SOFTTREEMAX: POLICY GRADIENT WITH TREE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy-gradient methods are widely used for learning control policies. They can be easily distributed to multiple workers and reach state-of-the-art results in many domains. Unfortunately, they exhibit large variance and subsequently suffer from high-sample complexity since they aggregate gradients over entire trajectories. At the other extreme, planning methods, like tree search, optimize the policy using single-step transitions that consider future lookahead. These approaches have been mainly considered for value-based algorithms. Planning-based algorithms require a forward model and are computationally intensive at each step, but are more sample efficient. In this work, we introduce SoftTreeMax, the first approach that integrates tree-search into policy gradient. Traditionally, gradients are computed for single state-action pairs. Instead, our tree-based policy structure leverages all gradients at the tree leaves in each environment step. This allows us to reduce the variance of gradients by three orders of magnitude and to benefit from better sample complexity compared with standard policy gradient. On Atari, SoftTreeMax demonstrates up to 5x better performance in faster run-time compared with distributed PPO.

1 INTRODUCTION

Nowadays, Policy Gradient (PG; Sutton et al. 1999) methods for Reinforcement Learning (RL; Kaelbling et al. 1996) are often the first choice for environments that allow numerous interactions at a fast pace. Their success is attributed to several reasons: they can be easily distributed to multiple workers, require no assumptions on an underlying value function, have both on-policy and off-policy variants, and more. Despite their popularity, PG algorithms are also notoriously unstable because they compute gradients 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 were proposed to mitigate this instability. These include baseline subtraction (Greensmith et al., 2004; Weaver & Tao, 2001; Thomas & Brunskill, 2017; Wu et al., 2018), anchor-point averaging (Papini et al., 2018), and other variance reduction techniques (Papini et al., 2018; Zhang et al., 2021; Shen et al., 2019; Pham et al., 2020).

Another family of algorithms that achieved state-of-the-art results in several challenging domains is based on expanding possible future trajectories using tree search (TS; Coulom 2006; Silver 2009). Prominent examples from this family include UCT (Gelly & Wang, 2006), Dreamer (Hafner et al., 2019), and MuZero (Schrittwieser et al., 2020). TS-based algorithms rely on a forward model (accurate or learned) and substantially reduce sample complexity by offloading the information gain from sampling to planning. Given a forward modhe main weakness of TS-based methods is that it is difficult to parallelize the learning process because expanding the tree is computationally expensive. However, recent works proposed to leverage recent advances in GPU-based simulation and deep model learning for expanding the tree in parallel (Dalal et al., 2021; Rosenberg et al., 2022).

In this work, we build on top of those recent advances in parallel search and combine PG with TS by introducing a parameterized policy that incorporates tree expansion. Our SoftTreeMax policy replaces the standard policy-logits of an action with the sum of all logits at the tree leaves that originate from that action. The inference process thus amounts to a fast and efficient breadth-first search followed by vectorized neural network inference, all in the GPU. During training, all gradients at the tree leaves are used for updating the policy, thus significantly mitigating the high variance of PG.

Our approach addresses a major limitation of TS methods, that their computational complexity grows exponentially with the size of the tree. We reduce the complexity by limiting the width of the tree; we do so by sampling only the most promising nodes at each level. This allows us to efficiently search Atari environments with trees that are 8-level deep. The fast search mechanism is enabled thanks to a GPU simulator that allows running multiple copies of Atari in parallel Dalton et al. (2020). We then integrate the resulting SoftTreeMax into the widely used PPO (Schulman et al., 2017) and compare it to PPO in its distributed variant. For a fair comparison, we also run the distributed PPO baseline with the parallel GPU emulator by (Dalton et al., 2020). In all tested Atari games, our results outperform the baseline and obtain up to 5x more reward. As we show in Section 4, the associated gradient variance is three orders of magnitude lower in all games.

2 PRELIMINARIES

We follow the standard notation by Puterman (2014). We consider a discounted Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, 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, and $\gamma \in (0, 1)$ is the discount factor.

Let $\pi : \mathcal{S} \to \mathcal{A}$ be a stationary policy, and $V^\pi \in \mathbb{R}^S$ be the value function of π defined by $V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \mid s_0 = s\right]$.

Our goal is to find the optimal policy $\pi^\star$ such that, for every $s \in \mathcal{S}$,

$$V^\star(s) = V^{\pi^\star}(s) = \max_{\pi:\mathcal{S}\to\mathcal{A}} V^\pi(s).$$

When the state space is very large, or continuous, a common practice is to use function approximation to represent the value function, the policy, or even the forward-model. Following the prominent success of deep RL Mnih et al. (2015) in recent years, deep neural networks are used nowadays almost exclusively in practice. Depending on the RL algorithm, a loss function is defined and gradients on the network weights can be calculated.

2.1 POLICY GRADIENT

PG schemes seek to maximize the cumulative reward as a function of the parameterized policy $\pi_\theta(a|s)$ by performing gradient steps on θ. For an observed trajectory $\tau = (s_0, a_0, r_0, s_1, \ldots, s_T, a_T, r_T, s_{T+1})$, let $r(\tau)$ be the cumulative reward of a trajectory τ, and $\pi_\theta(\tau)$ to be probability of observing trajectory τ under policy π_θ:

$$\pi_\theta(\tau) = \Pr(s_0) \cdot \prod_{t=0}^{T} \pi_\theta(a_t|s_t) \Pr(s_{t+1}, r_t|s_t, a_t). \tag{1}$$

Using the likelihood ratio trick, we have

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[r(\tau)\right] = \mathbb{E}_{\tau\sim\pi_\theta}\left[r(\tau)\nabla_\theta \log \pi_\theta(\tau\right] = \mathbb{E}_{\tau\sim\pi_\theta}\left[r(\tau)\sum_{t=0}^{T} \log \pi_\theta(a_t|s_t)\right]. \tag{2}$$

Subsequently, multiple trajectories can be sampled to apply stochastic gradient descent on the policy parameters θ forming the backbone of most PG algorithms.

When the action space is discrete, a commonly used parameterized policy is the soft-max applied over learned weights corresponding to each state-action pair:

$$\pi_\theta(a|s) \propto \exp\left(w_\theta(s, a)\right), \tag{3}$$

where the weights are the output of a neural network with A heads, where the state is the input. Soft-max policies became a canonical part of PG (Zhang et al., 2021; Mei et al., 2020b; Li et al., 2021; Schulman et al., 2017; Haarnoja et al., 2018). Some alternatives have been proposed to soft-max (Mei et al., 2020a; Miahi et al., 2021) with relatively rare usage in practice. Due to its popularity, we focus on a tree extension to the soft-max policy. However, the method proposed here is general and can be applied trivially to other discrete or continuous parameterized policies.

2.2 TREE SEARCH

Planning with a TS is the process of using a forward model to consider possible future trajectories and decide on the best action at the root. One famous such algorithm is Monte-Carlo TS (MCTS; Browne et al. 2012) used in AlphaGo Silver et al. (2016) and MuZero Schrittwieser et al. (2020). Other principal algorithms such as Value Iteration, Policy Iteration and DQN were also shown to give an improved performance with a tree search extensions (Efroni et al., 2019; Dalal et al., 2021). The applicability of these extensions largely benefits from the GPU-based simulator which allows spanning the tree efficiently at each step. This coincides with recently released simulators such as Atari-CuLE (Dalton et al., 2020), IsaacGym (Makoviychuk et al., 2021) and Brax Freeman et al. (2021) that support advancing multiple environments simultaneously on the GPU. When such a simulator is out of reach, it is also possible learn a forward model Ha & Schmidhuber (2018); Kim et al. (2020); Schrittwieser et al. (2020).

3 SOFTTREEMAX

We define a new parametric policy, SoftTreeMax, that operates as follows:

1. Given a state s_0, expand a TS up to depth d to obtain H trajectories:
$$\{(s_0^h, a_0^h, r_0^h, s_1^h, a_1^h, r_1^h, ..., a_{d-1}^h, r_{d-1}^h, s_d^h, a_d^h\}_{h=1}^H \tag{4}$$
The actions can be chosen in any manner, where in our experiments we consider all actions exhaustively and prune under-performing branches so we can reach bigger depths.
2. Feed all state-action pairs at the leaves into a policy network w_θ at each branch to obtain the logits $\{\sum_{t=0}^{d-1} \gamma^t r_t^h + \gamma^d w_\theta(s_d^h, a_d^h)\}_{h=1}^H$ that correspond to each action at the root a_0^h.
3. The probability to sample an action $a_0 = a$ is a soft-max over all trajectories starting with $a_0^h = a$. Mathematically:
$$\pi_{\text{SoftTreeMax}}(a|s;\theta) \propto \sum_{h:a_0^h=a} \exp\left[\beta\left(\sum_{t=0}^{d-1}\gamma^t r_t^h + \gamma^d w_\theta(s_d^h, a_d^h)\right)\right], \tag{5}$$
where β is an inverse temperature parameter controlling the exploration intensity.

Remark 1. SoftTreeMax is a natural planning-based generalization of soft-max: For $d = 0$, it reduces to the standard soft-max. When $d \to \infty$, the total weight of a trajectory is its infinite-horizon cumulative discounted reward.

Remark 2. SoftTreeMax considers the sum of all action values at the leaves, corresponding to Q-function based algorithms. Instead, one could use a single-headed network w_θ for correspondence with value-function based algorithms.

Remark 3. We add the discounted cumulative reward to the calculated weights. This exploits the additional reward information obtained from expanding the tree. The resulting sum also corresponds to the Boltzmann exploration practice Sutton et al. (1999) where the exponent mimics the Q-function.

Remark 4. When $\beta \to 0$ and the same number of trajectories is sampled per action, the policy samples actions uniformly (exploration). When $\beta \to \infty$, the policy greedily optimizes for the best trajectory (exploitation).

An illustration of the SoftTreeMax policy is given in Figure 1. Like other TS methods, SoftTreeMax requires a forward model that can be used to expand the tree. For an exhaustive search, the complexity of expanding the tree grows exponentially with the tree depth d. However, thanks to our pruning technique, we reduce the complexity to be linear in d.

When the gradient $\nabla_\theta \log \pi_\theta$ is calculated at each time step, it updates w_θ for all state-action pairs in the leaves, similarly to Siamese networks Bertinetto et al. (2016). Subsequently, the network is updated even for states that were not observed and actions that were not chosen throughout the simulation trajectory. This effect is essentially a variance reduction technique. As we show in Section 4, it indeed significantly reduces the noise of the PG process and leads to faster convergence and higher reward.

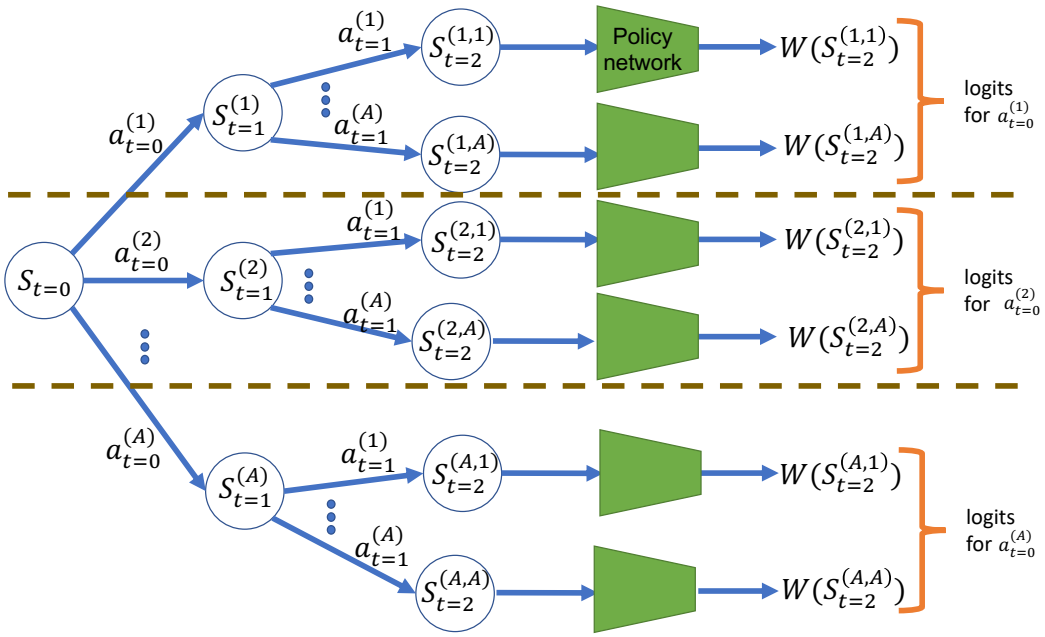

Figure 1: **SoftTreeMax policy**. For simplicity, we visualize the exhaustive TS that expands all actions at each state up to depth d ($= 2$ here). The leaf state of every trajectory is used as input to the policy network with a different head per action; for simpler visualization we aggregate all actions in the last level. The output is then added to trajectory's cumulative reward as described in equation 5 to form the logit of each action's probability.

4 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 workers in parallel. In a hyperparameter search, we found this number of workers to be the best in terms of run-time. 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 then train SoftTreeMax for depths $d = 1 \dots 8$, with a single worker. For brevity, we exclude a few of the depths from the plots. 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). We will release the code upon publication. 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 1 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.

We provide the training curves in Figure 2. As seen, there is a clear benefit for SoftTreeMax over distributed PPO with the standard soft-max policy. In most games, PPO with the SoftTreeMax policy shows very high sample efficiency: it achieves higher episodic reward even though it observes much less episodes, for the same running time.

In Figure 3, we show the gradient variance during training, as computed for all samples in the batch during the PPO training phase. As seen, the gradient variance of SoftTreeMax is consistently lower by three orders of magnitude than in PPO. We also observe a correlation between the gradient variance and the TS depth – in most games, the variance monotonically reduces as the depth increases.

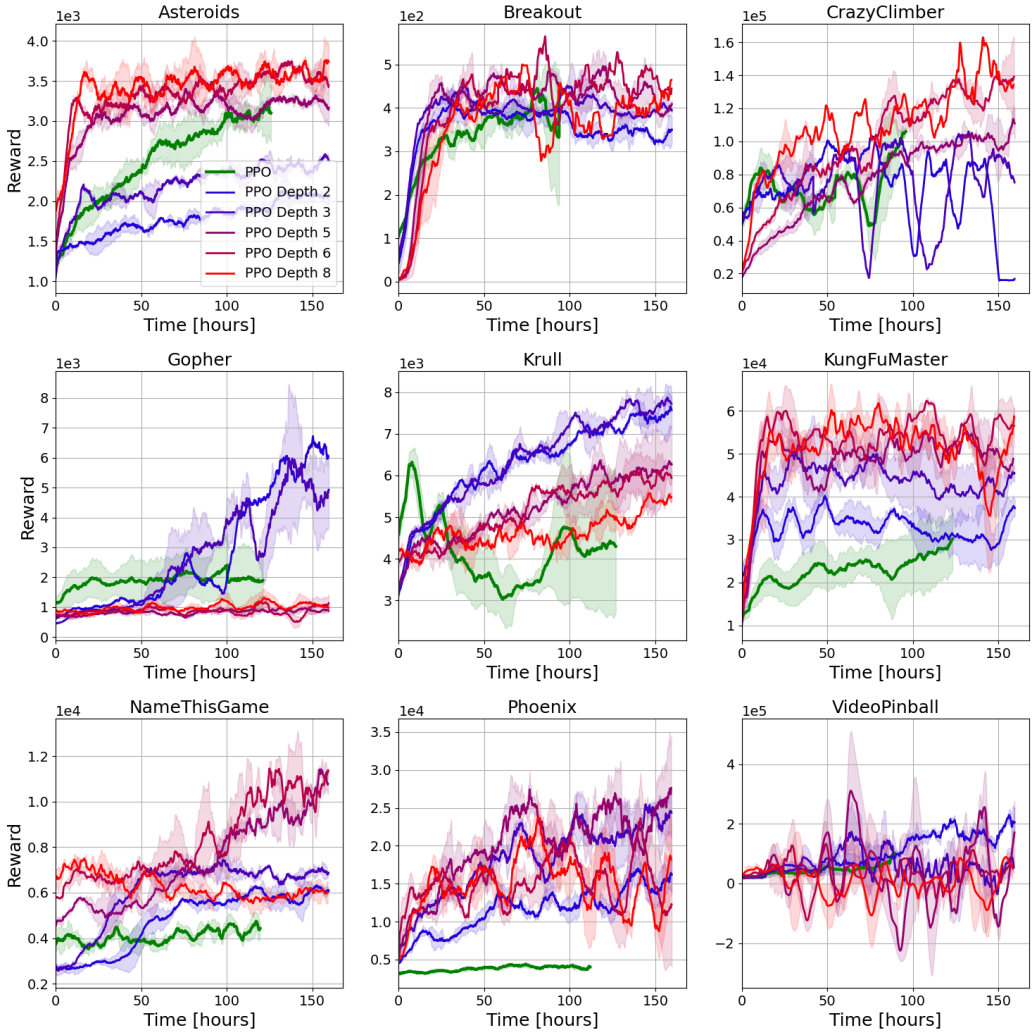

Figure 2: **Training curves: SoftTreeMax (single worker) vs PPO (256 workers).** The plots show average reward and std over five seeds. The x-axis is the wall-clock time. The maximum time-steps given were 200M, which the standard PPO finished in less than one week of running.

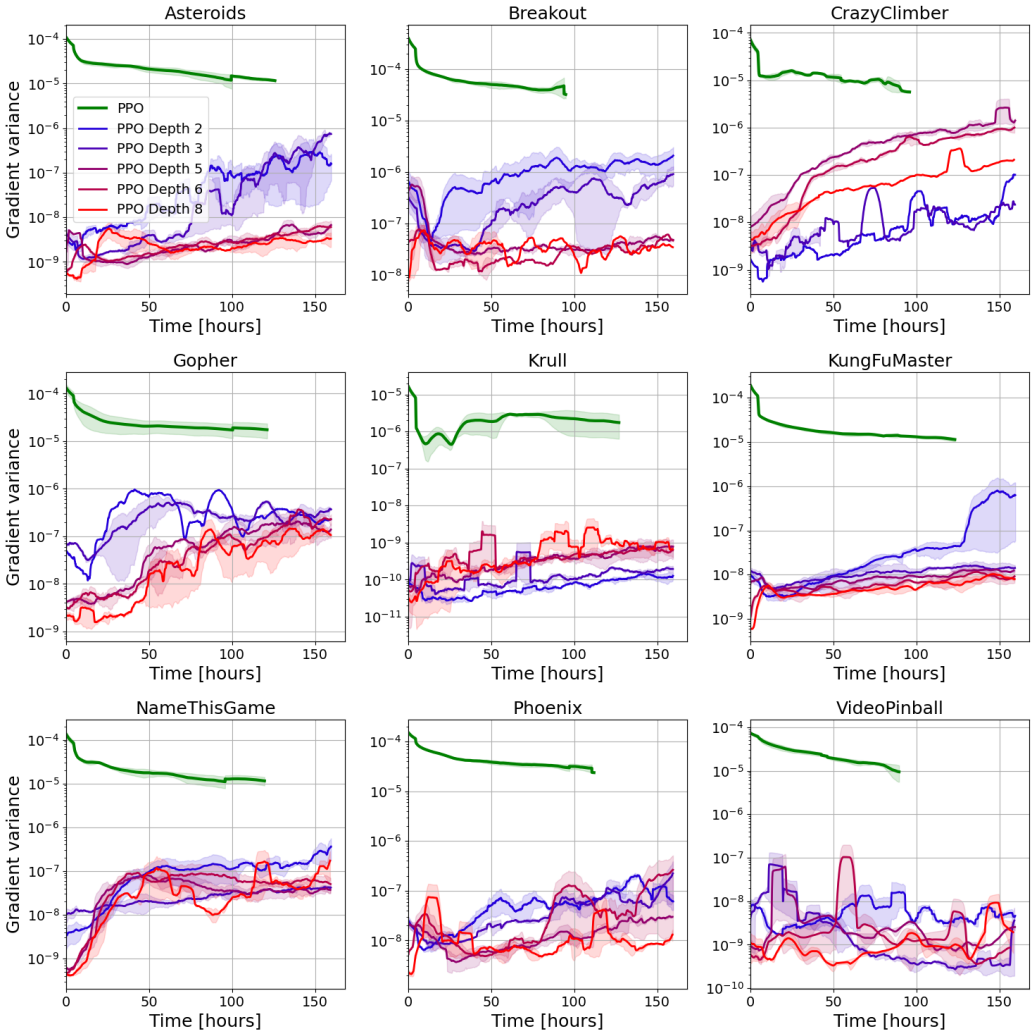

Figure 3: **Gradient variance: SoftTreeMax (single worker) vs PPO (256 workers).** The results correspond to the same runs as in Figure 2. Note the y-axis is in log-scale.

5 DISCUSSION

Planning in RL is typically conducted with value-based algorithms due its seamless integration with the bellman operator, leaving aside the popular class of policy gradient methods. In this work, we introduced for the first time a tree-search approach for policy gradient. We show why our SoftTreeMax is essentially a variance reduction technique. Mitigating the known sample-inefficiency issue, it achieves better performance than the widely used PPO with multiple workers and soft-max policy. Our method can be further applied to continuous control tasks, or in tasks where the forward model is learned with some estimation error. Other possible future directions are to study the theoretical implications of SoftTreeMax on the convergence rate of policy gradient, or to extend it to adaptively changing depths.

REFERENCES

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

Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pp. 850–865. Springer, 2016.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

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

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

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. How to combine tree-search methods in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3494–3501, 2019.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. 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.

Sylvain Gelly and Yizao Wang. Exploration exploitation in go: Uct for monte-carlo go. In *NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop*, 2006.

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

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

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

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. Learning to simulate dynamic environments with gamegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1231–1240, 2020.

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

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

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Jincheng Mei, Chenjun Xiao, Bo Dai, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. Escaping the gravitational pull of softmax. *Advances in Neural Information Processing Systems*, 33:21130–21140, 2020a.

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

Erfan Miahi, Revan MacQueen, Alex Ayoub, Abbas Masoumzadeh, and Martha White. Resmax: An alternative soft-greedy operator for reinforcement learning. 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In *International conference on machine learning*, pp. 4026–4035. PMLR, 2018.

Nhan Pham, Lam Nguyen, Dzung Phan, Phuong Ha Nguyen, Marten Dijk, and Quoc Tran-Dinh. A hybrid stochastic policy gradient algorithm for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 374–385. PMLR, 2020.

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

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

Aviv Rosenberg, Assaf Hallak, Shie Mannor, Gal Chechik, and Gal Dalal. Planning and learning with adaptive lookahead. *arXiv preprint arXiv:2201.12403*, 2022.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

Zebang Shen, Alejandro Ribeiro, Hamed Hassani, Hui Qian, and Chao Mi. Hessian aided policy gradient. In *International conference on machine learning*, pp. 5729–5738. PMLR, 2019.

David Silver. Reinforcement learning and simulation-based search in computer go. 2009.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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

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

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018.

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

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