On All-Action Policy Gradients

Michal Nauman[1,2]
nauman.mic@gmail.com

Marek Cygan[1,3]
cygan@mimuw.edu.pl

[1]*Informatics Institute, University of Warsaw*
[2]*IDEAS National Centre for Research and Development*
[3]*NoMagic.ai*

Abstract

In this paper, we analyze the variance of stochastic policy gradient with many action samples per state (all-action SPG). We decompose the variance of SPG and derive an optimality condition for all-action SPG. The optimality condition shows when all-action SPG should be preferred over single-action counterpart and allows to determine a variance-minimizing sampling scheme in SPG estimation. Furthermore, we propose dynamics-all-action (DAA) module, an augmentation that allows for all-action sampling without manipulation of the environment. DAA addresses the problems associated with using a Q-network for all-action sampling and can be readily applied to any on-policy SPG algorithm. We find that using DAA with a canonical on-policy algorithm (PPO) yields better sample efficiency and higher policy returns on a variety of continuous action environments.

1 Introduction

Stochastic policy gradient (SPG) is a method of optimizing stochastic policy through gradient ascent in the context of reinforcement learning (RL) [43, 39, 28, 37]. When paired with powerful function approximators, SPG-based algorithms have proven to be one of the most effective methods for achieving optimal performance in Markov Decision Processes (MDPs) with unknown transition dynamics [35, 32]. Unfortunately, exact calculation of the gradient is unfeasible and thus the objective has to be estimated [39]. Resulting variance is known to impact the learning speed, as well as performance of the trained agent [19, 41].

On-policy sample efficiency (ie. the amount of required interactions with the environment to achieve certain performance) is particularly affected by variance, as the gradient has to be evaluated over long trajectories to achieve sufficient quality of the SPG estimate [25]. As such, a variety of methods for on-policy SPG variance reduction have been proposed. Method of most prominence, baseline variance reduction, has been shown to improve algorithms stability and became indispensable to contemporary SPG implementations [28, 34]. Alternative approaches include Q-value bootstrapping [10], reducing the effect of long-horizon stochasticity via small discount [3], increasing number of samples via parallel agents [25] or using all-action estimators [1, 21, 29, 6].

In all-action SPG (AA-SPG), the gradient is estimated using more than one action sample per state, without including the follow-up states of those additional actions in the gradient calculation. The method builds upon conditional Monte-Carlo and yields variance that is smaller or equal to that of single-action SPG given fixed trajectory length [4]. Such additional action samples can be drawn with replacement [6] or without [21]. Furthermore, they can be generated by rewinding the environment [33] or by using a parametrized Q-value approximator [1]. Whereas using Q-value approximator does not incur environment interaction cost, it biases the gradient estimate [29]. In contrast to that, drawing additional action samples from the environment does not bias the gradient, but creates an

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

interaction cost which is unacceptable in a lot of settings. This leads to a question: *given a trajectory of length T and some fixed cost of sampling actions, is SPG variance more favourable when spending the resources on sampling additional actions or extending the trajectory*?

The contributions of this paper are twofold. Firstly, we analyze SPG variance theoretically. We quantify the variance reduction stemming from adopting AA estimation strategy as compared to extending the trajectory of a single-action agent. We prove that AA-SPG is a viable scheme for estimating gradients in RL problems. Secondly, we propose an implementation of AA, which we refer to as dynamics-all-action module (DAA). DAA leverages a learned dynamics model to sample state-action gradients and can be used in conjunction with any on-policy SPG algorithm. We show that augmenting SPG algorithm with DAA yields better sample efficiency and higher reward sums than Q-network all-action SPG, as well as single-action SPG baseline. We validate empirically our approach by augmenting state of the art on-policy SPG algorithm [35] with DAA and testing its performance on a variety of continuous control tasks.

2 Background

A Markov Decision Process (MDP) [30] is a tuple (S, A, R, p, γ), where S is a countable set of states, A is a countable set of actions, $R(s, a)$ is the state-action reward, $p(s'|s, a)$ is a transition kernel (with the initial state distribution denoted as p_0) and $\gamma \in (0, 1]$ is a discount factor. A policy $\pi(a|s)$ is a state-conditioned action distribution. Given a policy π, MDP becomes a Markov reward process with a transition kernel $p^\pi(s'|s) = \int_a \pi(a|s)\, p(s'|s, a)\, da$, which we refer to as the underlying Markov chain. The underlying Markov chain is assumed to have finite variance, an unique stationary distribution denoted as p_0^π [31, 19] and an unique discounted stationary distribution denoted as p_*^π. Interactions with the MDP according to some policy π are called trajectories and are denoted as $\tau_T^\pi(s_t) = ((s_t, a_t, r_t), (s_{t+1}, a_{t+1}, r_{t+1}), ..., (s_{t+T}, a_{t+T}, r_{t+T}))$, where $a_t \sim \pi(a_t|s_t)$, $r_t \sim R(s, a)$ and $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. If $T < \infty$, then the trajectory is referred to as finite. The value function $V^\pi(s) = \mathrm{E}_{\tau_\infty^\pi(s)}[\sum_{t=0}^\infty \gamma^t R(s_t, a_t)]$ and Q-value function $Q^\pi(s, a) = \mathrm{E}_{\tau_\infty^\pi(s|a)}[\sum_{t=0}^\infty \gamma^t R(s_t, a_t)] = R(s, a) + \gamma \mathrm{E}_{s' \sim p(s'|s,a)}[V^\pi(s')]$ sample a_t according to some fixed policy π. State-action advantage is defined as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. An optimal policy is a policy that maximizes discounted total return $J = \int_{s_0} V^\pi(s_0)\, ds_0$. Given a policy parametrized by θ, the values of θ can be updated via SPG [39] $\theta \leftarrow \theta + \nabla_\theta J^*$ which is is given by [38]:

$$
\begin{aligned}
\nabla_\theta J^* &\propto \int_s p_*^\pi(s) \int_a \pi(a|s)\, Q^\pi(s, a)\, \nabla_\theta \log \pi(a|s)\, da\, ds \\
&= \mathop{\mathbb{E}}_{s \sim p_*^\pi} \mathop{\mathbb{E}}_{a \sim \pi} Q^\pi(s, a) \nabla_\theta \log \pi(a|s)
\end{aligned}
\tag{1}
$$

As such, SPG is proportional to a double expectation of $Q^\pi(s, a)\nabla_\theta \log \pi(a|s)$, with the outer expectation taken wrt. the discounted stationary distribution p_*^π and the inner expectation taken wrt. policy π. By assuming ergodicity of the underlying Markov chain, policy gradient can be estimated via a trajectory generated according to the policy [28, 33, 27, 44]:

$$
\begin{aligned}
\nabla J &= \frac{1}{T} \sum_{t=0}^{T-1} \gamma^t\, Q^\pi(s_t, a_t)\, \nabla \log \pi(a_t|s_t) \\
&= \frac{1}{T} \sum_{t=0}^{T-1} \nabla J(s_t, a_t)
\end{aligned}
\tag{2}
$$

Where ∇J denotes the SPG estimator, $\nabla J(s_t, a_t) = \gamma^t Q^\pi(s_t, a_t) \nabla \log \pi(a_t|s_t)$ and $s_t, a_t \sim \tau_T^\pi$. In the setup above, the outer expectation of Equation 1 is estimated via Monte-Carlo [23] with T state samples drawn from the non-discounted stationary distribution p_0^π; and the inner expectation is estimated with a single action per state drawn from the policy $\pi(a|s)$. The resulting variance can be reduced to a certain degree by a control variate, with state value being a popular choice for such baseline [34]. Then, the Q-value from Equation 1 is replaced by $A^\pi(s_t, a_t)$. If state value is learned by a parametrized approximator, it is referred to as the *critic*. Critic bootstrapping [10] is defined

as $Q^\pi(s,a) = R(s,a) + V^\pi(s')$ with $s' \sim p(s'|s,a)$ and can be used to balance the bias-variance tradeoff of Q-value approximations. Given a control variate, the variance of policy gradient can be further reduced by approximating the inner integral of Equation 2 with a quadrature of $N > 1$ action samples. Then ∇J is equal to:

$$\nabla J = \underbrace{\frac{1}{T}\sum_{t=0}^{T-1} \underbrace{\frac{1}{N}\sum_{n=0}^{N-1} \nabla J(s_t, a_t^n)}_{N \text{ action samples per state}}}_{T \text{ state samples per trajectory}} \tag{3}$$

Where a_t^n denotes the nth action sampled at state s_t. Furthermore, MDP transitions are conditioned only on the first action performed (ie. $p^\pi(s_{t+1}|s_t, a_t^n) \neq p^\pi(s_{t+1}|s_t) \iff n = 0$). Such approach is referred to as *all-action policy gradient* or *expected policy gradient* [1, 29, 6]. As follows from law of iterated expectations, all-action (AA) estimator is unbiased and yields lower or equal variance as compared to single-action SPG with equal trajectory length [29]. Since the policy log-probabilities are known, using AA estimation requires approximating the Q-values of additional action samples. As such, AA is often implemented by performing rollouts in a rewinded environment [33, 20, 21] or by leveraging a Q-network at the cost of bias [1, 29, 6]. The variance reduction stemming from using AA-SPG has been shown to increase both performance and sample efficiency of SPG algorithms [6].

3 Variance of Stochastic Policy Gradient

Throughout the section, we assume no stochasticity induced by learning Q-values and we treat Q-values as known. Furthermore, when referring to SPG variance, we refer to the diagonal of policy parameter variance-covariance matrix. As shown by [29, 6], given fixed trajectory length T, AA-SPG variance is smaller or equal to the variance of regular SPG. However, approximating the inner expectation of SPG always uses some resources (ie. compute or environment interactions). Such resources could be used to reduce the SPG variance through other means, for example by extending the trajectory length. To this end, we build upon the result of [29, 6] by comparing the variance reduction effect stemming from employing AA-SPG as opposed to using regular single-action SPG with an extended trajectory length. The variance of SPG, denoted as $\mathbf{V}$, is given by the variance of empirical mean of T dependent random variables sampled from the MDP [17, 5]:

$$\mathbf{V} = \frac{1}{T} \operatorname*{Var}_{s,a\sim p_0^\pi,\pi} [\nabla J(s,a)] + 2\sum_{t=1}^{T-1} \frac{T-t}{T^2} \operatorname*{Cov}_{s_t,a_t\sim\tau_T^\pi} \left[\nabla J(s_0,a_0), \nabla_\theta J(s_t,a_t)\right] \tag{4}$$

As follows from the ergodic theorem [26], conditional probability of visiting state s_t given starting in state s_0 with action a_0^1 approaches the non-discounted stationary distribution p_0^π exponentially fast as t grows $\lim_{t\to\infty} p(s_t|s_0, a_0^1) = p_0^\pi(s_t)$. Therefore, $\mathrm{Cov}_t \geq \mathrm{Cov}_{t+1}$, as well as $\lim_{t\to\infty} \mathrm{Cov}_t = 0$. Equation 4 shows the well known result that increasing the trajectory length T decreases $\mathbf{V}$. This result is key to the success of parallel actor-critic implementations [25, 45]. Unfortunately, the form above does not allow us to inspect the effect of using all-action SPG (ie. sampling many actions per state in the trajectory). To quantify such effect, we decompose $\mathbf{V}$ into sub-components.

Lemma 1 - SPG variance decomposition *Given a finite trajectory τ_T^π and SPG estimation using T state samples and N action samples per state, $\mathbf{V}$ can be decomposed into*:

$$\operatorname*{Var}_{s,a\sim p_0^\pi,\pi} [\nabla J(s,a)] = \underbrace{\operatorname*{Var}_{s\sim p_0^\pi} \left[\operatorname*{E}_{a\sim\pi} \nabla J(s,a)\right]}_{\text{Variance with marginalized policy}} + \underbrace{\frac{1}{N} \operatorname*{E}_{s\sim p_0^\pi} \operatorname*{Var}_{a\sim\pi} [\nabla J(s,a)]}_{\text{State-conditioned variance of policy}} \tag{5}$$

$$\mathrm{Cov}_t = \underbrace{\operatorname*{Cov}_{s_t\sim\tau_T^\pi} \left[\operatorname*{E}_{a\sim\pi} \nabla J(s_0,a_0), \operatorname*{E}_{a\sim\pi} \nabla J(s_t,a_t)\right]}_{\text{Covariance of states with marginalized policy}} + \underbrace{\frac{1}{N} \operatorname*{E}_{s_0\sim p_0^\pi} \operatorname*{Cov}_{s_t,a_t\sim\tau_T^\pi} \left[\nabla J(s_0,a_0), \nabla J(s_t,a_t)\right]}_{\text{Starting state-conditioned policy covariance}} \tag{6}$$

The components can be grouped according to dependency on N:

$$T\,\mathbf{V} = \underbrace{\operatorname*{Var}_{s\sim p_0^\pi}\big[\operatorname*{\mathbb{E}}_{a\sim\pi}\nabla J(s,a)\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\operatorname*{Cov}_{s_t\sim\tau_T^\pi}\big[\operatorname*{\mathbb{E}}_{a\sim\pi}\nabla J(s_0,a_0),\ \operatorname*{\mathbb{E}}_{a\sim\pi}\nabla J(s_t,a_t)\big]}_{\text{Variance of the underlying Markov chain (ie. with marginalized policy stochasticity)}}$$
$$+\underbrace{\frac{1}{N}\operatorname*{\mathbb{E}}_{s_0\sim p_0^\pi}\left(\operatorname*{Var}_{a\sim\pi}\big[\nabla J(s,a)\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\operatorname*{Cov}_{s_t,a_t\sim\tau_T^\pi}\big[\nabla J(s_0,a_0),\nabla J(s_t,a_t)\big]\right)}_{\text{Starting state-conditioned variance of the Markov Decision Process}} \tag{7}$$

For derivation see Appendix A. Given $N = 1$, the variance simplifies to a single-action SPG case. The statement shows that SPG variance can be decomposed into: variance of the underlying Markov chain, which marginalizes policy stochasticity and is decreased only by increasing the trajectory length (T); and starting state-conditioned variance of the policy and covariance correction due to policy stochasticity, which indeed is reduced by both state and action samples (T and N respectively).

SPG variance component	Cartpole	Ball	Reacher	Finger	Cheetah	Walker
$\mathrm{Var}_s\,[\mathbb{E}_a\nabla J]$	0.026	0.011	0.24	0.019	0.001	0.033
$\mathbb{E}_s\,\mathrm{Var}_a\,[\nabla J]$	5.003	0.803	5.877	0.435	1.606	11.948
$\sum\frac{T-t}{T}\,\mathrm{Cov}\,[\mathbb{E}_a\nabla J_0,\mathbb{E}_a\nabla J_t]$	-0.02	0.015	2.029	0.007	0.005	0.047
$\sum\frac{T-t}{T}\,\mathbb{E}_s\mathrm{Cov}\,[\nabla J_0,\nabla J_t]$	0.733	0.015	-2.313	-0.313	0.009	-0.163
Markov chain variance	0.006	0.026	2.269	0.026	0.006	0.081
Policy-dependent variance	5.736	0.819	3.565	0.122	1.615	11.786

Table 1: Mean parameter variance components of SPG estimator with known Q-values. Actor network with single hidden layer and 128 nodes was initialized with orthogonal initialization for the measurement. The components were estimated with Equations 3, 5 and 6, using 110 000 non-baselined interactions, $T = 100$ and learning rate of 0.1 in the 20th policy update.

Table 1 shows sample variance components of SPG estimator. In particular, the table shows that with Q-values marginalized, the policy is responsible for around 90% of SPG variance in tested environments. We proceed with analytical analysis of the variance reduction stemming increasing N and T. For derivation of the Lemma below, see Appendix A.

Lemma 2 - SPG variance reduction wrt. N and T *Given a finite trajectory τ_T^π and SPG estimation using T state samples and N action samples per state, the SPG variance reduction stemming from increasing N by 1 is (denoted as Δ_N) and the SPG variance reduction stemming from increasing the length of trajectory to $T + \delta T$ with $\delta \in (0,\infty)$ (denoted as Δ_T) are equal to*:

$$\Delta_N = \frac{-1}{T(N^2+N)}\operatorname*{\mathbb{E}}_{s\sim p_0^\pi}\left(\operatorname*{Var}_{a\sim\pi}\big[\nabla J(s,a)\big] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\operatorname*{Cov}_{s_t,a_t\sim\tau_T^\pi}\big[\nabla J(s_0,a_0),\nabla J(s_t,a_t)\big]\right) \tag{8}$$

$$\Delta_T = \frac{-\delta}{T+\delta T}\left(\operatorname*{Var}_{s,a\sim p_0^\pi,\pi}\big[\nabla J(s,a)\big] + 2\sum_{t=1}^{T-1}\left(\frac{T-t}{T} - \frac{t}{T+\delta T}\right)\mathrm{Cov}_t\right) \tag{9}$$

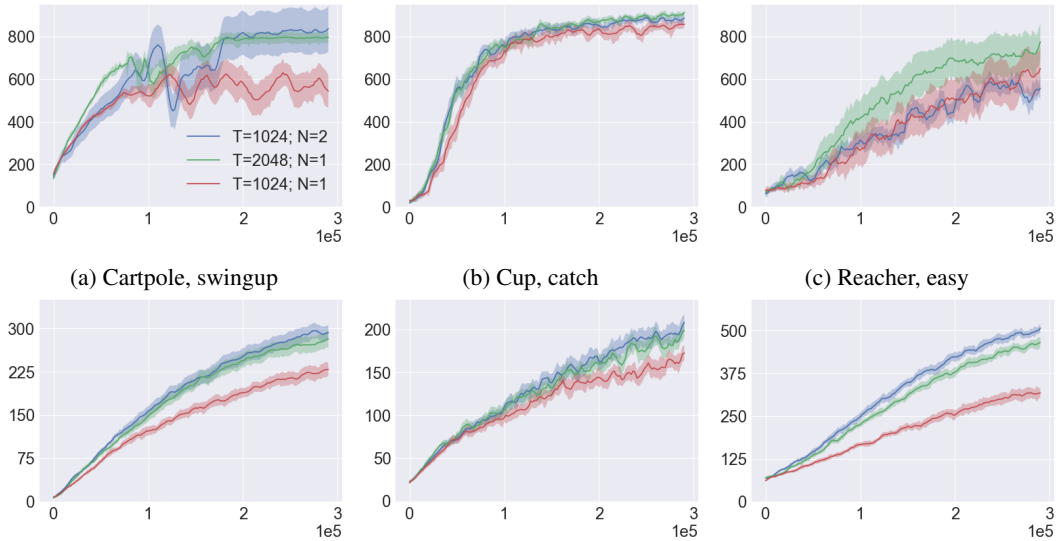
In Equations 9 and 10, $\mathrm{Cov}_t = \mathrm{Cov}_{s_t,a_t\sim\tau_T^\pi}\,[\nabla_\theta J(s_0,a_0),\nabla_\theta J(s_t,a_t)]$. Equations above show the diminishing variance reduction stemming from increasing N by 1 or T by δT. Incorporating δ

captures the notion of relative costs of increasing N and T. If $\delta = 1$, then the cost of increasing N by 1 (sampling one more action per state in trajectory) is equal to doubling the trajectory length. Now, it follows that AA-SPG is an optimal strategy only if $\Delta_N \leq \Delta_T$ for given values of N, T and δ.

Theorem 1 - AA-SPG optimality condition for MDPs *Given a finite trajectory τ_T^π and SPG estimation using T state samples and N action samples per state, the SPG variance reduction stemming from increasing N by 1 is bigger than SPG variance reduction stemming from increasing T by δT for $\delta = 1$ and $N = 1$ when*:

$$\sum_{t=1}^{T-1} \frac{t}{T} \operatorname{Cov}_t \geq \operatorname*{Var}_{s\sim p_0^\pi} \left[\operatorname*{\mathbb{E}}_{a\sim\pi} \nabla J(s,a) \right] + 2 \sum_{t=1}^{T-1} \frac{T-t}{T} \operatorname*{Cov}_{s_t\sim\tau_T^\pi} \left[\operatorname*{\mathbb{E}}_{a\sim\pi} \nabla J(s_0,a_0), \operatorname*{\mathbb{E}}_{a\sim\pi} \nabla J(s_t,a_t) \right] \quad (10)$$

For derivation with $N \geq 1$ and $\delta \in (0, \infty)$ see Appendix A. The theorem represents an optimality condition under which it is optimal to switch from regular SPG (AA-SPG with $N = 1$) to AA-SPG with $N = 2$. Surprisingly, the optimality condition for $\delta = 1$ and $N = 1$ is dependent solely on the covariance structure of the data. As follows from Theorem 1, AA-SPG is optimal when the weighted sum of MDP covariances exceeds the variance of the Markov Chain underlying the MDP. As follows, AA-SPG is most effective in problems where action-dependent covariance constitutes a sizeable portion of the total SPG variance (ie. problems where future outcomes largely depend on actions taken in the past and consequently, $\nabla_\theta J(s_{t+k}, a_{t+k})$ largely depends upon a_t). Figure below shows the potential of AA-SPG estimators on complex continuous action environments.



(a) Cartpole, swingup (b) Cup, catch (c) Reacher, easy

(d) Finger, spin (e) Cheetah, run (f) Walker, walk

Figure 1: Comparison of regular PPO ($T = 1024$; $N = 1$), additional sampled states PPO ($T = 2048$; $N = 1$) and AA-PPO ($T = 1024$; $N = 2$), with additional action sample recorded through environment rollout. The favourable performance of AA-SPG points towards big impact of policy-dependent covariance in selected environments. Results were generated with 20 random seeds.

Corollary 1 - AA-SPG optimality condition for contextual bandits *Given a finite trajectory τ_T^π and SPG estimation using T state samples and N action samples per state, the SPG variance reduction from increasing $\Delta N = 1$ is bigger than SPG variance reduction from $\Delta T = \delta T$ when*:

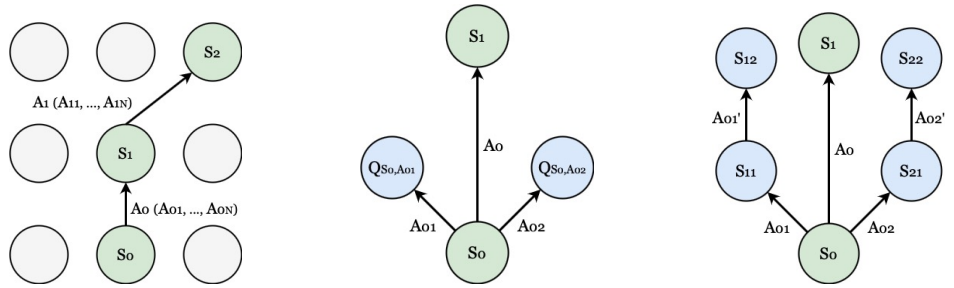
$$\frac{\operatorname*{Var}_{s\sim p_0^\pi} \left[\operatorname*{\mathbb{E}}_{a\sim\pi} \nabla J(s,a) \right]}{\operatorname*{\mathbb{E}}_{s\sim p_0^\pi} \operatorname*{Var}_{a\sim\pi} \left[\nabla J(s,a) \right]} \leq \frac{1-\delta N}{\delta(N^2+N)} \quad (11)$$

Corollary above is a specific case of Theorem 1. By assuming a contextual bandit problem, the covariances are equal to zero and the optimality condition is vastly simplified. As follows from the definition of variance, the LHS of Equation 11 is greater or equal to 0. However, the RHS becomes negative when $\delta N > 1$. Since $N \geq 1$, it follows that AA-SPG is never optimal for bandits if $\delta \geq 1$ (ie. the cost of acquiring an additional action sample is equal or greater than to the cost of acquiring an additional state sample). Whereas the efficiency of AA-SPG for contextual bandits is restricted, Theorem 1 shows that AA-SPG can be a preferable strategy for gradient estimation in MDPs. We leave determining the optimality condition with sampled Q-values for future work.

4 Dynamics-all-action SPG

One of the conclusions of our analysis is that AA-SPG estimators are guaranteed to be useful when $\delta \approx 0$. This is reflected in the existing AA-SPG methods, which utilize a Q-network (QAA-SPG) to approximate additional samples [1, 29, 6]. By using the interactions with environment as the point of reference, QAA-SPG methods assume $\delta = 0$. However, using a Q-network to approximate additional action samples yields bias. Whereas the bias can theoretically be reduced to zero, the conditions required for such bias annihilation are unrealistic [29]. Being fully dependent on policy, Q-network has to learn a constantly changing target [42]. Furthermore, single-action supervision challenges generating informative samples for multiple actions from Q-network. This results in unstable training when Q-network is used to bootstrap the policy gradient [24, 42, 10, 12].

Inspired by the success of model-based RL, we propose dynamics-all-action SPG (DAA-SPG) - an AA module that, contrary to existing all-action methods, does not require a Q-network for SPG approximation. Instead, DAA leverages a learned dynamics model for evaluation of the inner expectation from SPG. In DAA, the Q-value of the additional actions are approximated with a critic-bootstrapped finite trajectory generated by a latent dynamics model, consisting of transition and reward networks [11, 13, 18, 9, 32]. The main advantage of such approach when compared to QAA-SPG is that both reward and transition networks learn stationary targets throughout training, thus offering better convergence properties and lower bias. DAA does not assume an exact form of gradient calculation. As such, it can be used in conjunction with any on-policy SPG algorithm (eg. A2C, TRPO or PPO).



(a) AA-SPG (b) QAA-SPG (c) DAA-SPG

Figure 2: In AA-SPG (Subfigure 2a) policy gradient is estimated using a trajectory of states, with the gradient at each state evaluated using many action samples. The MDP is advanced using only the first sampled action (without including the follow-up states of the additional actions). Since $\log \pi(a|s)$ is known, performing AA-SPG calculation requires Q-values for sampled actions. In QAA-SPG (Subfigure 2b), the Q-values of additional action samples are given by a Q-network. As such, QAA-SPG is prone to all problems stemming from policy learning via Q-network. DAA-SPG (Subfigure 2c), approximates the Q-values using critic-bootstrapped trajectories given by a latent dynamics model. Such procedure yields the benefits of TD(λ) and allows to build the AA-SPG gradient upon approximations of models with static supervision, which have better convergence properties than Q-networks.

The proposed method builds upon two bodies of work: all-action SPG and model-based RL. Previous work on all-action SPG used either a Q-network [1, 29, 6] or rollouts achieved by environment rewinding [33, 21]. In comparison, DAA-SPG leverages a learned dynamics model for sample simulation. In contrast to model-based RL methods, we do not learn the policy inside the dynamics

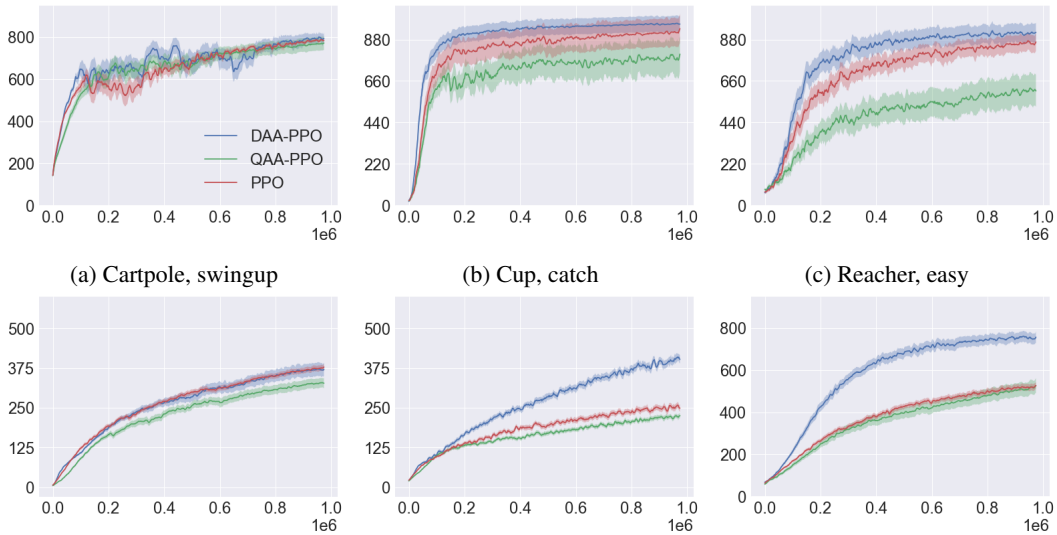
model [11, 18, 32, 14]. Instead, similarly to actor-critic methods, the policy gradient approximation is anchored on the trajectory stemming from the real environment, with the dynamics model used only to refine the estimator (ie. reduce its variance through Monte-Carlo conditioning).

5 Experiments

Our implementations are slight modifications of the PPO implementation provided in [15]. Furthermore, in all experiments and across all algorithms, we use the optimized hyperparameters provided in [15]. There are only two differences between our implementation and the baseline implementation. Firstly, we do not perform advantage normalization. Although it is known to sometimes increase the performance of SPG agents, it has no grounding in SPG theory and impacts the variance structure of the SPG estimate. We point the reader to Appendix B for results that use advantage normalization. Finally, we slightly increase the hidden layer sizes and switch activation functions from Tanh to ReLUs. All architecture details are provided in Appendix C. To simplify our DAA-PPO implementation, we do not use the state of the art transition/reward network learning schemes proposed in Dreamer [13] or SimPLe [18]. Instead, we use a simple action-conditioned two-layer MLP architecture trained using L2 loss on off-policy data sampled from an experience buffer. While using more advanced dynamics models would likely further refine the performance of DAA-SPG, we fill that it would give an unnecessary advantage over the baseline QAA-PPO. In QAA-PPO implementation, the Q-network is a two-layer MLP trained using L2 loss with TD(λ) advantages as targets.

5.1 Performance of DAA-PPO as compared to QAA-PPO and PPO

We investigate the performance of DAA-SPG on continuous action tasks from DeepMind Control Suite [40]. Whereas learning a latent dynamics model from images was shown to work [11, 13, 32], it is known to offer performance benefits over model-free counterparts stemming from backpropagation of additional non-sparse loss functions [16, 36, 47]. To mitigate such performance benefits for DAA-SPG, we use proprioceptive state representations given by the environment, with transition and reward network working directly on such state representations.



(a) Cartpole, swingup (b) Cup, catch (c) Reacher, easy

(d) Finger, spin (e) Cheetah, run (f) Walker, walk

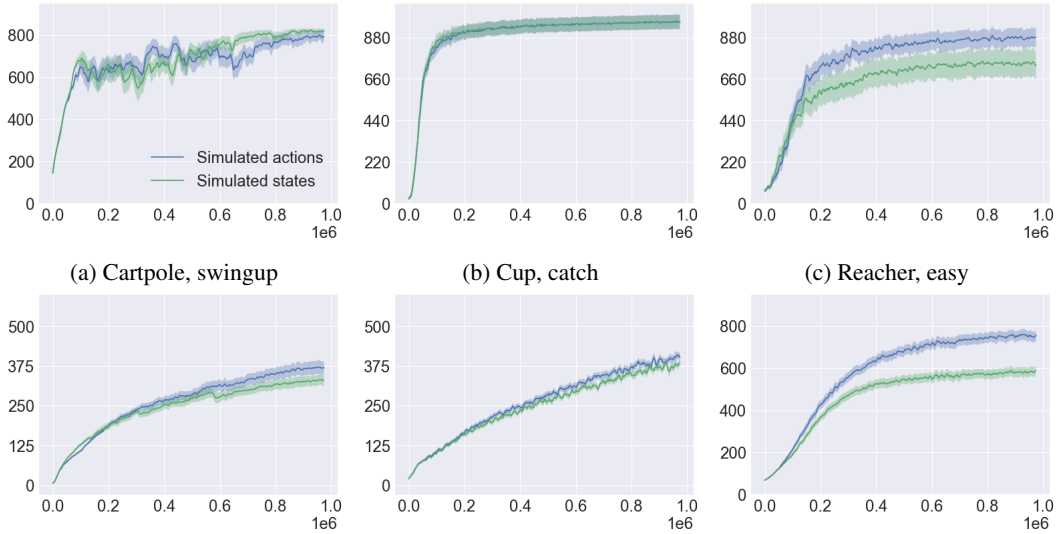
Figure 3: Comparison of PPO ($T = 1024$; $N = 1$) to QAA-PPO ($T = 1024$; $N = 4$) and DAA-PPO ($T = 1024$; $N = 4$). DAA-PPO yields better performance than the baselines on majority of the tasks, pointing towards favourable bias-variance mix offered by DAA-SPG. Results were generated with 25 random seeds.

Figure 3 shows the performance of regular PPO, QAA-PPO and DAA-PPO, with all algorithms using consistent trajectory length of $T = 1024$ and number of action samples $N = 4$ (except PPO for

which $N = 1$). In the experimental setup, PPO gradient is unbiased, but has strictly greater variance that QAA-PPO and DAA-PPO (as follows from Lemma 2). Furthermore, QAA-PPO and DAA-PPO gradients have proportional variance, but differ only with bias (QAA-PPO bias stems from Q-network; DAA-PPO bias stems from dynamics model). Figure 3 points towards more favourable bias-variance mix of DAA-PPO as compared to QAA-PPO. Furthermore, the bias of DAA-PPO can be further reduced by employing more advanced schemes for dynamics model learning. In Appendix B we conduct variety of ablation studies, where we compare the performance of DAA-PPO for various amounts of simulated action samples, as well as different simulation depth for Q-value estimation.

5.2 Simulating additional actions as compared to simulating additional states

When using a dynamics model to simulate actions, the bias of reward model, as well as compounded errors of transitions translate only to the bias of Q-values associated with the simulated action samples. Since the exact log-probabilities and underlying state representations are known, the resulting gradient is still largely approximated using unbiased, exact values. In contrast, using a dynamics model to simulate state samples requires not only the approximation of Q-values, but also prediction of the state representations. The inevitable error of state prediction thus anchors the gradient upon log-probability values which in fact should be associated with a different state. Such perspective is supported by the Lipschitz continuity analysis of approximate MDP models [2, 9]. Note that using dynamics model for simulating additional states in policy gradient approximation is the dominant approach in leveraging dynamics models for policy learning [18, 13, 7]. We therefore hypothesize that using dynamics model for AA estimation might yield more favourable bias-variance tradeoff as compared to using dynamics model to sample additional states given a fixed simulation budget. To this end, we compare performance of DAA-PPO with $N = 4$ to an agent that uses the dynamics model to simulate additional states (as compared to DAA-PPO that uses the dynamics model to simulate additional actions). As such, we compare DAA-PPO with $T = 1024$ and $N = 4$ to PPO with $T = 1024 + T'$, where T' is simulated by the dynamics model and $T' = N * 1024 = 4096$.



(a) Cartpole, swingup (b) Cup, catch (c) Reacher, easy

(d) Finger, spin (e) Cheetah, run (f) Walker, walk

Figure 4: Comparison of DAA-SPG to vanilla dynamics SPG. Green is PPO with $T = 5120$ from which 4096 state samples are generated using dynamics model; and blue is DAA-PPO with $T = 1024$ and $N = 4$. DAA-SPG yields comparable or better results on the tested environments. Whereas vanilla dynamics SPG has potentially lower variance, the performance gains stemming from variance reduction are consumed by compounding bias. Results were generated with 25 random seeds.

6 Related work

All-action SPG The idea of AA-SPG was proposed in an unfinished preprint[1] by Sutton et al. (2001). Later, the topic was expanded upon by several authors. TRPO [33] 'vine procedure' uses multiple without-replacement action samples per state generated via environment 'rewinding'. The without-replacement PG estimator was further refined by [21, 20], who also proposed to use the without-replacement samples as a free baseline. MAC [1] calculates the inner integral of SPG exactly (ie. sample the entire action space for given states) using Q-network, with the scheme applicable only to discrete action spaces and tested on simple environments. Similarly, [29] propose to estimate the inner integral with a quadrature of N samples given by a Q-network. The authors also derive basic theoretical properties of AA-SPG. Besides expanding on the theoretical framework, [6] propose an off-policy algorithm that, given a Gaussian actor and quadratic critic, can compute the inner integral analytically.

Model-based RL ME-TRPO [22] leverages ensemble of environment models to increase the sample efficiency of TRPO. WM [11] uses environment interactions to learn the dynamics model, with the policy learning done via evolutionary strategies inside the dynamics model. Similarly, SimPLe [18] learns the policy by simulating states via the dynamics model. Dreamer [13, 14] refines the dynamics model learning by proposing a sophisticated joint learning scheme for recurrent transition and discrete state representation models, but the policy learning is still done by simulating states inside the dynamics model. Notably, Dreamer was shown to solve notoriously hard Humanoid task [46]. MuZero [32] leverages the dynamics model to perform Monte-Carlo tree search inside the latent model. Perhaps the closest to the proposed approach is MBVE [8]. There, an off-policy DPG agent uses the dynamics model to estimate n-step Q-values and thus refine the approximation.

7 Conclusions

In this paper, we analyzed variance of the SPG estimator mathematically. We showed that it can be disaggregated into sub-components dependent on policy stochasticity, as well as the components which are dependent solely on the structure of the Markov process underlying the policy-embedded MDP. By optimizing such components with respect to the number of state and action samples, we derived an optimality condition which shows when AA-SPG is a preferable strategy as compared to traditional, single-action SPG. We used the result to show the difficult conditions AA-SPG has to meet to be an optimal choice for the case of contextual bandit problems. We hope that those theoretical results will reinvigorate research into AA estimation. Furthermore, we proposed DAA-SPG module - an approach that leverages dynamics models for AA estimation. We evaluated its performance against QAA-SPG and SPG baselines. Our experiments showed that despite the proposed method simplicity, it compares favourably in terms of both sample efficiency and final performance on most of the tested environments and for a wide range of hyperparameter settings. Finally, we discussed the bias-variance trade-off induced by using dynamics model to simulate action samples (DAA-SPG), as compared to using dynamics model to simulate state samples, as is done in variety of the recent model-based RL approaches. We release the code used for experiments under this url.

8 Acknowledgements

We would like to thank Witold Bednorz, Piotr Miłoś and Łukasz Kuciński for valuable discussions and notes. Marek Cygan is cofinanced by National Centre for Research and Development as a part of EU supported Smart Growth Operational Programme 2014-2020 (POIR.01.01.01-00-0392/17-00). The experiments were performed using the Entropy cluster funded by NVIDIA, Intel, the Polish National Science Center grant UMO-2017/26/E/ST6/00622 and ERC Starting Grant TOTAL.

[1] http://incompleteideas.net/papers/SSM-unpublished.pdf

References

[1] Asadi, K., Allen, C., Roderick, M., Mohamed, A.-r., Konidaris, G., Littman, M., and Amazon, B. U. Mean actor critic. *stat*, 1050:1, 2017.

[2] Asadi, K., Misra, D., and Littman, M. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 264–273. PMLR, 2018.

[3] Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[4] Bratley, P., Fox, B. L., and Schrage, L. E. *A guide to simulation*. Springer Science & Business Media, 2011.

[5] Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. *Handbook of markov chain monte carlo*. CRC press, 2011.

[6] Ciosek, K. and Whiteson, S. Expected policy gradients for reinforcement learning. *Journal of Machine Learning Research*, 21(2020), 2020.

[7] Deng, F., Jang, I., and Ahn, S. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 4956–4975. PMLR, 2022.

[8] Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[9] Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179. PMLR, 2019.

[10] Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations (ICLR 2017)*, 2017.

[11] Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[12] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[13] Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

[14] Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2020.

[15] Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., and Wang, W. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

[16] Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[17] Jones, G. L. On the markov chain central limit theorem. *Probability surveys*, 1:299–320, 2004.

[18] Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2019.

[19] Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[20] Kool, W., van Hoof, H., and Welling, M. Buy 4 reinforce samples, get a baseline for free! 2019.

[21] Kool, W., van Hoof, H., and Welling, M. Estimating gradients for discrete random variables by sampling without replacement. In *International Conference on Learning Representations*, 2019.

[22] Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[23] Metropolis, N. and Ulam, S. The monte carlo method. *Journal of the American statistical association*, 44 (247):335–341, 1949.

[24] 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.

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

[26] Norris, J. R. and Norris, J. R. *Markov chains*. Cambridge university press, 1998.

[27] Nota, C. and Thomas, P. S. Is the policy gradient a gradient? In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 939–947, 2020.

[28] Peters, J. and Schaal, S. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2219–2225. IEEE, 2006.

[29] Petit, B., Amdahl-Culleton, L., Liu, Y., Smith, J., and Bacon, P.-L. All-action policy gradient methods: A numerical integration approach. *NeurIPS 2019 Optimization Foundations of Reinforcement Learning Workshop*, 2019.

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

[31] Ross, S. M., Kelly, J. J., Sullivan, R. J., Perry, W. J., Mercer, D., Davis, R. M., Washburn, T. D., Sager, E. V., Boyce, J. B., and Bristow, V. L. *Stochastic processes*, volume 2. Wiley New York, 1996.

[32] 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.

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

[34] 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.

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

[36] Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2020.

[37] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. PMLR, 2014.

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

[39] 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.

[40] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[41] Tucker, G., Bhupatiraju, S., Gu, S., Turner, R., Ghahramani, Z., and Levine, S. The mirage of action-dependent baselines in reinforcement learning. In *International conference on machine learning*, pp. 5015–5024. PMLR, 2018.

[42] Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[43] Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[44] Wu, S., Shi, L., Wang, J., and Tian, G. Understanding policy gradient algorithms: A sensitivity-based approach. In *International Conference on Machine Learning*, pp. 24131–24149. PMLR, 2022.

[45] Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Advances in neural information processing systems*, 30, 2017.

[46] Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2021.

[47] Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10674–10681, 2021.

A Appendix A - derivations

To simplify the notation, we define:

$$\nabla_\theta J(s_t, a_t) = \Upsilon^t_{s,a} \quad \text{and} \quad \mathop{\mathbb{E}}_{a\sim\pi} \nabla_\theta J(s_t, a_t) = \Upsilon^t_s \tag{12}$$

Furthermore, we write:

$$\mathbf{V}_1 = \mathop{\text{Var}}_{s,a\sim p_0^\pi,\pi} [\Upsilon_{s,a}] \quad \text{and} \quad \mathbf{V}_2 = \sum_{t=1}^{T-1} \frac{T-t}{T^2} \mathop{\text{Cov}}_{s_t,a_t\sim\tau_T^\pi} \left[\Upsilon^0_{s,a}, \Upsilon^t_{s,a}\right] \tag{13}$$

With:

$$\mathbf{V} = \frac{1}{T}(\mathbf{V}_1 + 2\,\mathbf{V}_2) \tag{14}$$

For convenience, throughout the Appendix we will assume finite state and action spaces. However, same reasoning works for continuous spaces. For the sake of completeness, we include derivation of expected value of the AA-SPG estimator defined in Equation 3:

$$\begin{aligned}
\mathbb{E}\,\nabla_\theta J &= \sum_s p_0^\pi(s) \prod_{n=1}^{N} \sum_{a^n} \pi(a^n|s) \left(\frac{\Upsilon_{s,a^1}}{N} + ... + \frac{\Upsilon_{s,a^N}}{N}\right) \\
&= \frac{1}{N} \sum_{n=1}^{N} \sum_s p_0^\pi(s) \sum_a \pi(a|s)\Upsilon_{s,a} \prod_{n=1}^{N-1} \sum_{a^n} \pi(a^n|s) \\
&= \sum_s p_0^\pi(s) \sum_a \pi(a|s)\Upsilon_{s,a} = \mathbb{E}\,\nabla_\theta J^*
\end{aligned} \tag{15}$$

A.1 Derivation of Lemma 1

Following the AA-SPG definition outlined in Equation 3, $\mathbf{V}_1$ is equal to:

$$\begin{aligned}
&\mathbf{V}_1 + \left(\mathbb{E}\,\nabla_\theta J\right)^2 = \sum_s p_0^\pi(s) \prod_{n=1}^{N} \sum_{a^n} \pi(a^n|s) \left(\frac{\Upsilon_{s,a^1}}{N} + ... + \frac{\Upsilon_{s,a^N}}{N}\right)^2 \\
&= \frac{N}{N^2} \sum_s p_0^\pi(s) \sum_a \pi(a|s) \left(\Upsilon_{s,a}\right)^2 + \frac{2}{N^2} \binom{N}{2} \sum_s p_0^\pi(s) \left(\sum_a \pi(a|s)\, \Upsilon_{s,a}\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2
\end{aligned} \tag{16}$$

Thus:

$$\begin{aligned}
\mathbf{V}_1 &= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2 - \left(\mathbb{E}\,\nabla_\theta J\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2 - \left(\mathbb{E}\,\nabla_\theta J\right)^2 \\
&= \frac{1}{N} \left(\mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} \left(\Upsilon_{s,a}\right)^2 - \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2\right) + \mathop{\mathbb{E}}_{s\sim p_0^\pi} \left(\Upsilon_s\right)^2 - \left(\mathbb{E}\,\nabla_\theta J\right)^2 \\
&= \mathop{\text{Var}}_{s\sim p_0^\pi} [\Upsilon_s] + \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\text{Var}}_{a\sim\pi} [\Upsilon_{s,a}] \\
&= \mathop{\text{Var}}_{s\sim p_0^\pi} [\mathop{\mathbb{E}}_{a\sim\pi} \nabla_\theta J(s,a)] + \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\text{Var}}_{a\sim\pi} [\nabla_\theta J(s,a)]
\end{aligned} \tag{17}$$

The above result for $N = 1$ is reported in [29], noting it as stemming from law of total variance. However, we could not find the proof in existing literature. We proceed with calculation of $\mathbf{V}_2$ for $N \geq 1$:

$$
\begin{aligned}
&\mathbb{E}\big[\Upsilon^0_{s,a}\Upsilon^t_{s,a}\big] = \sum_{s_0} p^\pi_0(s_0) \prod_{n=1}^{N} \sum_{a^n_0} \pi(a^n_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a^1_0) \prod_{m=1}^{N} \sum_{a^m_t} \pi(a^m_t|s_t) \\
&\left(\frac{\Upsilon^0_{s,a^1}}{N} + ... + \frac{\Upsilon^0_{s,a^N}}{N}\right)\left(\frac{\Upsilon^t_{s,a^1}}{N} + ... + \frac{\Upsilon^t_{s,a^N}}{N}\right) \\
&= \sum_{s_0} p^\pi_0(s_0) \prod_{n=1}^{N} \sum_{a^n_0} \pi(a^n_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a^1_0) \prod_{m=1}^{N} \sum_{a^m_t} \pi(a^m_t|s_t)\left(\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\Upsilon^0_{s,a^i}}{N}\frac{\Upsilon^t_{s,a^j}}{N}\right) \\
&= \frac{1}{N} \sum_{s_0} p^\pi_0(s_0) \sum_{a^1_0} \pi(a^1_0)\, \Upsilon^0_{s,a^1_0} \prod_{n=2}^{N} \sum_{a^n_0} \pi(a^n_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a^1_0) \sum_{a_t} \pi(a_t|s_t)\, \Upsilon^t_{s,a} \\
&+ \frac{N-1}{N} \sum_{s_0} p^\pi_0(s_0) \sum_{a^2_0} \pi(a^2_0)\, \Upsilon^0_{s,a^2_0} \sum_{a^1_0} \pi(a^1_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a^1_0) \sum_{a_t} \pi(a_t|s_t)\, \Upsilon^t_{s,a} \\
&= \frac{1}{N} \sum_{s_0} p^\pi_0(s_0) \sum_{a^1_0} \pi(a^1_0)\, \Upsilon^0_{s,a^1_0} \sum_{s_t} p^\pi_t(s_t|s_0,a^1_0)\, \Upsilon^t_s \\
&+ \frac{N-1}{N} \sum_{s_0} p^\pi_0(s_0)\, \Upsilon^0_s \sum_{a^1_0} \pi(a^1_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a^1_0)\, \Upsilon^t_s
\end{aligned} \tag{18}
$$

Where $p^\pi_t(s_t|s_0,a^1_0)$ denotes the t step transition kernel conditioned on s_0 and a^1_0 (ie. the first sampled action in s_0). Thus, the tth covariance of AA-SPG is equal to:

$$
\begin{aligned}
&\underset{s_t,a_t\sim\tau^\pi_T}{\mathrm{Cov}}\big[\Upsilon^0_{s,a}, \Upsilon^t_{s,a}\big] = \\
&= \frac{1}{N} \sum_{s_0} p^\pi_0(s_0) \sum_{a_0} \pi(a_0)\, \Upsilon^0_{s,a} \sum_{s_t} p^\pi_t(s_t|s_0,a_0)\, \Upsilon^t_s \\
&+ \frac{N-1}{N} \sum_{s_0} p^\pi_0(s_0)\, \Upsilon^0_s \sum_{a_0} \pi(a_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a_0)\, \Upsilon^t_s \\
&- \left(\sum_{s_0} p^\pi_0(s_0) \sum_{a_0} \pi(a_0)\, \Upsilon^0_{s,a}\right)\left(\sum_{s_t} p^\pi_t(s_t) \sum_{a_t} \pi(a_t)\, \Upsilon^t_{s,a}\right) \\
&= \frac{1}{N} \sum_{s_0} p^\pi_0(s_0) \sum_{a_0} \pi(a_0)\, \Upsilon^0_{s,a} \sum_{s_t} p^\pi_t(s_t|s_0,a_0)\, \Upsilon^t_s \\
&+ \frac{N-1}{N} \sum_{s_0} p^\pi_0(s_0)\, \Upsilon^0_s \sum_{a_0} \pi(a_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a_0)\, \Upsilon^t_s - \big(\mathrm{E}\,\Upsilon^0_{s,a}\big)\big(\mathrm{E}\,\Upsilon^t_{s,a}\big) \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s_0\sim p^\pi_0}\left(\sum_{a_0} \pi(a_0)\, \Upsilon^0_{s,a} \sum_{s_t} p^\pi_t(s_t|s_0,a_0)\, \Upsilon^t_s - \Upsilon^0_s \sum_{a_0} \pi(a_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a_0)\, \Upsilon^t_s\right) \\
&+ \left(\sum_{s_0} p^\pi_0(s_0)\, \Upsilon^0_s \sum_{a_0} \pi(a_0|s_0) \sum_{s_t} p^\pi_t(s_t|s_0,a_0)\, \Upsilon^t_s - \big(\mathrm{E}\,\Upsilon^0_{s,a}\big)\big(\mathrm{E}\,\Upsilon^t_{s,a}\big)\right) \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s_0\sim p^\pi_0} \mathrm{Cov}\big[\Upsilon^0_{s,a}, \Upsilon^t_{s,a}\big] + \underset{s_t\sim\tau^\pi_T}{\mathrm{Cov}}\big[\Upsilon^0_s, \Upsilon^t_s\big]
\end{aligned} \tag{19}
$$

Combining Equation 17 with Equation 19 and Equation 13 concludes the derivation of Lemma 1.

A.2 Derivation of Lemma 2

Since N is defined to be a natural number, we calculate the variance reduction effect stemming from increasing N via the forward difference operator:

$$
\Delta_N = \mathbf{V}(N+1) - \mathbf{V}(N) \tag{20}
$$

We also use the shorthand notation:

$$\underset{s_t,a_t\sim\tau_T^\pi}{\mathrm{Cov}}\left[\Upsilon_s^0,\Upsilon_s^t\right]=\alpha_t \tag{21}$$

$$\underset{s\sim p_0^\pi}{\mathbb{E}}\underset{s_t,a_t\sim\tau_T^\pi(s)}{\mathrm{Cov}}\left[\Upsilon_{s,a}^0,\Upsilon_{s,a}^t\right]=\alpha_t^e \tag{22}$$

$$\mathrm{Cov}_t=\alpha_t^e+\alpha_t \tag{23}$$

Thus:

$$\mathbf{V}=\frac{1}{T}\left(\underset{s\sim p_0^\pi}{\mathrm{Var}}\left[\Upsilon_s^0\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t+\frac{1}{N}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}^0\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t^e\right)\right) \tag{24}$$

We proceed with calculation of the forward difference:

$$\begin{aligned}\Delta_N=&\frac{1}{T}\left(\underset{s\sim p_0^\pi}{\mathrm{Var}}\left[\Upsilon_s\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t+\frac{1}{N+1}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t^e\right)\right)\\&-\frac{1}{T}\left(\underset{s\sim p_0^\pi}{\mathrm{Var}}\left[\Upsilon_s\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t+\frac{1}{N}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t^e\right)\right)\\&=\frac{1}{T(N+1)}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t^e\right)\\&-\frac{1}{T\,N}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t^e\right)\\&=\frac{-1}{T(N^2+N)}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\alpha_t^e\right)\end{aligned} \tag{25}$$

Similarly, we calculate $\Delta_T\mathbf{V}$:

$$\begin{aligned}&\Delta_T=\\&=\frac{1}{T+\delta T}\underset{s,a\sim p_0^\pi,\pi}{\mathrm{Var}}[\Upsilon_{s,a}]+2\sum_{t=1}^{T+\delta T-1}\frac{T+\delta T-t}{(T+\delta T)^2}\mathrm{Cov}_t-\frac{1}{T}\underset{s,a\sim p_0^\pi,\pi}{\mathrm{Var}}[\Upsilon_{s,a}]-2\sum_{t=1}^{T-1}\frac{T-t}{T^2}\mathrm{Cov}_t\\&=\frac{-\delta T}{T+\delta T}\underset{s,a\sim p_0^\pi,\pi}{\mathrm{Var}}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\left(\frac{T+\delta T-t}{(T+\delta T)^2}-\frac{T-t}{T^2}\right)\mathrm{Cov}_t+2\sum_{k=T}^{T+\delta T-1}\frac{T-t}{(T+\delta T)^2}\mathrm{Cov}_t\\&=\frac{-\delta}{T+\delta T}\left(\underset{s,a\sim p_0^\pi,\pi}{\mathrm{Var}}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\left(\frac{T-t}{T}-\frac{t}{T+\delta T}\right)\mathrm{Cov}_t-\frac{2}{\delta}\sum_{k=T}^{T+\delta T-1}\frac{T+\delta T-k}{T+\delta T}\mathrm{Cov}_k\right)\end{aligned} \tag{26}$$

Now, we assume that the trajectory length guarantees reaching a regenerative state, and thus $\sum_{k=T}^{T+\delta T-1}\frac{T+\delta T-k}{T+\delta T}\mathrm{Cov}_k=0$:

$$\Delta_T=\frac{-\delta}{T+\delta T}\left(\underset{s,a\sim p_0^\pi,\pi}{\mathrm{Var}}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\left(\frac{T-t}{T}-\frac{t}{T+\delta T}\right)\mathrm{Cov}_t\right) \tag{27}$$

Combining Equation 27 with Equation 17 concludes derivation of Lemma 2.

A.3 Derivation of Theorem 1

We start the derivation by stating that AA-SPG is advantageous in terms of variance reduction as compared to increased trajectory length SPG when $-\Delta_N\geq-\Delta_T$. As such:

$$\frac{1+\delta}{\delta(N^2+N)}\left(\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_t^e\right)\geq\mathop{\text{Var}}_{s,a\sim p_0^\pi,\pi}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\left(\frac{T-t}{T}-\frac{t}{T+\delta T}\right)\text{Cov}_t \tag{28}$$

We use Equations 17 and 19 to expand the RHS:

$$\begin{aligned}&\mathop{\text{Var}}_{s,a\sim p_0^\pi,\pi}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\left(\frac{T-t}{T}-\frac{t}{T+\delta T}\right)\text{Cov}_t=\\&=\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s]+\frac{1}{N}\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\left(\frac{T-t}{T}-\frac{t}{T+\delta T}\right)\left(\alpha_t+\frac{1}{N}\,\alpha_t^e\right)\end{aligned} \tag{29}$$

Now, we use Equations 28 and 29. Furthermore, we move all terms dependent on the policy to the LHS:

$$\begin{aligned}&\frac{1-\delta N}{\delta(N^2+N)}\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}[\Upsilon_{s,a}]+2\sum_{t=1}^{T-1}\left(\frac{(1+\delta-\delta N-\delta^2N)T-(1-2\delta N-\delta^2N)t}{(\delta T+\delta^2T)(N^2+N)}\right)\alpha_t^e\geq\\&\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s]+2\sum_{t=1}^{T-1}\left(\frac{T-t}{T}-\frac{t}{T+\delta T}\right)\alpha_t\end{aligned} \tag{30}$$

Now, in order to recover the Corollary 1, we assume a contextual bandit setup (ie. $p^\pi(s'|s)=p^\pi(s')$). Then:

$$\frac{1-\delta N}{\delta(N^2+N)}\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}[\Upsilon_{s,a}]\geq\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s] \tag{31}$$

Which is equivalent to:

$$\frac{\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s]}{\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}[\Upsilon_{s,a}]}\leq\frac{1-\delta N}{\delta(N^2+N)} \tag{32}$$

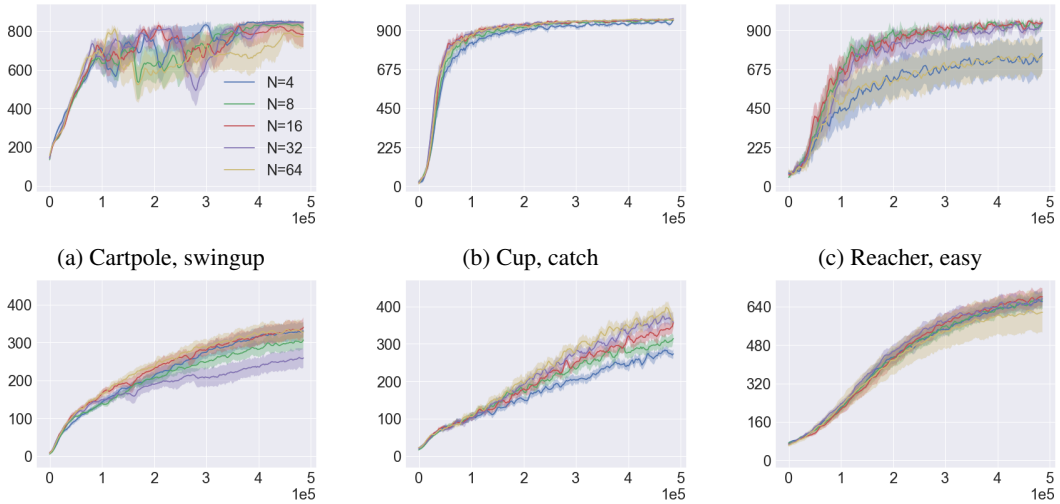
We proceed with the derivation for the MDP setup, where $p^\pi(s'|s)\neq p^\pi(s')$. We write $N=1$, which implies that we start in the regular single-action SPG setup. Furthermore, we assume $\delta=1$, which according to the setup implies equal cost of sampling additional action and state samples. Thus, Equation 30 simplifies to:

$$\begin{aligned}&\sum_{t=1}^{T-1}\frac{t}{T}\,\alpha_t^e\geq\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s]+\sum_{t=1}^{T-1}\frac{2T-3t}{T}\,\alpha_t\\&\equiv\sum_{t=1}^{T-1}\frac{t}{T}\,\alpha_t^e\geq\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_t-\sum_{t=1}^{T-1}\frac{t}{T}\,\alpha_t\\&\equiv\sum_{t=1}^{T-1}\frac{t}{T}\left(\alpha_t^e+\alpha_t\right)\geq\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_t\\&\equiv\sum_{t=1}^{T-1}\frac{t}{T}\,\text{Cov}_t\geq\mathop{\text{Var}}_{s\sim p_0^\pi}[\Upsilon_s]+2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_t\end{aligned} \tag{33}$$

Which concludes the derivation of Theorem 1.

B Appendix B - Ablation studies

We start with analysis of the effect of increase of N. Since simulating action samples with a dynamics model induces bias, we hypothesize that there exists some value of N after which further increase of N will deteriorate the learning. Figure 5 shows DAA-PPO learning curves for different values of N. Interestingly, we see that $N = 4$ chosen for the experiment is an extremely conservative value and further significant performance boost is possible via the increase of N.

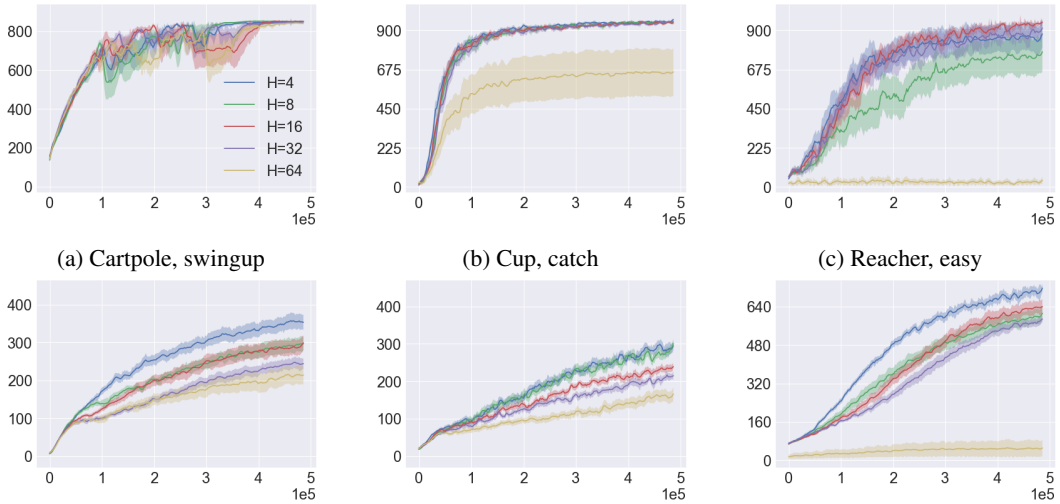


(a) Cartpole, swingup (b) Cup, catch (c) Reacher, easy

(d) Finger, spin (e) Cheetah, run (f) Walker, walk

Figure 5: Comparison of DAA-PPO with different values of N. Blue is $N = 4$; green is $N = 8$; red is $N = 16$; violet is $N = 32$; and yellow is $N = 64$. Note, that in main experiments we use $N = 4$.

As follows from Figure 5, increasing N in DAA-PPO yields performance benefits due to further decrease in variance. Such improvement is not monotonic, as at some point the bias of dynamics model might peturb the learning process (ie. $N = 64$ in Reacher, easy). Furthermore, we analyse the effect of the simulation horizon (depth of simulation for Q-value estimation) on DAA-PPO learning curves. The results are shown in Figure 6



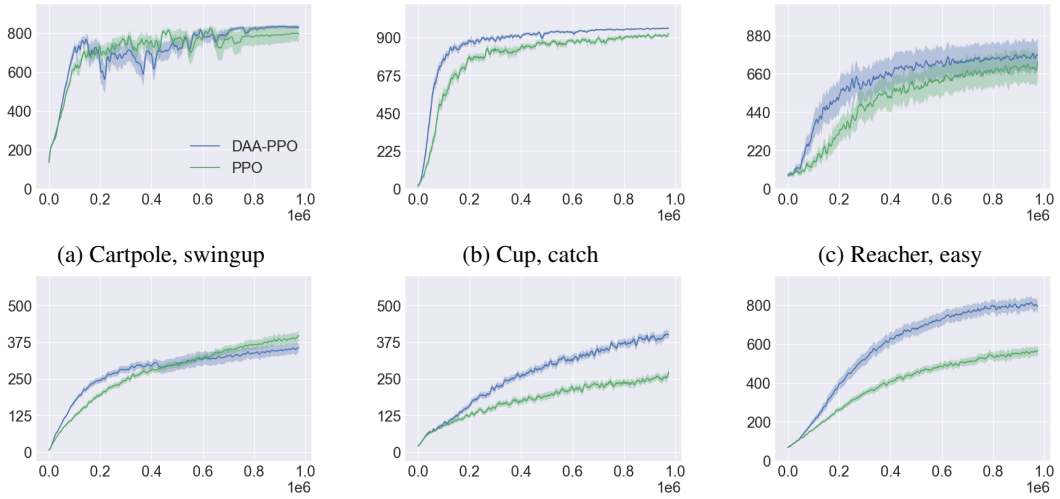
(a) Cartpole, swingup (b) Cup, catch (c) Reacher, easy

(d) Finger, spin (e) Cheetah, run (f) Walker, walk

Figure 6: Comparison of DAA-PPO with different depths of simulation horizon for estimating Q-values. Blue is $N = 4$; green is $N = 8$; red is $N = 16$; violet is $N = 32$; and yellow is $N = 64$.

As can be seen in Figure 6, DAA-PPO performance degenerates for larger simulation horizons for Q-value estimation. For larger simulation horizons, the Q-value estimations are largely dependent on the dynamics model, which might result in extreme levels of bias early in the training. This effect might perhaps be alleviated by using more advanced dynamics model architectures or some sort of annealing scheme, which would increase the simulation horizon as the dynamics model becomes more stable. We leave testing those hypotheses for future work. Results for the above ablations were generated with 20 random seeds.

Finally, we investigate the effect of using advantage normalization in conjunction with DAA. Figure below shows the learning curves for PPO and DAA-PPO when using advantage normalization.



(a) Cartpole, swingup (b) Cup, catch (c) Reacher, easy

(d) Finger, spin (e) Cheetah, run (f) Walker, walk

Figure 7: Comparison of DAA-PPO and PPO when using advantage normalization. Results generated using 15 random seeds.

As follows from Figure 7, using advantage normalization with DAA-PPO further boosts the performance of the proposed algorithm.

C Appendix C - implementation details

Table 2 lists all hyperparameters used in the experiments.

Parameter	PPO	QAA-PPO	DAA-PPO
action repeat	4	4	4
actor optimizer	Adam	Adam	Adam
critic optimizer	Adam	Adam	Adam
dynamics optimizer	NA	NA	Adam
Q-net optimizer	NA	Adam	NA
actor learning rate	0.0003	0.0003	0.0003
critic learning rate	0.0003	0.0003	0.0003
dynamics learning rate	NA	NA	0.0007
Q-net learning rate	NA	0.0003	NA
actor epsilon	1e-5	1e-5	1e-5
critic epsilon	1e-5	1e-5	1e-5
dynamics epsilon	NA	NA	1e-5
Q-net epsilon	NA	1e-5	NA
hidden layer size	512	512	512
λ	0.95	0.95	0.95
discount	0.99	0.99	0.99
batch size	2048	2048	2048
minibatch size	64	64	64
PPO epochs	10	10	10
dynamics buffer size	NA	NA	50 000
dynamics batch size	NA	NA	256
number of simulated actions	NA	4	4
simulation horizon	NA	NA	8
clip coefficient	0.2	0.2	0.2
maximum gradient norm	0.5	0.5	0.5
value coefficient	0.5	0.5	0.5
entropy coefficient	0	0	0

Table 2: Hyperparameter settings used in the main experiment