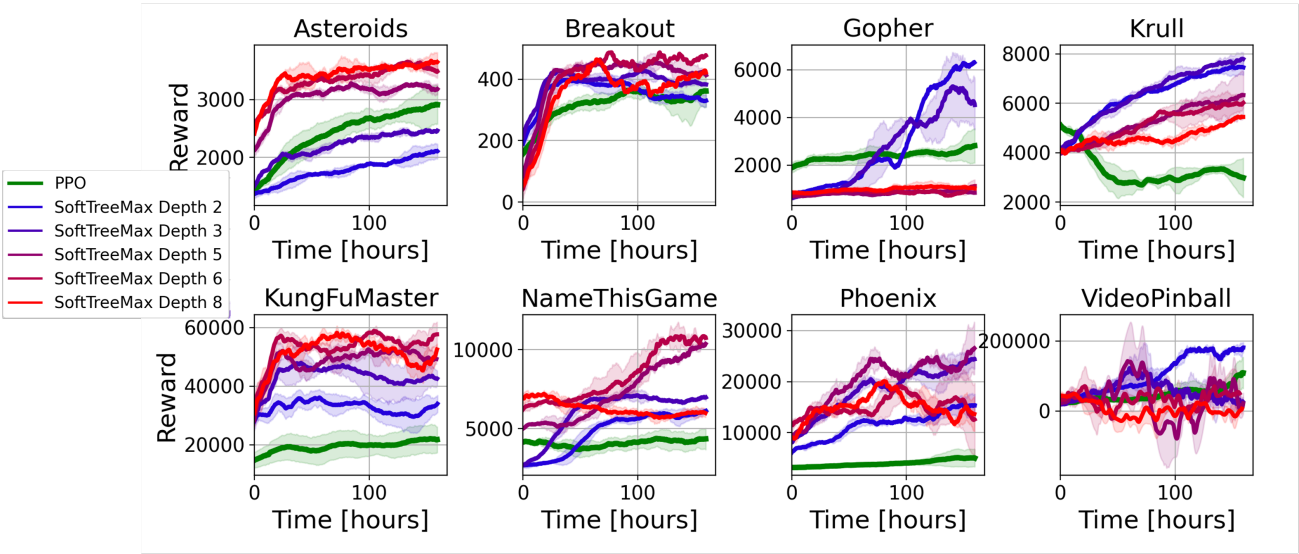

Figure 5. **Training curves: GPU SoftTreeMax (single worker) vs PPO (256 GPU workers).** The plots show average reward and standard deviation over 5 seeds. The x-axis is the wall-clock time. The runs ended after one week with varying number of time-steps. The training curves correspond to the evaluation runs in Figure 3.

B.5. Step-Based Training Curves

In Figure 6 we also provide the same convergence plots where the x-axis is now the number of online interactions with the environment, thus excluding the tree expansion complexity. As seen, due to the complexity of the tree expansion, less steps are conducted during training (limited to one week) as the depth increases. In this plot, the monotone improvement of the reward with increasing tree depth is noticeable in most games.

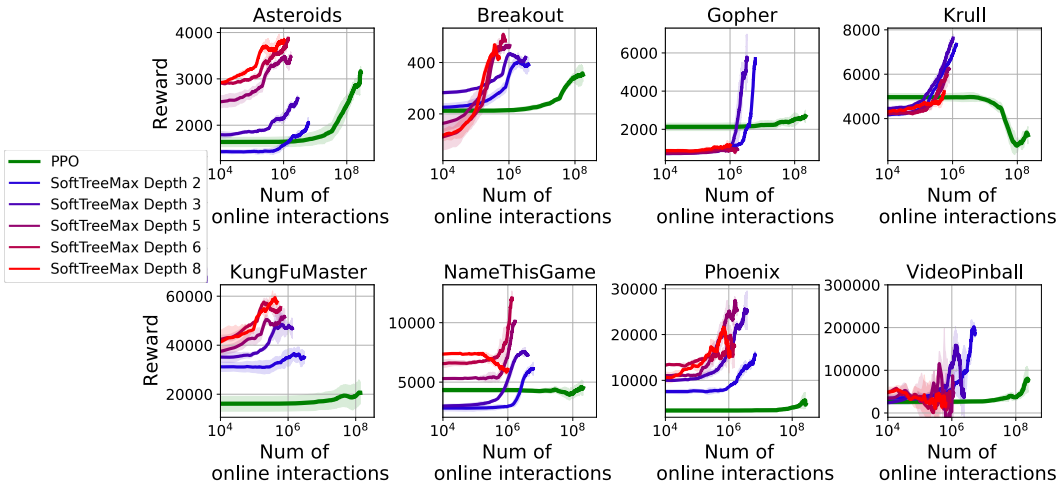

Figure 6. **Training curves: GPU SoftTreeMax (single worker) vs PPO (256 GPU workers).** The plots show average reward and standard deviation over 5 seeds. The x-axis is the number of online interactions with the environment. The runs ended after one week with varying number of time-steps. The training curves correspond to the evaluation runs in Figure 3.

We note that not for all games we see monotonicity. Our explanation for this phenomenon relates to how immediate reward contributes to performance compared to the value. Different games benefit differently from long-term as opposed to short-term planning. Games that require longer-term planning need a better value estimate. A good value estimate takes longer to obtain with larger depths, in which we apply the network to states that are very different from the ones observed so far in the buffer (recall that as in any deep RL algorithm, we train the model only on states in the buffer). If the model hasn't learned a good enough value function yet, and there is no guiding dense reward along the trajectory, the policy becomes noisier, and can take more steps to converge – even more than those we run in our week-long experiment.

For a concrete example, let us compare Breakout to Gopher. Inspecting Fig. 6, we observe that Breakout quickly (and monotonically) gains from large depths since it relies on the short term goal of simply keeping the paddle below the moving ball. In Gopher, however, for large depths (¿=5), learning barely started even by the end of the training run. Presumably, this is because the task in Gopher involves multiple considerations and steps: the agent needs to move to the right spot and then hit the mallet the right amount of times, while balancing different locations. This task requires long-term planning and thus depends more strongly on the accuracy of the value function estimate. In that case, for depth 5 or more, we would require more train steps for the value to "kick in" and become beneficial beyond the gain from the reward in the tree.

The figures above convey two key observations that occur for at least some non-zero depth: (1) The final performance with the tree is better than PPO (Fig. 3); and (2) the intermediate step-based results with the tree are better than PPO (Fig. 6). This leads to our main takeaway from this work — there is no reason to believe that the vanilla policy gradient algorithm should be better than a multi-step variant. Indeed, we show that this is not the case.

C. Further discussion

C.1. The case of $\lambda_2(P^{\pi_b}) = 0$

When P^{π_b} is rank one, it is not only its variance that becomes 0, but also the norm of the gradient itself (similarly to the case of $d \to \infty$). Note that such a situation will happen rarely, in degenerate MDPs. This is a local minimum for SoftTreeMax and it would cause the PG iteration to get stuck, and to the optimum in the (desired but impractical) case where π_b is the optimal policy. However, a similar phenomenon was also discovered in the standard softmax with deterministic policies: $\theta(s,a) \to \infty$ for one a per s. PG with softmax would suffer very slow convergence near these local equilibria, as observed in (Mei et al., 2020a). To see this, note that the softmax gradient is $\nabla_\theta \log \pi_\theta(a|s) = e_a - \pi_\theta(\cdot|s)$, where $e_a \in [0,1]^A$ is the vector with 0 everywhere except for the a-th coordinate. I.e., it will be zero for a deterministic policy. SoftTreeMax avoids these local optima by integrating the reward into the policy itself (but may get stuck in another, as discussed above).