On Many-Actions Policy Gradient

Anonymous Authors[1]

Abstract

We study variance of stochastic policy gradients (SPGs) with many action samples per state. We derive a many-actions optimality condition, which determines when many-actions SPG yields lower variance as compared to single-action agent with proportionally extended trajectory. We propose Model-Based Many-Actions (MBMA), an approach leveraging dynamics models for many-actions sampling in the context of SPG. MBMA addresses issues associated with existing implementations of many-actions SPG and yields lower bias and comparable variance to SPG estimated from states in model-simulated rollouts. We find that MBMA bias and variance structure matches that predicted by theory. As a result, MBMA achieves improved sample efficiency and higher returns on a range of continuous action environments as compared to model-free, many-actions and model-based on-policy SPG baselines.

1. Introduction

Stochastic policy gradient (SPG) is a method of optimizing stochastic policy through gradient ascent in the context of reinforcement learning (RL) (Williams, 1992; Sutton et al., 1999; Peters & Schaal, 2006). 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 (Schulman et al., 2017). Unfortunately, exact calculation of the gradient is unfeasible and thus the objective has to be estimated (Sutton et al., 1999). Resulting variance is known to impact the learning speed, as well as performance of the trained agent (Konda & Tsitsiklis, 1999; Tucker et al., 2018).

On-policy sample efficiency (ie. the number of environment interactions needed to achieve a certain performance level) is particularly affected by variance, as the gradient must be evaluated over long sequences in order to produce sufficient quality of the SPG estimate (Mnih et al., 2016). As such, a variety of methods for SPG variance reduction have been proposed. The most widely used is baseline variance reduction, which has been shown to improve algorithms stability and became indispensable to contemporary SPG implementations (Peters & Schaal, 2006; Schulman et al., 2015b). Alternative approaches include Q-value bootstrapping (Gu et al., 2017), reducing the effect of long-horizon stochasticity via small discount (Baxter & Bartlett, 2001), increasing number of samples via parallel agents (Mnih et al., 2016) or using many-actions estimator (Asadi et al., 2017; Kool et al., 2019b; Petit et al., 2019; Ciosek & Whiteson, 2020).

In many-actions SPG (MA), the gradient is calculated using more than one action sample per state, without including the follow-up states of additional actions. 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 (Bratley et al., 2011). These additional action samples can be drawn with (Ciosek & Whiteson, 2020) or without replacement (Kool et al., 2019b) and can be generated through rewinding the environment (Schulman et al., 2015a) or using a parametrized Q-value approximator (Asadi et al., 2017). However, drawing additional action samples from the environment is unacceptable in certain settings, while using a Q-network may introduce bias to the gradient estimate. Furthermore, a diminishing variance reduction effect can be achieved through extending the trajectory. This leads to the following questions:

1. Given fixed trajectory length and cost of sampling actions, is SPG variance more favourable when sampling additional actions or extending the trajectory?

2. Given that more samples translate to smaller variance, what is the bias associated with simulating such additional samples via neural networks?

The contributions of this paper are twofold. Firstly, we analyze SPG variance theoretically. We quantify the variance reduction stemming from sampling multiple actions per state as compared to extending the trajectory of a single-action agent. We calculate conditions under which adopting

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

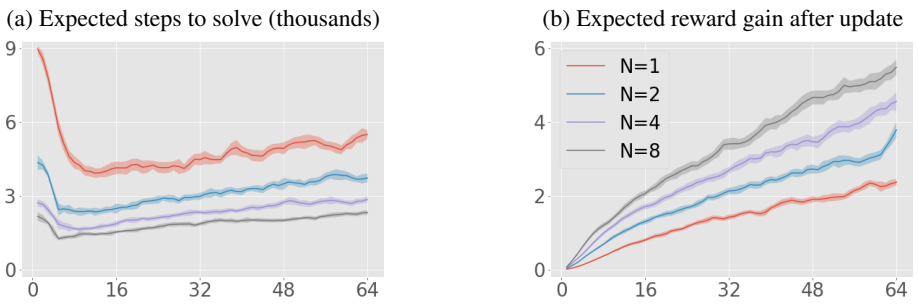

Figure 1. Variance reduction leads to better sample efficiency. We train a CartPole Actor-Critic agent with different batch sizes and many action samples per state (denoted as N). On Figures 1a and 1b X-axis denotes batch size (ie. trajectory length) and Y-axis denote thousands of steps and average performance gain resulting from a single policy update. Increasing batch size leads to better gradient quality at the cost of less updates during training. Sampling more actions yields better gradient quality with less environment steps. However, similarly to extending trajectory, sampling more actions yields diminishing variance reduction. 45 seeds.

MA estimation leads to bigger variance reduction than extending trajectory length. We show that the conditions are often met in RL, but are impossible for contextual bandits. Secondly, we propose an implementation of MA, which we refer to as Model-Based Many-Actions module (MBMA). The module leverages learned dynamics model to sample state-action gradients and can be used in conjunction with any on-policy SPG algorithm. MBMA yields favourable bias/variance structure as compared to learning from states simulated in the dynamics model rollout (Janner et al., 2019; Kaiser et al., 2019; Hafner et al., 2019) in the context of on-policy SPG. We validate our approach and show empirically that using MBMA alongside PPO (Schulman et al., 2017) yields better sample efficiency and higher reward sums on variety of continuous action environments as compared to many-actions, model-based and model-free PPO baselines.

2. Background

A Markov Decision Process (MDP) (Puterman, 2014) 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 (Ross et al., 1996; Konda & Tsitsiklis, 1999), t-step stationary transition kernel p_t^π 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 $\tau_T^\pi(s_t) = ((s_t, a_t, r_t), ..., (s_{t+T}, a_{t+T}, r_{t+T}))$, where $a_t \sim \pi(a_t|s_t)$, $r_t \sim R(s,a)$ and $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. 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 $\theta \leftarrow \theta + \nabla_\theta J$. Since we are interested only in gradient wrt. θ, we drop it from the gradient notation in further uses. The SPG is given by (Sutton & Barto, 2018):

$$\nabla J = \mathop{\mathbb{E}}_{s\sim p_*^\pi} \mathop{\mathbb{E}}_{a\sim\pi} Q^\pi(s,a)\, \nabla \log \pi(a|s) \tag{1}$$

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. The gradient can be estimated via a trajectory sampled according to the policy (Nota & Thomas, 2020; Wu et al., 2022). We denote $\nabla \hat{J}$ as the estimator, $\nabla J(s_t, a_t) = Q^\pi(s_t,a_t) \nabla \log \pi(a_t|s_t)$ with $s_t, a_t \sim p_t^\pi, \pi$. Then, SPG can be calculated:

$$\nabla \hat{J} = \frac{1}{T} \sum_{t=0}^{T-1} \gamma^t\, \nabla J(s_t, a_t) \tag{2}$$

In the setup above, the outer expectation of Equation 1 is estimated via Monte-Carlo (Metropolis & Ulam, 1949) 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 (Schulman et al., 2015b). 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 (Gu et al., 2017) is defined as $Q^\pi(s,a) = R(s,a) + \gamma V^\pi(s')$ 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, $\nabla \hat{J}$ is equal to:

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

Where a_t^n denotes the n^{th} action sampled at state s_t. Furthermore, MDP transitions are conditioned only on the first action performed (ie. $p^\pi(s_{t+1}|s_t, a_t^n) \neq p^\pi(s_{t+1}|s_t) \iff n = 0$). Implementations of such approach were called *all-action policy gradient* or *expected policy gradient* (Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020). As follows from law of iterated expectations, many-actions (MA) estimator is unbiased and yields lower or equal variance as compared to single-action SPG with equal trajectory length (Petit et al., 2019). Since the policy log-probabilities are known, using MA requires approximating the Q-values of additional action samples. As such, MA is often implemented by performing rollouts in a rewinded environment (Schulman et al., 2015a; Kool et al., 2019a;b) or by leveraging a Q-network at the cost of bias (Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020). The variance reduction stemming from using MA has been shown to increase both performance and sample efficiency of SPG algorithms (Schulman et al., 2015a; Kool et al., 2019b).

3. Variance of Stochastic Policy Gradients

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. Finally, to unburden the notation, we define $\Upsilon^t = \gamma^t \nabla J(s_t, a_t)$ and $\bar{\Upsilon}^t = \gamma^t \mathbb{E}_{a\sim\pi} \nabla J(s_t, a_t)$, where we skip the superscript when $t = 0$. Similarly, we use $\mathbb{O}_a(\cdot) = \mathbb{O}_{a\sim\pi}(\cdot)$, $\mathbb{O}_s(\cdot) = \mathbb{O}_{s\sim p_0^\pi}(\cdot)$ and $\mathbb{O}_{s,a}(\cdot) = \mathbb{O}_{s_t, a_t \sim p_t^\pi, \pi}(\cdot)$, where $\mathbb{O}$ denotes expected value, variance and covariance operators. As shown, given fixed trajectory length T, MA-SPG variance is smaller or equal to the variance of single-action agent Petit et al. (2019); Ciosek & Whiteson (2020). However, approximating the inner expectation of SPG always uses resources (ie. compute or environment interactions), which could be used to reduce the SPG variance through other means (eg. extending the trajectory length). To this end, we extend existing results (Petit et al., 2019; Ciosek & Whiteson, 2020) by comparing the variance reduction stemming from employing MA as opposed to using regular single-action SPG with an extended trajectory length.

If the underlying Markov chain is ergodic the variance of SPG, denoted as $\mathbf{V}$, can be calculated via Markov chain Central Limit Theorem (Jones, 2004; Brooks et al., 2011):

$$\mathbf{V} = \frac{1}{T} \operatorname*{Var}_{s,a}\left[\Upsilon\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T^2} \operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right] \tag{4}$$

Note that the states underlying Υ and Υ^t are sampled from the undiscounted stationary distribution p_0^π and the t-step stationary transition kernel p_t^π respectively. As follows from the ergodic theorem (Norris & Norris, 1998), conditional probability of visiting state s_t given starting in state s_0 with action a_0^0 approaches the undiscounted 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 (Mnih et al., 2016; Wu et al., 2017). Unfortunately, the form above assumes single action per state. To quantify variance reduction stemming from many action samples, we decompose $\mathbf{V}$ into sub-components. We include derivations in Appendix A.1.

Lemma 3.1. *Given ergodic MDP, SPG with N action samples per state and T states, $\mathbf{V}$ can be decomposed into:*

$$\begin{aligned}
\operatorname*{Var}_{s,a}\left[\Upsilon\right] &= \operatorname*{Var}_{s}\left[\bar{\Upsilon}\right] + \frac{1}{N}\mathop{\mathbb{E}}_{s} \operatorname*{Var}_{a}\left[\Upsilon\right] \\
\operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right] &= \operatorname*{Cov}_{s,a}\left[\hat{\Upsilon}, \hat{\Upsilon}^t\right] + \frac{1}{N}\mathop{\mathbb{E}}_{s} \operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right]
\end{aligned} \tag{5}$$

Combining Lemma 3.1 with Equation 4 yields expression for decomposed SPG variance, where we group components according to dependence on N:

$$\begin{aligned}
T\,\mathbf{V} = &\underbrace{\operatorname*{Var}_{s}\left[\hat{\Upsilon}\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T} \operatorname*{Cov}_{s,a}\left[\hat{\Upsilon}, \hat{\Upsilon}^t\right]}_{\text{Marginalized policy variance}} \\
&+ \underbrace{\frac{1}{N}\mathop{\mathbb{E}}_{s}\left(\operatorname*{Var}_{a}\left[\Upsilon\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T} \operatorname*{Cov}_{s,a}\left[\Upsilon, \Upsilon^t\right]\right)}_{\text{Policy-dependent variance}}
\end{aligned} \tag{6}$$

Given $N = 1$, the variance simplifies to a single-action case. The statement shows that SPG variance can be decomposed into: marginalized policy variance, which stems from the underlying Markov chain and is decreased only by trajectory length (T); and policy dependent variance, which

indeed is reduced by both sampling more actions per state (N) and increasing trajectory length (T). Table 1 shows estimated variance components and performance of two SPG estimators ($T = 1024; N = 2$ and $T = 2048; N = 1$) for 6 Deepmind Control Suite (DMC) environments. 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.

Lemma 3.2. *Given ergodic MDP, SPG with N action samples per state and T states, variance reduction stemming from increasing N by 1 (denoted as Δ_N) and variance reduction stemming from increasing the trajectory length to $T + \delta T$ with $\delta \in (0, \infty)$ (denoted as Δ_T) are equal to:*

$$\frac{\Delta_N}{\alpha_N} = \mathop{\mathbb{E}}_{s}\left(\mathop{\text{Var}}_{a}\left[\Upsilon\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\mathop{\text{Cov}}_{s,a}\left[\Upsilon, \Upsilon^t\right]\right)$$
$$\frac{\Delta_T}{\alpha_T} = \mathop{\text{Var}}_{s,a}\left[\Upsilon\right] + 2\sum_{t=1}^{T-1}\left(\frac{T-t}{T} - \frac{t}{T+\delta T}\right)\mathop{\text{Cov}}_{s,a}\left[\Upsilon, \Upsilon^t\right]$$
$$\alpha_N = \frac{-1}{T(N^2+N)} \quad \textit{and} \quad \alpha_T = \frac{-\delta}{T+\delta T} \tag{7}$$

Derivation of Lemma 3.2 is detailed in Appendix A.2. Lemma 3.2 shows 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 many-actions yields better variance reduction than increasing trajectory length only if $\Delta_N \leq \Delta_T$ for given values of N, T and δ.

Theorem 3.3. *Given ergodic MDP, SPG with N action samples per T states, variance reduction stemming from increasing N by 1 is bigger than variance reduction stemming from increasing T by δT for $\delta = 1$ and $N = 1$ when:*

$$\sum_{t=1}^{T-1}\frac{t}{T}\mathop{\text{Cov}}_{s,a}\left[\Upsilon, \Upsilon^t\right] \geq \mathop{\text{Var}}_{s}\left[\hat{\Upsilon}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\mathop{\text{Cov}}_{s,a}\left[\hat{\Upsilon}, \hat{\Upsilon}^t\right] \tag{8}$$

For derivation with $N \geq 1$ and $\delta \in (0, \infty)$ see Equation 14 in Appendix A.3. The theorem represents a condition under which optimal to switch from regular SPG (MA-SPG with $N = 1$) to MA-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 3.3, MA is optimal when the weighted sum of MDP covariances exceeds the variance of the Markov Chain underlying the MDP. As follows, MA 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).

Corollary 3.4. *Given ergodic MDP, SPG with N action samples per state and T states, the SPG variance reduction from increasing $\Delta N = 1$ is bigger than SPG variance reduction from $\Delta T = \delta T$ when:*

$$\frac{\mathop{\text{Var}}_{s}\left[\hat{\Upsilon}\right]}{\mathop{\mathbb{E}}_{s}\mathop{\text{Var}}_{a}\left[\Upsilon\right]} \leq \frac{1-\delta N}{\delta(N^2+N)} \tag{9}$$

Corollary above is a specific case of Theorem 3.3. 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 9 is greater or equal to 0. However, the RHS becomes negative when $\delta N > 1$. Since $N \geq 1$, it follows that MA 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 MA for contextual bandits is restricted, Theorem 3.3 shows that MA can be a preferable strategy for gradient estimation in MDPs. We leave researching the optimality condition for setting with sampled Q-values or deterministic policy gradients for future work.

4. Model-Based Many-Actions SPG

A somewhat obvious conclusion of our analysis is that MA-SPG is guaranteed to be useful when $\delta \approx 0$. This is reflected in existing MA-SPG implementations, which utilize a Q-network (QMA) to approximate additional samples (Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020). By using environment interactions as the point of reference, QMA assumes $\delta = 0$. However, using a Q-network to approximate additional action samples yields bias and incurs non-trivial computational cost. Whereas the bias can theoretically be reduced to zero, the conditions required for such bias annihilation are unrealistic (Petit et al., 2019). Being fully dependent on policy, Q-network learns a non-stationary target (Van Hasselt et al., 2016). 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 (Mnih et al., 2015; Van Hasselt et al., 2016; Gu et al., 2017; Haarnoja et al., 2018).

Inspired by the success of model-based RL, we propose Model-Based Many-Actions (MBMA) - an MA module that, contrary to existing many-actions methods, does not require a Q-network for SPG approximation. Instead, MBMA

Table 1. Decomposed trace of variance-covariance matrix divided by the number of parameters. The components were estimated by marginalizing Q-values, with Equation 3 and Lemma 3.1 using 125000 non-baselined interactions. Last two columns record best performance during $500k$ environment steps (average performance shown in the brackets). Performance of SPG variants was measured during $500k$ training steps with additional action sample drawn from the environment. With majority of variance depending on the policy, MA often yields better performance as compared to single-action agent with extended trajectory. We detail the setting in Appendix C.

	VARIANCE COMPONENT		PERFORMANCE	
TASK	MARGINALIZED POLICY	POLICY DEPENDENT	$(T,\ N) = (1024,\ 2)$	$(T,\ N) = (2048,\ 1)$
BALL CATCH	0.026 (3%)	0.819 (97%)	905 (708)	920 (715)
CART SWINGUP	0.006 (1%)	5.736 (99%)	837 (670)	801 (669)
CHEETAH RUN	0.006 (1%)	1.615 (99%)	208 (131)	204 (126)
FINGER SPIN	0.026 (18%)	0.122 (82%)	304 (187)	281 (179)
REACHER EASY	2.269 (39%)	3.565 (61%)	428 (262)	776 (488)
WALKER WALK	0.081 (1%)	11.786 (99%)	509 (315)	465 (287)

leverages a learned dynamics model for better evaluation of SPG inner expectation. In MBMA, Q-values of the additional actions are approximated with a critic-bootstrapped trajectory generated within a dynamics model, consisting of transition and reward networks (Ha & Schmidhuber, 2018; Hafner et al., 2019; Kaiser et al., 2019; Gelada et al., 2019; Schrittwieser et al., 2020). The advantage of such approach when compared to QMA is that both reward and transition networks learn stationary targets throughout training, thus offering better convergence properties and lower bias. MDMA does not assume any form of gradient calculation. As such, it can be used in conjunction with any on-policy SPG algorithm (eg. A2C or PPO). The proposed method builds upon two bodies of work: many-actions SPG and model-based SPG. Previous work on many-actions SPG used either a Q-network (Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020) or rollouts achieved by environment rewinding (Schulman et al., 2015a; Kool et al., 2019b). In comparison, MBMA leverages a learned dynamics model for sample simulation. In contrast to MBRL methods, we do not anchor the policy gradient on states simulated via the dynamics model (Janner et al., 2019; Kaiser et al., 2019; Hafner et al., 2020). Instead, similarly to actor-critic methods, the gradient approximation is anchored on the trajectory sampled from the real environment, with the dynamics model used only to refine the estimator (ie. reduce variance through Monte-Carlo conditioning).

Using a dynamics model to simulate actions biases the policy gradient through Q-values, due to the bias in the reward model and compounded errors in transitions. Since the log-probabilities and underlying states 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 (Asadi et al., 2018; Gelada et al., 2019). Note that using dynamics model for simulating additional states in policy gradient approximation is the dominant approach in leveraging dynamics models for policy learning (Buckman et al., 2018; Janner et al., 2019; Clavera et al., 2019; Kaiser et al., 2019; Hafner et al., 2019; 2020; Deng et al., 2022). We therefore hypothesize that using dynamics model for MA estimation might yield more favourable bias-variance tradeoff as compared to using dynamics model to sample additional states given a fixed simulation budget.

5. Experiments

We investigate performance of MA-SPG on the widest range of continuous action locomotion tasks in many-actions literature (Schulman et al., 2015a; Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020) - we test MBMA on 14 DMC tasks (Tassa et al., 2018) of varying difficulty. We compare MBMA to benchmarks: QMA; model-free SPG; and a variant of model-based policy optimization (MBPO) (Janner et al., 2019). MBPO leverages a dynamics model to simulate trajectories branching from the on-policy data. Note, that we consider on-policy SPG setting. As such, we pair MBPO with a on-policy PPO agent, as opposed to off-policy SAC agent considered in the original implementation. We ask the reader to note that off-policy SPG methods tend to perform better in terms of environment steps [NEW CITE]. Below we describe the algorithms used in our experiments and discuss their bias-variance structure.

PPO Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a model-free on-policy SPG algorithm that leverages multiple actor-critic updates on single batch of on-policy data. PPO uses a trust-region type of objective that prevents the policy to diverge too much between up-

dates. We use PPO as the single-action agent that performs unbiased policy updates. The algorithm does not reduce the SPG variance beyond using baseline. As such, we expect PPO variance to be the highest across the tested algorithms.

MBMA PPO that leverages dynamics model to sample additional actions. The algorithm uses simple MLP transition and reward networks that are trained using MSE loss before performing actor updates. Similarly to QMA, the algorithm performs biased policy updates, with the bias stemming only from the dynamics model Q-value approximation error. Since the dynamics model rollouts depend on the sampled actions, the Q-value approximate has non-zero variance.

QMA PPO that uses an auxiliary Q-network to sample additional actions for every visited state(Asadi et al., 2017; Petit et al., 2019; Ciosek & Whiteson, 2020). To stabilize the training, we implement QMA-PPO using two Q-networks and choose the smaller prediction for given state-action pair (Van Hasselt et al., 2016; Haarnoja et al., 2018). Q-networks are trained using MSE loss using TD(λ) as targets, which we found to be performing better on average than expected SARSA proposed in the literature (Petit et al., 2019; Ciosek & Whiteson, 2020). The updates performed by QMA are biased, as they are depend on the output of biased Q-network. Q-network determinism reduces the absolute variance beyond the reduction stemming from many-actions.

MBPO PPO that leverages dynamics model to perform finite horizon rollouts branching from the on-policy data (Janner et al., 2019). MBPO allows to estimate SPG using a mix of real and simulated states (ie. extend the trajectory length). As such, the algorithm leverages the most common paradigm in model-based SPG - using dynamics model to generate trajectories (Hafner et al., 2019; Kaiser et al., 2019). Similarly to MBMA, transition and reward networks are trained using MSE loss. Using dynamics model generated trajectories for SPG updates biases the gradient in two ways. Firstly, similarly to QMA and MBMA, there is bias stemming from Q-value approximation. Secondly, contrary to MA methods, log-probabilities become anchored on biased transitions stemming from the model. Due to extended trajectory, the gradient updates have reduced variance.

We base our implementations on the PPO codebase provided by CleanRL (Huang et al., 2022b) and hyperparameters optimized for PPO Huang et al. (2022a). To accommodate more complex tasks, we increase the number of parameters in actor and critic networks across all tasks. Furthermore, we do not use advantage normalization: it has no grounding in SPG theory and can impact the variance structure of the problem at hand; but it can also adversely impact learning in certain environments (Andrychowicz et al., 2021). All algorithms use the same number of parameters in the actor and critic networks, which are updated the same number of times. QMA, MBPO and MBMA use equal number of additional samples (which are tuned for best performance of baselines, see Appendix E): for QMA and MBMA we use additional 8 actions per state; for MBPO we sample rollout of 8 states per state (which results in extending the trajectory 9-fold) and $TD(\lambda)$ for value estimation. We anneal the number of additional samples until $150k$ step of the training for all methods. Whereas learning dynamics models from images is known to work (Hafner et al., 2019; Schrittwieser et al., 2020), it is known to offer performance benefits over model-free counterparts stemming from backpropagation of additional non-sparse loss functions (Jaderberg et al., 2016; Schwarzer et al., 2020; Yarats et al., 2021b). To mitigate such benefits for algorithms using dynamics models, we use proprioceptive representations given by the environment, with transition and reward networks working on such representations. Similarly, neither MBPO nor MBMA use ensemble of dynamics models (Buckman et al., 2018; Kurutach et al., 2018; Janner et al., 2019). Note, that using the same number of simulated samples for all methods yields different computational cost for each method. Calculating Q-value with a dynamics model requires unrolling model for multiple steps (forward pushes) before bootstrapping it with critic. In contrast, QMA calculates them in a single step. Relative compute time measurements are provided in Appendix C.6. Hyperparameters and architectures are detailed in Appendix C.5. We also provide implementation details and pseudocode in Appendix C.6.

Table 2. SPG per-parameter bias associated with action (MA) and state (MS) sample simulation. Q and $\hat{Q}$ denote the true Q-value and approximate Q-value of given state-action pair respectively; s^* denotes the output of the transition model; and $\mathcal{K}$ denotes the Lipschitz norm of $f_s = \nabla \log \pi(a|s)$. For MS the bias is an upper-bound. We include extended calculations in Appendix B.

	$\nabla J(s,a) - \nabla \hat{J}(s,a)$
MA $=$	$f_s(Q - \hat{Q})$
MS $\leq$	$f_s(Q - \hat{Q}) + \sqrt{(\mathcal{K}(s - \hat{s}))^2 + f_s^2(Q^2 - Q)}$

5.1. Agent performance

We measure sample efficiency and quality of trained agents by recording performance within two experience regimes - $500k$ and $1M$ environment steps. To this end, we measure the average best performance within designed data limit. The $1M$ performance reflects the final quality of the agent within our training regime, while the $500k$ performance measures sample efficiency. To account for stochasticity of 15 seeds we aggregate results using IQM (Agarwal et al., 2021). We provide further details about the experimental setting

Table 3. Best and average policy performance for 14 DMC tasks measured via interquartile range with sample standard deviations (for full-sample statistics and learning curves see Appendices D and F respectively). Bold text indicates best in class results. 15 seeds.

TASK	PPO	MBMA (OURS)	MBPO	QMA
ACRO SWINGUP	$56 \pm 6\,(32 \pm 7)$	$\mathbf{65 \pm 8\,(42 \pm 6)}$	$62 \pm 12\,(40 \pm 7)$	$34 \pm 10\,(17 \pm 7)$
BALL CATCH	$948 \pm 8\,(831 \pm 20)$	$\mathbf{974 \pm 2\,(898 \pm 10)}$	$971 \pm 3\,(888 \pm 10)$	$934 \pm 5\,(770 \pm 30)$
CART SWINGUP	$788 \pm 9\,(656 \pm 32)$	$\mathbf{853 \pm 3\,(738 \pm 41)}$	$834 \pm 18\,(702 \pm 47)$	$801 \pm 3\,(686 \pm 17)$
CART 2-POLES	$303 \pm 9\,(253 \pm 8)$	$\mathbf{623 \pm 55\,(402 \pm 28)}$	$405 \pm 41\,(300 \pm 18)$	$330 \pm 10\,(254 \pm 7)$
CART 3-POLES	$227 \pm 15\,(198 \pm 11)$	$223 \pm 15\,(197 \pm 10)$	$\mathbf{271 \pm 12\,(225 \pm 9)}$	$262 \pm 5\,(213 \pm 5)$
CHEETAH RUN	$278 \pm 13\,(186 \pm 12)$	$\mathbf{506 \pm 18\,(313 \pm 16)}$	$484 \pm 15\,(289 \pm 14)$	$213 \pm 10\,(136 \pm 8)$
FINGER SPIN	$\mathbf{406 \pm 14\,(283 \pm 9)}$	$353 \pm 7\,(262 \pm 10)$	$321 \pm 9\,(252 \pm 13)$	$355 \pm 17\,(245 \pm 13)$
FINGER TURN	$405 \pm 68\,(196 \pm 64)$	$\mathbf{416 \pm 77\,(206 \pm 64)}$	$404 \pm 106\,(176 \pm 63)$	$357 \pm 131\,(175 \pm 62)$
POINT EASY	$907 \pm 8\,(851 \pm 8)$	$\mathbf{914 \pm 7\,(868 \pm 7)}$	$911 \pm 9\,(868 \pm 8)$	$690 \pm 82\,(84 \pm 15)$
REACHER EASY	$943 \pm 9\,(713 \pm 52)$	$973 \pm 2\,(\mathbf{842 \pm 31})$	$\mathbf{975 \pm 4}\,(821 \pm 26)$	$454 \pm 119\,(286 \pm 86)$
REACHER HARD	$684 \pm 115\,(404 \pm 83)$	$952 \pm 4\,(734 \pm 40)$	$\mathbf{963 \pm 3\,(783 \pm 33)}$	$515 \pm 17\,(395 \pm 20)$
WALKER STAND	$934 \pm 8\,(755 \pm 19)$	$\mathbf{960 \pm 4\,(858 \pm 12)}$	$958 \pm 3\,(841 \pm 14)$	$898 \pm 13\,(663 \pm 20)$
WALKER WALK	$510 \pm 11\,(375 \pm 15)$	$\mathbf{900 \pm 11\,(696 \pm 16)}$	$743 \pm 24\,(583 \pm 19)$	$514 \pm 10\,(346 \pm 14)$
WALKER RUN	$201 \pm 6\,(150 \pm 4)$	$\mathbf{336 \pm 10\,(251 \pm 7)}$	$239 \pm 11\,(192 \pm 7)$	$211 \pm 7\,(141 \pm 5)$
NORMALIZED	1 (1)	1.28 (1.32)	1.17 (1.22)	0.88 (0.81)

in Appendix C.3, as well as performance rankings, score distributions and aggregate performance scores (Agarwal et al., 2021) in [NEW APPENDIX]. We find that MBMA performs better wrt. both metrics in 10 out of 14 DMC tasks, while MBPO and PPO have better performance in 3 and 1 environment respectively. However, the performance differences are within the margin of statistical error for some cases. Note that we use hyperparameters tuned wrt. PPO and MBPO. We observe greater performance gaps benefiting MBMA for different hyperparameter settings (see Appendix E, where we compare the performance for different number of simulated samples and various simulation horizons). The results for the tested algorithms are shown in Table 3, with full-sample statistics and learning curves provided in Appendices D and F.

5.2. Bias-variance tradeoff

We evaluate the sample bias and variance of updates for four different methods. To do so, we train a PPO agent and sample policy gradients at 10 different points in training. For each point, we gather 125 gradient estimates per method using a batch size of 2500 states for each gradient estimation. The experimental gradients are calculated using the regular actor-critic update to avoid introducing bias of the clipping procedure (ie. Equation 2). We assume that the actor-critic method is unbiased and use it as a reference to calculate the bias of the other methods. Specifically, we estimate the point bias by taking the difference between the average gradients of the actor-critic method and the gradients of the other methods, with the averages taken over 125 gradient samples per point. Finally, we normalize the bias values by dividing them element-wise by the absolute value of respective gradient vector. The variance is estimated by the diagonal of parameter sample covariance matrix calculated over 125 gradients samples per training point. To get comparable values we divide sample variance element-wise by the squared value of gradient mean (ie. relative variance). Both bias and variance are calculated per parameter by summing over the parameter space and dividing the results by the total number of parameters. We provide further details about this experimental setting in Appendix C.4. We find that MBMA produces consistently less bias than than other methods, while offering greater or comparable variance reduction. This is in line with observed performance, as well as MBMA robustness to amount of simulated samples (see Appendix E). On average, MBMA measures lowest bias and lowest variance. Furthermore, we find that QMA produces smaller gradients than other methods given the same data. This points towards low variation of the Q-network output and subsequent gradient cancellation. We find that QMA has the highest relative bias despite MA approach. We find this unsurprising, since as noted in earlier sections, Q-networks pursue a more difficult target than dynamics models. Furthermore, even though QMA has the lowest absolute variance (due to no stochasticity in Q-value estimation), its smallest expected gradient gradient size leads to greater impact its variance and thus has highest relative variance amongst methods. The results for the four tested methods are shown in Table 4 with disaggregated results provided in Appendix G.

Table 4. Relative bias and variance of the tested methods measured via interquartile range with sample standard deviations (for full sample statistics and training curves see Appendix F). We bold the best in class result. 15 seeds.

	Relative Bias			Relative Variance			
Task	MBMA	MBPO	QMA	AC	MBMA	MBPO	QMA
acro swingup	$\mathbf{8.09 \pm 0.3}$	8.71 ± 0.4	10.6 ± 0.6	44.6 ± 1.5	31.5 ± 1.1	$\mathbf{30.6 \pm 1.1}$	31.3 ± 1.3
ball catch	$\mathbf{5.10 \pm 0.3}$	5.94 ± 0.3	12.4 ± 1.0	48.1 ± 1.5	19.4 ± 1.0	$\mathbf{17.6 \pm 0.8}$	19.2 ± 1.4
cart swingup	$\mathbf{3.23 \pm 0.3}$	3.48 ± 0.3	9.04 ± 1.2	22.4 ± 1.6	8.24 ± 0.6	$\mathbf{8.09 \pm 0.7}$	11.3 ± 1.4
cart 2-pole	$\mathbf{6.38 \pm 0.4}$	6.80 ± 0.5	10.1 ± 0.6	40.9 ± 1.8	21.9 ± 1.3	$\mathbf{20.6 \pm 1.4}$	30.6 ± 1.3
cart 3-pole	7.88 ± 0.6	$\mathbf{7.67 \pm 0.7}$	11.0 ± 0.9	45.5 ± 2.5	$\mathbf{30.1 \pm 2.8}$	32.4 ± 2.7	34.6 ± 2.1
cheetah run	$\mathbf{11.4 \pm 0.5}$	12.0 ± 0.5	21.3 ± 0.9	77.4 ± 2.9	38.8 ± 1.4	$\mathbf{37.9 \pm 1.4}$	51.4 ± 1.8
finger spin	$\mathbf{4.49 \pm 0.4}$	4.88 ± 0.5	9.58 ± 0.5	38.9 ± 1.3	$\mathbf{24.9 \pm 1.7}$	35.1 ± 2.4	28.4 ± 1.1
finger turn	$\mathbf{3.45 \pm 0.5}$	4.05 ± 0.5	10.1 ± 0.9	47.5 ± 3.4	$\mathbf{24.5 \pm 2.1}$	29.4 ± 3.1	27.3 ± 2.4
point easy	$\mathbf{1.36 \pm 0.1}$	1.50 ± 0.1	3.91 ± 0.5	15.8 ± 1.3	4.57 ± 0.4	$\mathbf{3.95 \pm 0.4}$	4.55 ± 0.5
reacher easy	$\mathbf{3.94 \pm 0.2}$	4.64 ± 0.3	10.2 ± 1.0	35.8 ± 2.2	13.5 ± 0.6	$\mathbf{13.0 \pm 0.6}$	14.1 ± 1.3
reacher hard	$\mathbf{5.29 \pm 0.3}$	6.20 ± 0.38	11.7 ± 0.8	40.4 ± 1.7	19.7 ± 1.1	$\mathbf{18.7 \pm 1.0}$	21.8 ± 1.6
walker stand	$\mathbf{9.70 \pm 0.4}$	11.5 ± 0.5	19.2 ± 0.9	69.7 ± 1.9	$\mathbf{39.9 \pm 1.2}$	43.1 ± 1.3	50.8 ± 2.0
walker walk	$\mathbf{11.5 \pm 0.4}$	12.7 ± 0.5	16.2 ± 0.9	78.4 ± 1.9	$\mathbf{41.2 \pm 1.3}$	42.9 ± 1.4	60.9 ± 1.2
walker run	$\mathbf{11.3 \pm 0.4}$	12.2 ± 0.5	15.8 ± 1.0	78.3 ± 2.0	41.6 ± 1.1	$\mathbf{40.6 \pm 1.2}$	60.8 ± 1.4
Average	6.65	7.31	12.22	48.84	25.70	26.71	31.93

6. Related work

Many-Actions SPG The idea of sampling many actions per state was proposed in an unfinished preprint[1] by Sutton et al. (2001). Later, the topic was expanded upon by several authors. TRPO (Schulman et al., 2015a) 'vine procedure' uses multiple without-replacement action samples per state generated via environment rewinding. The without-replacement PG estimator was further refined by using the without-replacement samples as a free baseline (Kool et al., 2019b;a). MAC (Asadi et al., 2017) 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, Petit et al. (2019) 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 MA SPG. Besides expanding on the theoretical framework, Ciosek & Whiteson (2020) propose an off-policy algorithm that, given a Gaussian actor and quadratic critic, can compute the inner integral analytically.

Model-Based RL ME-TRPO (Kurutach et al., 2018) leverages ensemble of environment models to increase the sample efficiency of TRPO. WM (Ha & Schmidhuber, 2018) uses environment interactions to learn the dynamics model, with the policy learning done via evolutionary strategies inside the dynamics model. Similarly, SimPLe (Kaiser et al., 2019) learns the policy by simulating states via the dynamics model. Dreamer (Hafner et al., 2019; 2020) refines the latent 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 starting from sampled off-policy transitions. Notably, Dreamer was shown to solve notoriously hard Humanoid task (Yarats et al., 2021a). Differentiable dynamics models allow for direct gradient optimization of the policy as an alternative to traditional SPG. Methods like MAAC [NEWCITE] and DDPPO [NEWCITE] explore policy optimization via backpropagating through the dynamics model. MuZero (Schrittwieser et al., 2020) leverages the dynamics model to perform Monte-Carlo tree search inside the latent model. Perhaps the closest to the proposed approach is MBVE (Feinberg et al., 2018). There, an off-policy DPG agent uses the dynamics model to estimate n-step Q-values and thus refine the approximation. However, our analysis is restricted to model-based on-policy SPG and we leave analysis of MBMA in the context of off-policy agents and backpropagating through dynamics model for future work.

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

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

number of state and action samples, we derived an optimality condition which shows when MA is a preferable strategy as compared to traditional, single-action SPG. We used the result to show the difficult conditions MA 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 MA estimation in the context of RL. Main limitation of our theoretical analysis is its dependence on Markov chain Central Limit Theorem, as such its results hold only if the underlying Markov chain is ergodic. Furthermore, it is conducted in the context of on-policy SPG and its conclusions are applicable only to such settings.

Furthermore, we discussed the bias-variance trade-off induced by using Q-network and dynamics model to simulate action or state samples. We showed that the bias associated with simulating additional states is of more complex form than the bias associated with simulating actions, while offering similar variance reduction benefits. We measured relative bias and variance of policy gradients calculated via each method and found the measurements in line with theoretical predictions, showing the analytical importance of bias and variance of SPG. We hope that those results will impact the domain of model-based on-policy SPG, where leveraging dynamics model for trajectory simulation is the dominating approach for stochastic policy gradient.

Finally, we proposed MBMA module - an approach that leverages dynamics models for MA estimation at the cost of additional compute. We evaluated its performance against QMA, MBPO and PPO on-policy baselines. Our experiments showed that it compares favourably in terms of both sample efficiency and final performance on most of the tested environments. Following our theoretical analysis, we consider only on-policy SPG algorithms. We consider expanding MA analysis to off-policy setting an interesting avenue for future research. We release the code used for experiments url.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al. What matters in on-policy reinforcement learning? a large-scale empirical study. In *ICLR 2021-Ninth International Conference on Learning Representations*, 2021.

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

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.

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

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

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

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018.

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

Clavera, I., Fu, Y., and Abbeel, P. Model-augmented actor-critic: Backpropagating through paths. In *International Conference on Learning Representations*, 2019.

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.

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.

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.

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.

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

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.

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

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

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*, 2022a. 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/.

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

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.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

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

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.

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

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

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

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

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

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

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

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

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.

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.

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.

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

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.

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

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

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

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.

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

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

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.

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.

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.

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

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.

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.

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

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, 2021b.

A. Derivations - variance

First, we augment the notation to encompass many action samples:

$$\Upsilon^t_{s,a^n} = \nabla J(s_t, a^n_t), \quad \Upsilon^t_{s,a} = \nabla J(s_t, a_t) \quad \text{and} \quad \Upsilon^t_s = \mathop{\mathbb{E}}_{a\sim\pi} \Upsilon^t_{s,a}$$

For convenience, throughout the Appendix we will assume finite state and action spaces. However, same reasoning works for continuous spaces.

A.1. Derivation of Lemma 3.1

Following the MA-SPG definition outlined in Equation 3, $\mathrm{Var}_{s,a\sim p_0^\pi,\pi}\left[\Upsilon_{s,a}\right]$ is equal to:

$$\begin{aligned}
\mathop{\mathrm{Var}}_{s,a\sim p_0^\pi,\pi}\left[\Upsilon_{s,a}\right] &= \sum_s p_0^\pi(s) \prod_{n=1}^N \sum_{a^n} \pi(a^n|s) \left(\frac{\Upsilon_{s,a^1}}{N} + ... + \frac{\Upsilon_{s,a^N}}{N}\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{N}{N^2} \sum_s p_0^\pi(s) \sum_a \pi(a|s)\, (\Upsilon_{s,a})^2 + \frac{2}{N^2}\binom{N}{2} \sum_s p_0^\pi(s) \left(\sum_a \pi(a|s)\,\Upsilon_{s,a}\right)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} (\Upsilon_{s,a})^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} (\Upsilon_s)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} (\Upsilon_{s,a})^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} (\Upsilon_s)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} (\Upsilon_{s,a})^2 + \frac{N-1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} (\Upsilon_s)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \frac{1}{N}\left(\mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathbb{E}}_{a\sim\pi} (\Upsilon_{s,a})^2 - \mathop{\mathbb{E}}_{s\sim p_0^\pi} (\Upsilon_s)^2\right) + \mathop{\mathbb{E}}_{s\sim p_0^\pi} (\Upsilon_s)^2 - \left(\mathbb{E}\,\nabla J\right)^2 \\
&= \mathop{\mathrm{Var}}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + \frac{1}{N} \mathop{\mathbb{E}}_{s\sim p_0^\pi} \mathop{\mathrm{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right] \\
&= \mathop{\mathrm{Var}}_{s_0\sim p_0^\pi}\left[\mathop{\mathbb{E}}_{a_0\sim\pi} \nabla_\theta J(s_0, a_0)\right] + \frac{1}{N} \mathop{\mathbb{E}}_{s_0\sim p_0^\pi} \mathop{\mathrm{Var}}_{a_0\sim\pi}\left[\nabla J(s_0, a_0)\right]
\end{aligned} \tag{10}$$

The above result for $N = 1$ is reported in (Petit et al., 2019), noting it as stemming from law of total variance. However, we could not find the proof in existing literature. Below, $p_t^\pi(s_t|s_0, a_0^1)$ denotes the t step transition kernel conditioned on s_0 and a_0^1 (ie. the first sampled action in s_0).

$$\begin{aligned}
&\mathbb{E}\left[\Upsilon_{s,a}\Upsilon^t_{s,a}\right] = \\
&= \sum_{s_0} p_0^\pi(s_0) \prod_{n=1}^N \sum_{a_0^n} \pi(a_0^n|s_0) \sum_{s_t} p_t^\pi(s_t|s_0, a_0^1) \prod_{m=1}^N \sum_{a_t^m} \pi(a_t^m|s_t) \left(\frac{\Upsilon_{s,a^1}}{N} + ... + \frac{\Upsilon_{s,a^N}}{N}\right)\left(\frac{\Upsilon^t_{s,a^1}}{N} + ... + \frac{\Upsilon^t_{s,a^N}}{N}\right) \\
&= \sum_{s_0} p_0^\pi(s_0) \prod_{n=1}^N \sum_{a_0^n} \pi(a_0^n|s_0) \sum_{s_t} p_t^\pi(s_t|s_0, a_0^1) \prod_{m=1}^N \sum_{a_t^m} \pi(a_t^m|s_t) \left(\sum_{i=1}^N \sum_{j=1}^N \frac{\Upsilon_{s,a^i}}{N}\frac{\Upsilon^t_{s,a^j}}{N}\right)
\end{aligned}$$

Therefore:

$$
\begin{aligned}
\mathbb{E}\big[\Upsilon_{s,a}\Upsilon^t_{s,a}\big] = &= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a^1_0}\pi(a^1_0)\,\Upsilon_{s,a^1_0}\prod_{n=2}^{N}\sum_{a^n_0}\pi(a^n_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\sum_{a_t}\pi(a_t|s_t)\,\Upsilon^t_{s,a} \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a^2_0}\pi(a^2_0)\,\Upsilon_{s,a^2_0}\sum_{a^1_0}\pi(a^1_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\sum_{a_t}\pi(a_t|s_t)\,\Upsilon^t_{s,a} \\
&= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a^1_0}\pi(a^1_0)\,\Upsilon_{s,a^1_0}\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\,\Upsilon^t_s \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\,\Upsilon_s\sum_{a^1_0}\pi(a^1_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a^1_0)\,\Upsilon^t_s
\end{aligned}
$$

Thus, the t^{th} covariance of MA is equal to:

$$
\begin{aligned}
&\operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\Upsilon_{s,a},\Upsilon^t_{s,a}\big] = \\
&= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a_0}\pi(a_0)\,\Upsilon_{s,a}\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\,\Upsilon_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s \\
&- \Big(\sum_{s_0} p^\pi_0(s_0)\sum_{a_0}\pi(a_0)\,\Upsilon_{s,a}\Big)\Big(\sum_{s_t} p^\pi_t(s_t)\sum_{a_t}\pi(a_t)\,\Upsilon^t_{s,a}\Big) \\
&= \frac{1}{N}\sum_{s_0} p^\pi_0(s_0)\sum_{a_0}\pi(a_0)\,\Upsilon_{s,a}\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s \\
&+ \frac{N-1}{N}\sum_{s_0} p^\pi_0(s_0)\,\Upsilon_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s - \big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon_{s,a}\big)\big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon^t_{s,a}\big) \\
&= \frac{1}{N}\mathop{\mathbb{E}}_{s_0\sim p^\pi_0}\Big(\sum_{a_0}\pi(a_0)\,\Upsilon^0_{s,a}\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s - \Upsilon^0_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s\Big) \\
&+ \Big(\sum_{s_0} p^\pi_0(s_0)\,\Upsilon^0_s\sum_{a_0}\pi(a_0|s_0)\sum_{s_t}p^\pi_t(s_t|s_0,a_0)\,\Upsilon^t_s - \big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon_{s,a}\big)\big(\mathop{\mathbb{E}}_{a\sim\pi}\Upsilon^t_{s,a}\big)\Big) \\
&= \operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\Upsilon_s,\Upsilon^t_s\big] + \frac{1}{N}\mathop{\mathbb{E}}_{s_0\sim p^\pi_0}\operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\Upsilon_{s,a},\Upsilon^t_{s,a}\big] \\
&= \operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\mathop{\mathbb{E}}_{a_0\sim\pi}\nabla_\theta J(s_0,a_0),\mathop{\mathbb{E}}_{a_0\sim\pi}\nabla_\theta J(s_t,a_t)\big] + \frac{1}{N}\mathop{\mathbb{E}}_{s_0\sim p^\pi_0}\operatorname*{Cov}_{s_t,a_t\sim p^\pi_t,\pi}\big[\nabla J(s_0,a_0),\nabla J(s_t,a_t)\big]
\end{aligned}
\tag{11}
$$

Combining Equations 10 and 11 concludes derivation of Lemma 3.1.

A.2. Derivation of Lemma 3.2

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

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

We also use the shorthand notation:

$$\alpha_e^t = \underset{s_t,a_t\sim p_t^\pi,\pi}{\mathrm{Cov}}\big[\Upsilon_s,\Upsilon_s^t\big],\quad \alpha^t = \underset{s_t,a_t\sim p_t^\pi,\pi|s_0}{\mathrm{Cov}}\big[\Upsilon_{s,a},\Upsilon_{s,a}^t\big]\quad\text{and}\quad \mathrm{C}^t = \underset{s_t,a_t\sim p_t^\pi,\pi}{\mathrm{Cov}}\left[\Upsilon_{s,a},\Upsilon_{s,a}^t\right] = \alpha_e^t + \frac{1}{N}\underset{s\sim p_0^\pi}{\mathbb{E}}\alpha^t$$

Thus:

$$\mathbf{V} = \frac{1}{T}\left(\underset{s_0\sim p_0^\pi}{\mathrm{Var}}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_e^t + \frac{1}{N}\underset{s_0\sim p_0^\pi}{\mathbb{E}}\big(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}^0\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\big)\right)$$

We proceed with calculation of the forward difference:

$$\begin{aligned}
\Delta_N &= \frac{1}{T}\left(\underset{s\sim p_0^\pi}{\mathrm{Var}}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_e^t + \frac{1}{N+1}\underset{s\sim p_0^\pi}{\mathbb{E}}\big(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\big)\right)\\
&- \frac{1}{T}\left(\underset{s\sim p_0^\pi}{\mathrm{Var}}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha_e^t + \frac{1}{N}\underset{s\sim p_0^\pi}{\mathbb{E}}\big(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\big)\right)\\
&= \frac{1}{T(N+1)}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\right)\\
&- \frac{1}{T\,N}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\right)\\
&= \frac{-1}{T(N^2+N)}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\,\alpha^t\right)\\
&= \frac{-1}{T(N^2+N)}\underset{s\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\underset{s_t,a_t\sim p_t^\pi,\pi|s_0}{\mathrm{Cov}}\left[\Upsilon_{s,a}^0,\Upsilon_{s,a}^t\right]\right)\\
&= \frac{-1}{T(N^2+N)}\underset{s_0\sim p_0^\pi}{\mathbb{E}}\left(\underset{a\sim\pi}{\mathrm{Var}}\left[\nabla J(s_0,a_0)\right] + 2\sum_{t=1}^{T-1}\frac{T-t}{T}\underset{s_t,a_t\sim p_t^\pi,\pi|s_0}{\mathrm{Cov}}\left[\nabla J(s_0,a_0),\nabla J(s_t,a_t)\right]\right)
\end{aligned}\tag{12}$$

Similarly, we calculate Δ_T:

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

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

$$\begin{aligned}
\Delta_T &= \frac{-\delta}{T+\delta T}\left(\underset{s,a\sim p_0^\pi,\pi}{\mathrm{Var}}[\Upsilon_{s,a}] + 2\sum_{t=1}^{T-1}\big(\frac{T-t}{T} - \frac{t}{T+\delta T}\big)\,\mathrm{C}^t\right)\\
&= \frac{-\delta}{T+\delta T}\left(\underset{s,a\sim p_0^\pi,\pi}{\mathrm{Var}}[\Upsilon_{s,a}] + 2\sum_{t=1}^{T-1}\big(\frac{T-t}{T} - \frac{t}{T+\delta T}\big)\underset{s_t,a_t\sim p_t^\pi,\pi}{\mathrm{Cov}}\left[\Upsilon_{s,a}^0,\Upsilon_{s,a}^t\right]\right)
\end{aligned}\tag{13}$$

Combining Equations 12 and 13 concludes derivation of Lemma 3.2.

A.3. Derivation of Theorem 3.3

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

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

We use Equations 10 and 11 to expand the RHS:

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

We move all terms dependent on the policy to the LHS:

$$\begin{aligned}&\frac{1-\delta N}{\delta(N^2+N)}\mathop{\mathbb{E}}_{s\sim p_0^\pi}\mathop{\text{Var}}_{a\sim\pi}\left[\Upsilon_{s,a}\right] + 2\sum_{t=1}^{T-1}\left(\frac{(1+\delta-\delta N-\delta^2 N)T-(1-2\delta N-\delta^2 N)t}{(\delta T+\delta^2 T)(N^2+N)}\right)\alpha^t \geq \\ &\mathop{\text{Var}}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1}\left(\frac{T-t}{T} - \frac{t}{T+\delta T}\right)\alpha_e^t\end{aligned} \tag{14}$$

Now, in order to recover the Corollary 9, we assume a contextual bandit setup (ie. $p^\pi(s'|s) = p^\pi(s')$). Then:

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

Which is equivalent to:

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

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 14 simplifies to:

$$
\begin{aligned}
&\sum_{t=1}^{T-1} \frac{t}{T}\, \alpha^t \geq \operatorname*{Var}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + \sum_{t=1}^{T-1} \frac{2T-3t}{T}\, \alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\, \alpha^t \geq \operatorname*{Var}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\, \alpha_e^t - \sum_{t=1}^{T-1} \frac{t}{T}\, \alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\, \left(\alpha^t + \alpha_e^t\right) \geq \operatorname*{Var}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\, \alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\, \mathrm{C}_t \geq \operatorname*{Var}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\, \alpha_e^t \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\, \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\left[\Upsilon_{s,a}, \Upsilon_{s,a}^t\right] \geq \operatorname*{Var}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\, \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\left[\Upsilon_s, \Upsilon_s^t\right] \\
&\equiv \sum_{t=1}^{T-1} \frac{t}{T}\, \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\left[\Upsilon_{s,a}, \Upsilon_{s,a}^t\right] \geq \operatorname*{Var}_{s\sim p_0^\pi}\left[\Upsilon_s\right] + 2\sum_{t=1}^{T-1} \frac{T-t}{T}\, \operatorname*{Cov}_{s_t,a_t\sim p_t^\pi,\pi}\left[\Upsilon_s, \Upsilon_s^t\right]
\end{aligned}
\tag{15}
$$

Which concludes the derivation of Theorem 3.3.

B. Derivations - bias

First we calculate the bias associated with MA (ie. MBMA and QMA), which stems from approximated state-action Q-value. We denote the approximated Q-value as $\hat{Q}^\pi(s,a)$ and write:

$$\begin{aligned}\text{bias}^{MA} &= \nabla J(s,a) - \nabla \hat{J}(s,a) \\ &= \nabla \log \pi(a|s)\, Q^\pi(s,a) - \nabla \log \pi(a|s)\, \hat{Q}^\pi(s,a) \\ &= \nabla \log \pi(a|s) \big(Q^\pi(s,a) - \hat{Q}^\pi(s,a)\big)\end{aligned} \tag{16}$$

Furthermore, we calculate bias associated with using dynamics models to simulate state samples. Firstly, we denote the result of a n-step transition via dynamics model as s^*, such that the absolute difference between true transition and dynamics model transition is equal to $|s - s^*|$. Furthermore, we denote Lipschitz norm of $\nabla \log \pi(a|s)$ as $\mathcal{K}$. As such, it follows that:

$$|\nabla \log \pi(a|s) - \nabla \log \pi(a|s^*)| \leq \mathcal{K}|s - s^*|$$

We write the bias:

$$\begin{aligned}\text{bias}^{MS} &= \nabla J(s,a) - \nabla \hat{J}(s,a) \\ &= \nabla \log \pi(a|s)\, Q^\pi(s,a) - \nabla \log \pi(a|s^*)\, \hat{Q}^\pi(s^*,a) \\ &= \big(\nabla \log \pi(a|s) - \nabla \log \pi(a|s^*)\big)\hat{Q}^\pi(s^*,a) + \nabla \log \pi(a|s)\big(Q^\pi(s,a) - \hat{Q}^\pi(s^*,a)\big)\end{aligned}$$

We use the Lipschitz continuity:

$$\left| \frac{\text{bias}^{MS} - \nabla \log \pi(a|s)\big(Q^\pi(s,a) - \hat{Q}^\pi(s^*,a)\big)}{\hat{Q}^\pi(s^*,a)} \right| \leq \mathcal{K}|s - s^*|$$

Where we assume that $\hat{Q}^\pi(s^*,a) \neq 0$. Squaring both sides leads to the solution:

$$\begin{aligned}&\text{bias}^{MS} \geq \nabla \log \pi(a|s)\big(Q^\pi(s,a) - \hat{Q}^\pi(s,a)\big) - \sqrt{\nabla \log \pi(a|s)^2\big(Q^\pi(s,a)^2 - Q^\pi(s,a)\big) + \big(\mathcal{K}(s-s^*)\big)^2} \\ &\text{And:} \\ &\text{bias}^{MS} \leq \nabla \log \pi(a|s)\big(Q^\pi(s,a) - \hat{Q}^\pi(s,a)\big) + \sqrt{\nabla \log \pi(a|s)^2\big(Q^\pi(s,a)^2 - Q^\pi(s,a)\big) + \big(\mathcal{K}(s-s^*)\big)^2}\end{aligned} \tag{17}$$

Which concludes the derivation.

C. Experimental setting

C.1. Figure 1

In Figure 1, we use the OpenAI gym CartPole environment. We define solving the environment as reaching an average of 190 reward during 25 evaluations. We perform policy evaluations every 50 environment steps. If the trajectory length is shorter than environment termination we bootstrap the Q-value with critic. To sample more actions per state we perform environment rewinding. Similarly to regular actions, the Q-values of additional action samples are bootstrapped via critic when reaching the trajectory length. Note that CartPole environment has only two actions, as such there is minimal variance associated with policy. We smoothen the results with Savitsky-Golay filter and use 45 random seeds.

C.2. Table 1

In Table 1, we use a subset of environments from DM Control Suite. We marginalize Q-values by performing 100 rollouts for every state-action. We get 125000 on-policy states, with one additional action per state. We use Equation 3 and Lemma 3.1 to isolate the variance components. Note that Q-value marginalization is required by Lemma 3.1. Note that if Q-values are stochastic, we observe more variance reduction stemming from sampling additional actions than expected. The performance of agents was measured during 500000 environment steps, with average performance recorded in 122 different episodes. Additional action sample is drawn from the environments (via environment rewinding). To reduce the compute load used in the experiment, the performance is measured without Q-value marginalization. We use 10 random seeds.

C.3. Table 3

In Table 3, we use 14 different environments from DM Control Suite. We measure performance in 244 different episodes throughout the training. We perform single-sided Welsch t-test to compare two highest performing agents with 95% confidence. To record the normalized average performance, we first divide each record by the PPO performance in this task (as such PPO has normalized performance equal to 1) and the calculate average across tasks. When calculating performance metrics, we first average recorded results over seeds and later take maximum or average over the step dimension (for *best performance* and *best performance* respectively). The code used in the experiment is available url. We use 15 random seeds.

C.4. Table 4

In Table 3, we use 14 different environments from DM Control Suite. We record 125 gradient estimates for every method during 10 points in training for 15 random seeds. Each of 125 gradient estimates is calculated using a batch size of 2500 states. The gradients are always calculated wrt. the same policy. To this end, there is one agent gathering the data and serving as policy for all methods, and the recorded gradients stemming from all methods are never applied to the actor network (ie. using one agent per random seed). We denote $*$ as the tested method and P as the total number of parameters in the model. We calculate relative bias with the following equation:

$$\text{Bias}^* = \frac{1}{P}\sum_{p}^{P}\frac{|\nabla J_p^* - \nabla J_p^{AC}|}{|\nabla J_p^*|}$$

Where ∇J_p^* and ∇J_p^{AC} denote the gradient wrt. p^{th} parameter calculated via tested method and actor critic respectively (averaged over 125 gradient examples). As such, at each testing point we calculate the absolute difference between the 'oracle' AC gradient (which is unbiased) and the respective method average. Furthermore, we calculate relative variance via:

$$\text{Var}^* = \frac{1}{P}\sum_{p}^{P}\frac{\underset{\tau}{\text{Var}}\left[\nabla J_p^*\right]}{(\nabla J_p^*)^2}$$

Where $\text{Var}_\tau\left[\nabla J_p^*\right]$ denotes the p^{th} unit of the diagonal of variance-covariance matrix calculated over 125 gradient examples. Dividing bias and variance by the size of gradient allows to inspect the relative size (i.e. if gradient is small then bias and variance might also be small, but big in comparison to the gradient that we are looking for). The results in Table 3 represents averages across 10 points during the training and 15 random seeds.

C.5. Hyperparameters

To allow for experiment validation and reproduction we provide a detailed list of hyperparameter settings in the table below.

HYPERPARAMETER	PPO	QMA	MBMA	MBPO
ACTION REPEAT	4	4	4	4
ACTOR OPTIMIZER	ADAM	ADAM	ADAM	ADAM
CRITIC OPTIMIZER	ADAM	ADAM	ADAM	ADAM
DYNAMICS OPTIMIZER	—	—	ADAM	ADAM
Q-NET OPTIMIZER	—	ADAM	—	—
ACTOR LEARNING RATE	$3e-4$	$3e-4$	$3e-4$	$3e-4$
CRITIC LEARNING RATE	$3e-4$	$3e-4$	$3e-4$	$3e-4$
DYNAMICS LEARNING RATE	—	—	$3e-4$	$3e-4$
Q-NET LEARNING RATE	—	$3e-4$	—	—
ACTOR OPTIMIZER EPSILON	$1e-5$	$1e-5$	$1e-5$	$1e-5$
CRITIC OPTIMIZER EPSILON	$1e-5$	$1e-5$	$1e-5$	$1e-5$
DYNAMICS OPTIMIZER EPSILON	—	—	$1e-5$	$1e-5$
Q-NET OPTIMIZER EPSILON	—	$1e-5$	—	—
ACTOR HIDDEN LAYER SIZE	512	512	512	512
CRITIC HIDDEN LAYER SIZE	1024	1024	1024	1024
DYNAMICS HIDDEN LAYER SIZE	—	—	1024	1024
Q-NETWORK HIDDEN LAYER SIZE	—	1024	—	—
λ	0.95	0.95	0.95	0.95
DISCOUNT RATE	0.99	0.99	0.99	0.99
BATCH SIZE (T)	2048	2048	2048	2048
MINIBATCH SIZE	64	64	64	64
PPO EPOCHS	10	10	10	10
DYNAMICS BUFFER SIZE	—	—	25000	25000
DYNAMICS BATCH SIZE	—	—	128	128
NUMBER OF SIMULATED ACTIONS PER STATE (T*)	—	8	8	—
NUMBER OF SIMULATED STATES PER STATE (N)	—	—	—	8
SIMULATION HORIZON	—	—	10	10
CLIP COEFFICIENT	0.2	0.2	0.2	0.2
MAXIMUM GRADIENT NORM	0.5	0.5	0.5	0.5
VALUE COEFFICIENT	0.5	0.5	0.5	0.5

C.6. Implementation

D. Extended results - statistics

We also provide table of full-sample statistics for the performance experiment:

TASK	PPO	MBMA	MBPO	QMA
ACRO SWINGUP	$47 \pm 10\,(32 \pm 7)$	$\mathbf{59 \pm 6}\,(\mathbf{40 \pm 6})$	$56 \pm 8\,(31 \pm 7)$	$34 \pm 10\,(17 \pm 7)$
BALL CATCH	$948 \pm 8\,(831 \pm 20)$	$\mathbf{974 \pm 3}\,(\mathbf{898 \pm 8})$	$969 \pm 2\,(888 \pm 8)$	$934 \pm 5\,(770 \pm 30)$
CART SWINGUP	$736 \pm 46\,(617 \pm 46)$	$\mathbf{828 \pm 16}\,(\mathbf{707 \pm 44})$	$825 \pm 13\,(702 \pm 38)$	$802 \pm 2\,(677 \pm 18)$
CART 2-POLE	$308 \pm 16\,(253 \pm 8)$	$\mathbf{575 \pm 52}\,(\mathbf{388 \pm 27})$	$435 \pm 48\,(317 \pm 22)$	$315 \pm 22\,(248 \pm 12)$
CART 3-POLE	$229 \pm 15\,(199 \pm 10)$	$229 \pm 14\,(202 \pm 10)$	$261 \pm 6\,(\mathbf{221 \pm 9})$	$\mathbf{262 \pm 5}\,(211 \pm 4)$
CHEETAH RUN	$283 \pm 12\,(185 \pm 10)$	$\mathbf{507 \pm 14}\,(\mathbf{316 \pm 14})$	$473 \pm 17\,(284 \pm 14)$	$201 \pm 8\,(135 \pm 7)$
FINGER SPIN	$\mathbf{391 \pm 21}\,(\mathbf{280 \pm 14})$	$350 \pm 14\,(266 \pm 12)$	$305 \pm 16\,(248 \pm 14)$	$359 \pm 15\,(245 \pm 12)$
FINGER TURN	$\mathbf{396 \pm 67}\,(\mathbf{213 \pm 54})$	$368 \pm 59\,(206 \pm 52)$	$318 \pm 73\,(184 \pm 51)$	$296 \pm 75\,(176 \pm 50)$
POINT EASY	$895 \pm 6\,(839 \pm 13)$	$\mathbf{910 \pm 5}\,(866 \pm 7)$	$909 \pm 6\,(\mathbf{867 \pm 7})$	$467 \pm 97\,(106 \pm 50)$
REACHER EASY	$885 \pm 44\,(649 \pm 66)$	$\mathbf{968 \pm 2}\,(\mathbf{815 \pm 39})$	$854 \pm 71\,(729 \pm 74)$	$472 \pm 27\,(316 \pm 72)$
REACHER HARD	$601 \pm 103\,(385 \pm 78)$	$767 \pm 96\,(606 \pm 84)$	$\mathbf{892 \pm 61}\,(\mathbf{722 \pm 61})$	$488 \pm 53\,(361 \pm 47)$
WALKER STAND	$914 \pm 22\,(737 \pm 24)$	$\mathbf{955 \pm 5}\,(\mathbf{839 \pm 22})$	$944 \pm 8\,(815 \pm 29)$	$854 \pm 20\,(654 \pm 21)$
WALKER WALK	$514 \pm 14\,(377 \pm 14)$	$\mathbf{892 \pm 9}\,(\mathbf{686 \pm 18})$	$720 \pm 19\,(576 \pm 19)$	$500 \pm 17\,(340 \pm 14)$
WALKER RUN	$203 \pm 7\,(152 \pm 5)$	$\mathbf{331 \pm 13}\,(\mathbf{251 \pm 9})$	$233 \pm 12\,(190 \pm 11)$	$208 \pm 7\,(141 \pm 4)$

And for the bias-variance experiment:

	RELATIVE BIAS			RELATIVE VARIANCE			
TASK	MBMA	MBPO	QMA	AC	MBMA	MBPO	QMA
ACRO SWINGUP	$\mathbf{8.25 \pm 0.3}$	8.66 ± 0.3	10.5 ± 0.5	44.3 ± 1.5	31.6 ± 1.1	30.8 ± 1.0	$\mathbf{27.3 \pm 1.1}$
BALL CATCH	$\mathbf{5.28 \pm 0.3}$	6.04 ± 0.3	12.1 ± 0.8	48.7 ± 1.4	20.0 ± 0.9	$\mathbf{18.0 \pm 0.7}$	18.8 ± 1.1
CART SWINGUP	$\mathbf{3.16 \pm 0.3}$	3.46 ± 0.3	8.89 ± 1.1	21.2 ± 1.5	8.08 ± 0.6	$\mathbf{7.84 \pm 0.6}$	11.1 ± 1.1
CART 2-POLE	$\mathbf{6.32 \pm 0.4}$	6.79 ± 0.5	10.6 ± 0.7	39.8 ± 1.8	21.6 ± 1.6	$\mathbf{20.5 \pm 1.6}$	29.8 ± 1.4
CART 3-POLE	7.48 ± 0.7	$\mathbf{7.29 \pm 0.6}$	11.0 ± 0.7	43.0 ± 2.7	$\mathbf{27.7 \pm 2.7}$	29.6 ± 2.8	32.4 ± 2.3
CHEETAH RUN	$\mathbf{11.5 \pm 0.4}$	12.0 ± 0.4	21.4 ± 0.8	77.1 ± 2.5	39.0 ± 1.1	$\mathbf{37.7 \pm 1.1}$	52.9 ± 1.7
FINGER SPIN	$\mathbf{4.50 \pm 0.3}$	4.91 ± 0.4	9.60 ± 0.4	38.8 ± 1.1	$\mathbf{24.2 \pm 1.3}$	33.6 ± 2.0	28.2 ± 0.9
FINGER TURN	$\mathbf{3.55 \pm 0.4}$	3.94 ± 0.4	11.3 ± 0.8	51.0 ± 2.8	$\mathbf{25.4 \pm 1.8}$	30.9 ± 2.7	30.4 ± 2.1
POINT EASY	$\mathbf{1.33 \pm 0.1}$	1.49 ± 0.1	3.57 ± 0.6	15.8 ± 1.4	4.40 ± 0.4	$\mathbf{3.84 \pm 0.3}$	4.08 ± 0.5
REACHER EASY	$\mathbf{4.07 \pm 0.2}$	4.85 ± 0.3	10.3 ± 1.0	36.2 ± 1.8	14.3 ± 0.7	$\mathbf{13.8 \pm 0.7}$	15.6 ± 1.4
REACHER HARD	$\mathbf{5.70 \pm 0.3}$	6.58 ± 0.4	13.0 ± 0.7	41.6 ± 1.4	21.1 ± 1.2	$\mathbf{20.0 \pm 1.1}$	23.1 ± 1.4
WALKER STAND	$\mathbf{9.77 \pm 0.4}$	11.4 ± 0.5	18.8 ± 0.8	71.2 ± 2.0	$\mathbf{39.9 \pm 1.2}$	43.0 ± 1.4	51.5 ± 2.1
WALKER WALK	$\mathbf{11.1 \pm 0.4}$	12.3 ± 0.4	16.6 ± 0.8	76.7 ± 1.8	$\mathbf{40.7 \pm 1.2}$	42.0 ± 1.2	59.4 ± 1.1
WALKER RUN	$\mathbf{11.1 \pm 0.4}$	12.1 ± 0.4	15.7 ± 0.8	77.9 ± 1.8	41.3 ± 1.3	$\mathbf{40.7 \pm 1.0}$	59.5 ± 1.6

E. Extended results - ablations

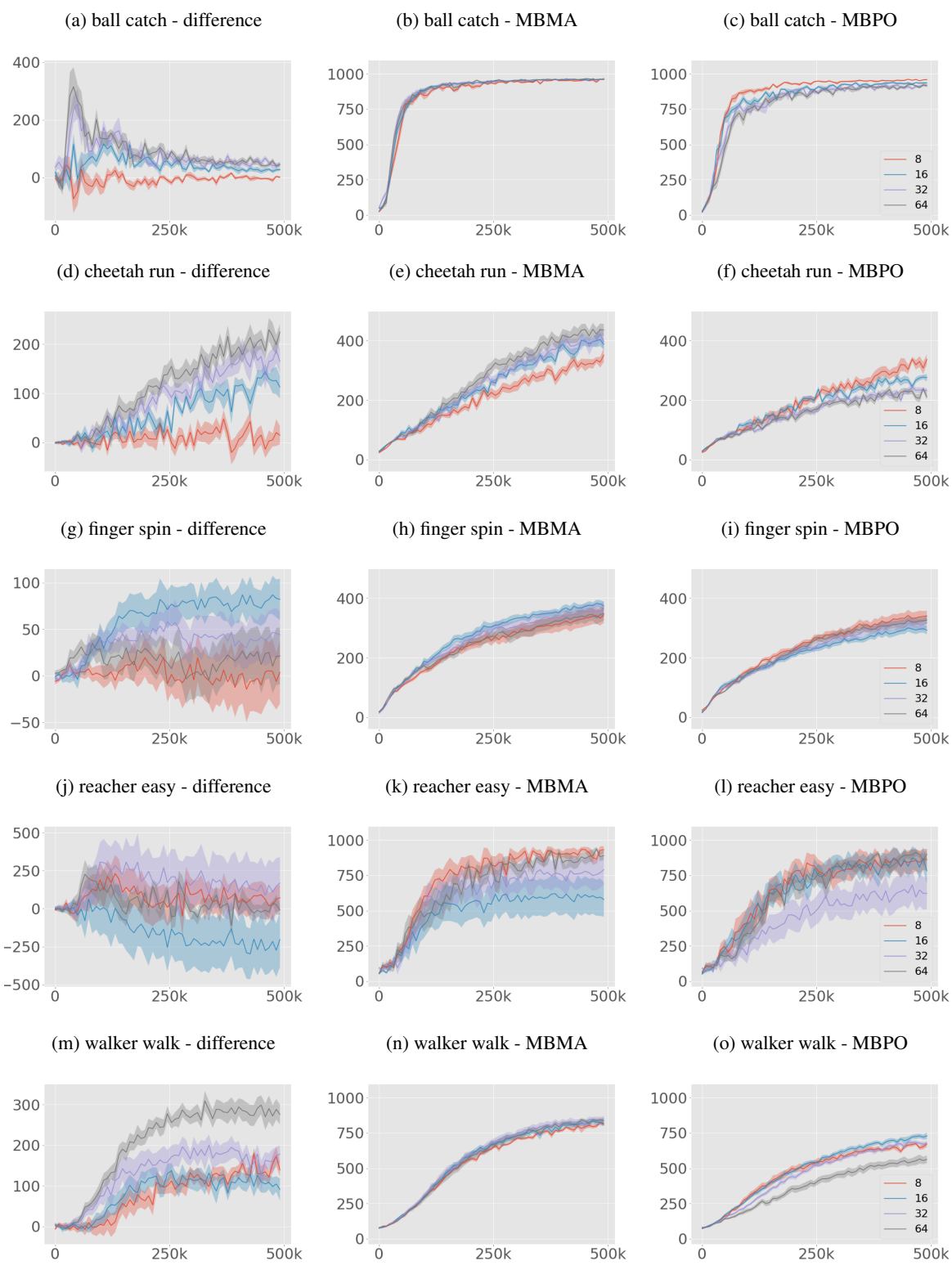

Figure 2. Comparison of MBMA and MBPO for different N and T. The left column shows performance difference between MBMA and MBPO (positive when MBMA performs better). Middle and right columns show performance of MBMA and MBPO.

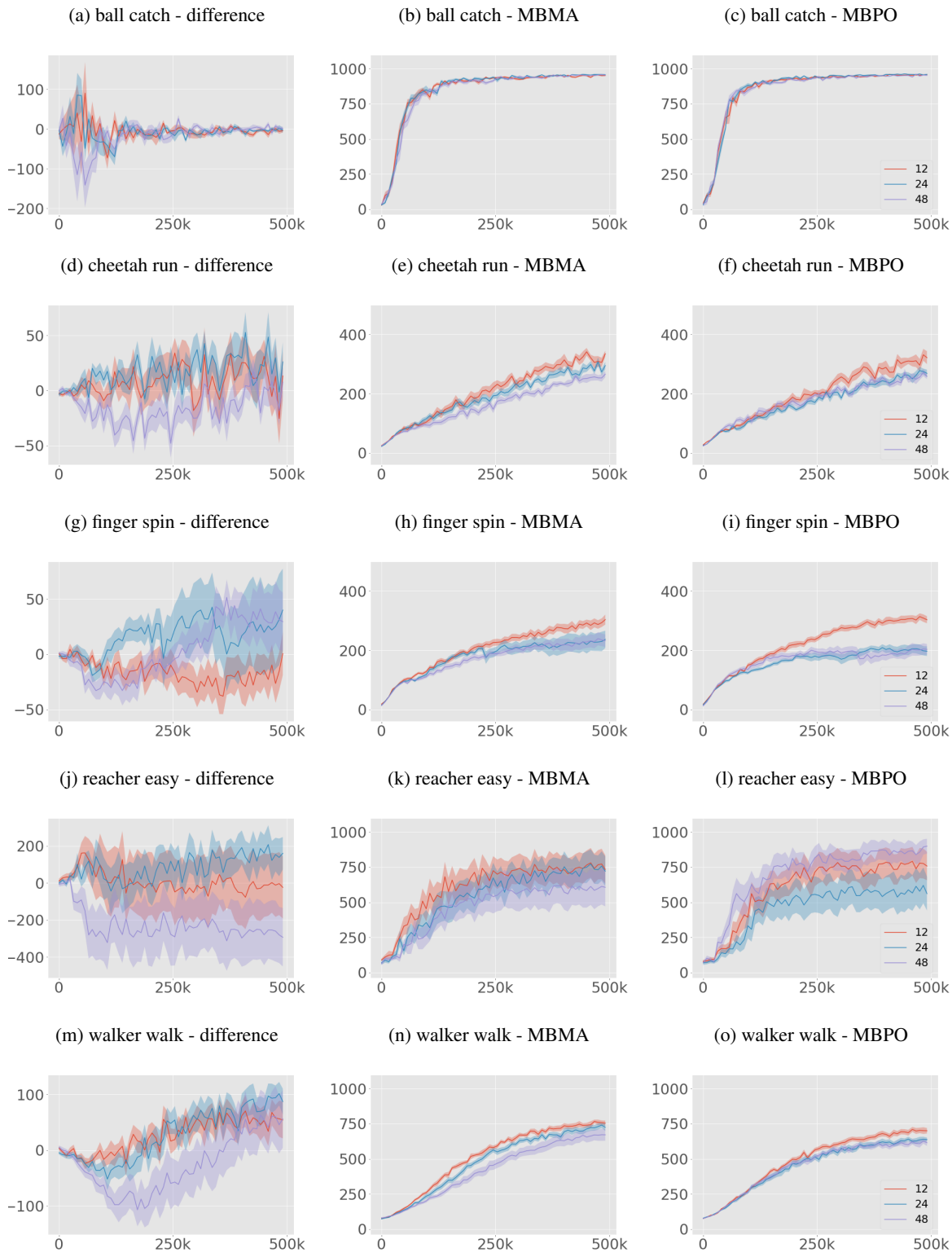

Figure 3. Comparison of MBMA and MBPO for different simulation horizons. The left column shows performance difference between MBMA and MBPO (positive when MBMA performs better). Middle and right columns show performance of MBMA and MBPO.

We find that MBMA is more robust to hyperparameters setting as compared to MBPO. This most likely stems from more favourable bias variance structure of MBMA.

F. Extended results - performance

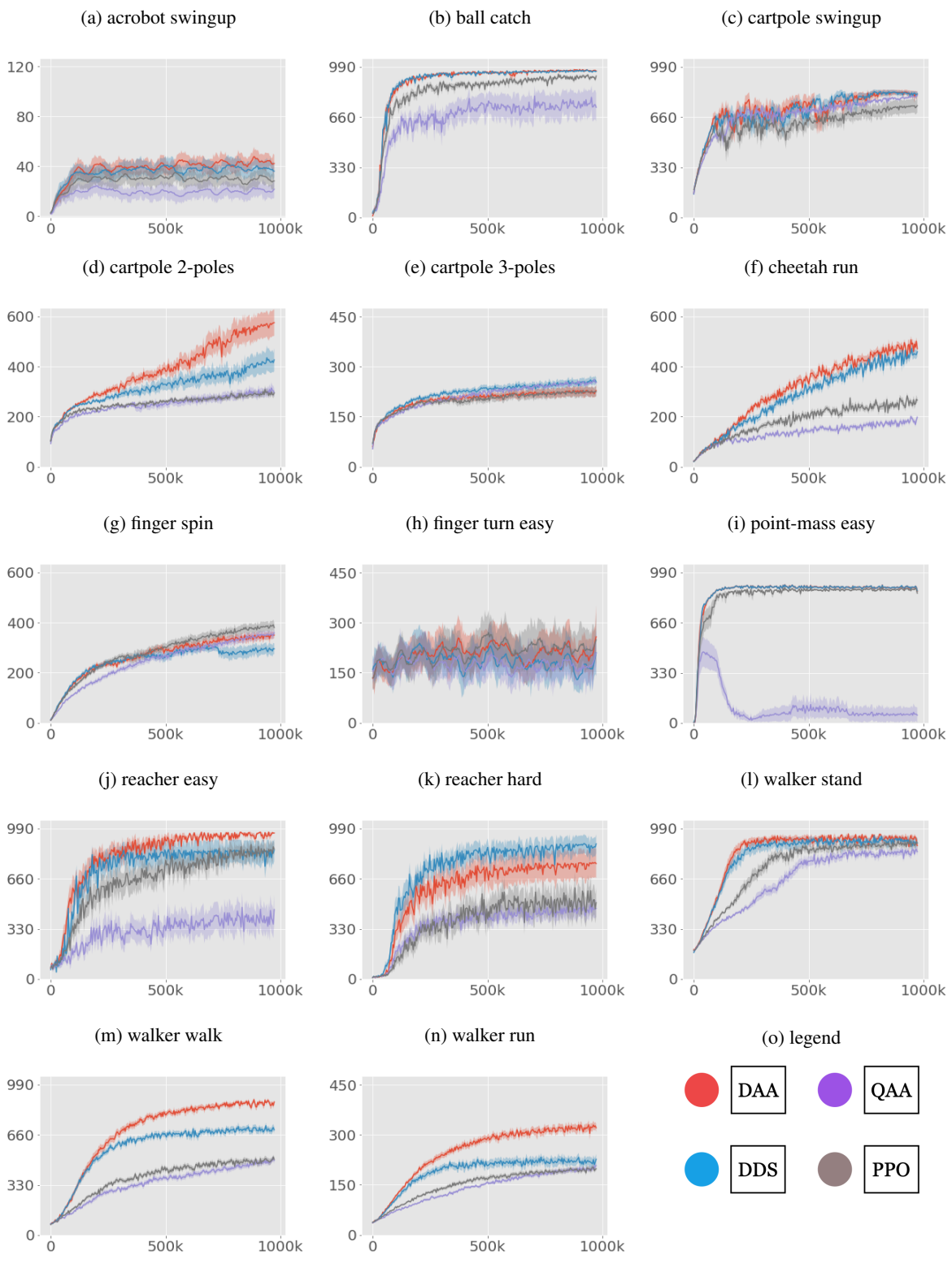

Figure 4. Learning curves corresponding to results from Table 3. Shaded region denotes one standard deviation of the mean. 15 seeds.

G. Extended results - bias/variance

Below we show the recorded bias and variance throughout the training.

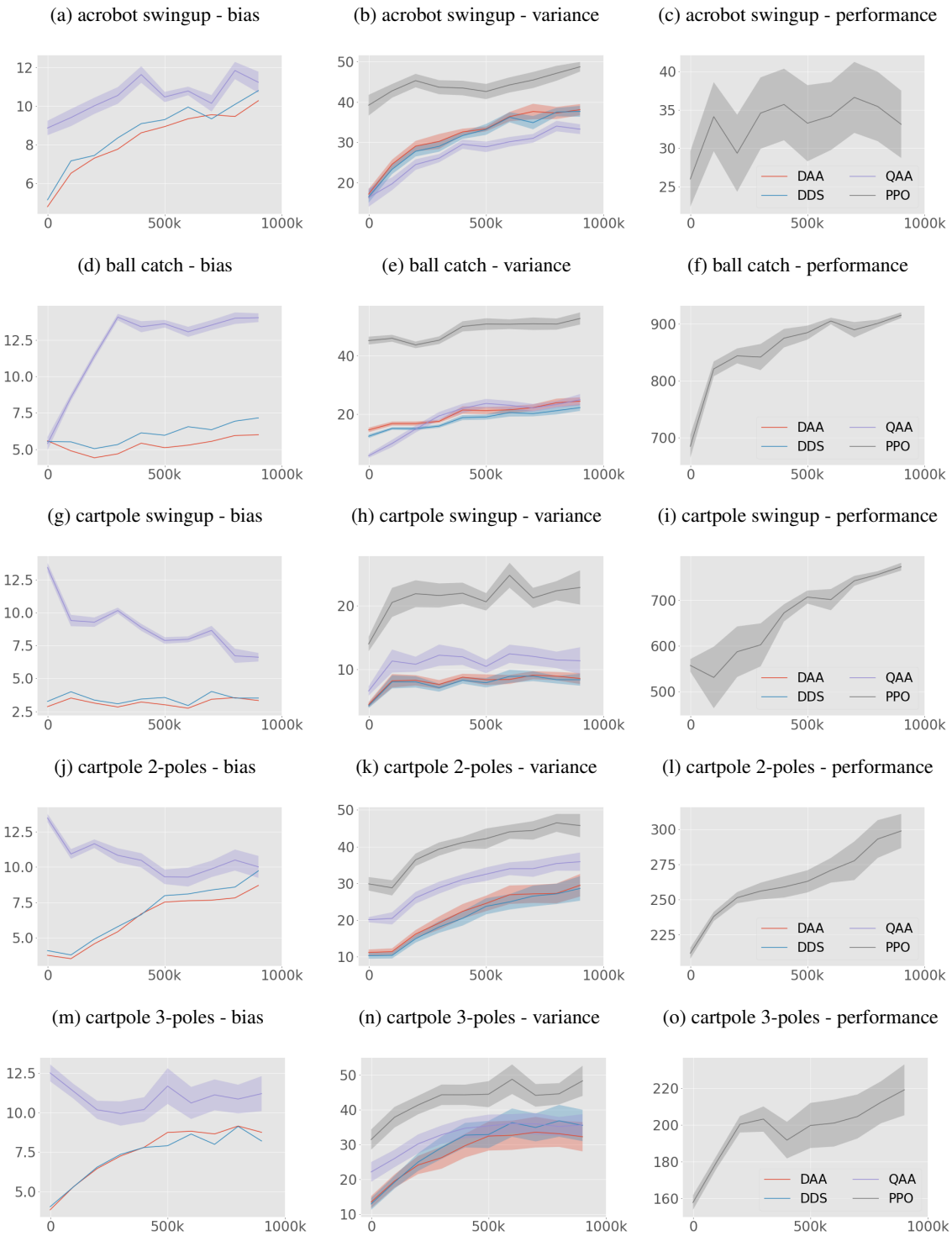

Figure 5. Bias and variance curves. Part 1 of 3.

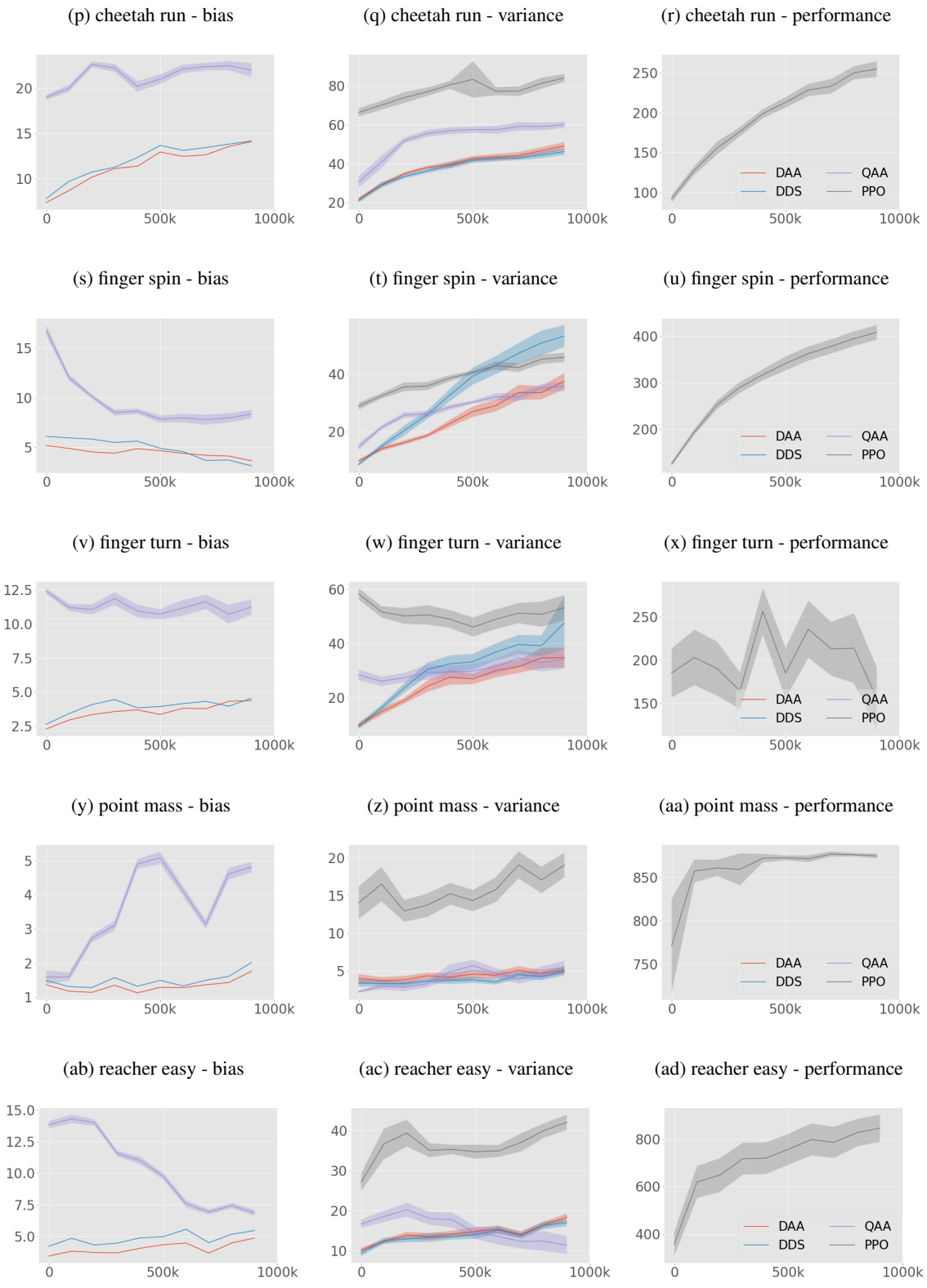

Figure 5. Bias and variance curves. Part 2 of 3.

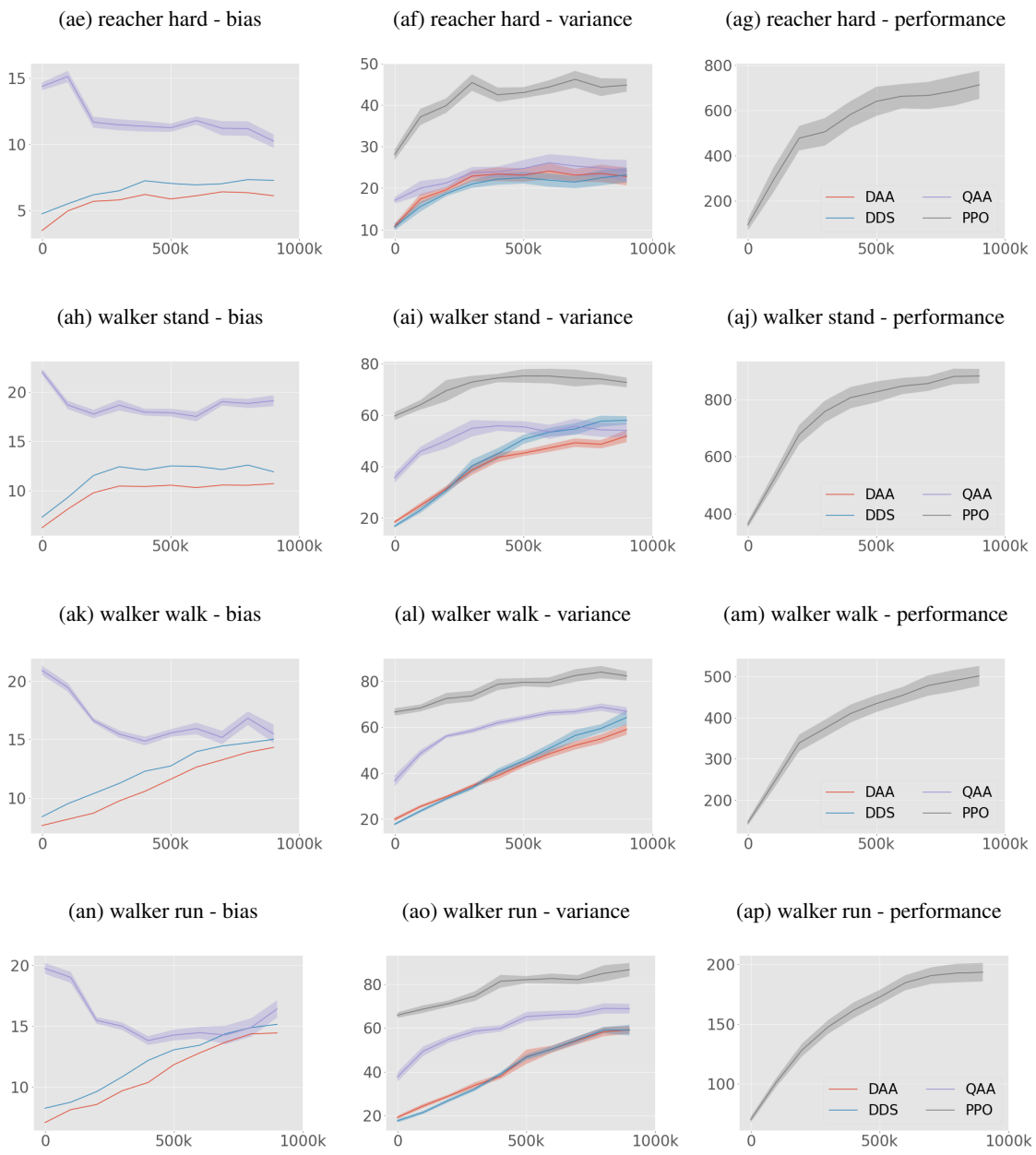

Figure 5. Bias and variance curves. Part 3 of 3.