Clipped Action Policy Gradient

Yasuhiro Fujita [1] **Shin-ichi Maeda** [1]

Abstract

Many continuous control tasks have bounded action spaces. When policy gradient methods are applied to such tasks, out-of-bound actions need to be clipped before execution, while policies are usually optimized as if the actions are not clipped. We propose a policy gradient estimator that exploits the knowledge of actions being clipped to reduce the variance in estimation. We prove that our estimator, named clipped action policy gradient (CAPG), is unbiased and achieves lower variance than the conventional estimator that ignores action bounds. Experimental results demonstrate that CAPG generally outperforms the conventional estimator, indicating that it is a better policy gradient estimator for continuous control tasks. The source code is available at `https://github.com/pfnet-research/capg`.

1. Introduction

Reinforcement learning (RL) has achieved remarkable success in recent years in a wide range of challenging tasks, such as games (Mnih et al., 2015; Silver et al., 2016; 2017), robotic manipulation (Levine et al., 2016), and locomotion (Schulman et al., 2015; 2017; Heess et al., 2017), with the help of deep neural networks. Policy gradient methods are among the most successful model-free RL algorithms (Mnih et al., 2016; Schulman et al., 2015; 2017; Gu et al., 2017b). They are particularly suitable for continuous control tasks, i.e., environments with continuous action spaces, because they directly improve policies that represent continuous distributions of actions to maximize expected returns. For continuous control tasks, policies are typically represented by Gaussian distributions conditioned on current and past observations.

Although Gaussian policies have unbounded support, continuous control tasks often have bounded action sets that they can execute (Duan et al., 2016; Brockman et al., 2016; Tassa et al., 2018). For example, when controlling the torques of motors, effective torque values will be physically constrained. Policies with unbounded support like Gaussian policies are usually applied to such tasks by clipping sampled actions into their bounds (Duan et al., 2016; Dhariwal et al., 2017). Policy gradients for such policies are estimated as if actions were not clipped (Chou et al., 2017).

In this study, we demonstrate that we can improve policy gradient methods by exploiting the knowledge of actions being clipped. We prove that the variance of policy gradient estimates can be strictly reduced under mild assumptions that hold for popular policy representations such as Gaussian policies with diagonal covariance matrices. Our proposed algorithm, named clipped action policy gradient (CAPG), is an alternative unbiased policy gradient estimator with a lower variance than the conventional estimator. Our experimental results on MuJoCo-simulated continuous control benchmark problems (Todorov et al., 2012; Brockman et al., 2016) show that CAPG can improve the performance of existing policy gradient-based deep RL algorithms.

2. Preliminaries

We consider a Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{U}, P, r, \rho_0, \gamma)$, where $\mathcal{S}$ is a set of possible states, $\mathcal{U}$ is a set of possible actions, P is a state-transition probability distribution, $r : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$ is a reward function, ρ_0 is a distribution of the initial state s_0, and $\gamma \in (0, 1]$ is a discount factor.

A probability distribution of action conditioned on state is referred to as a policy. The probability density function (PDF) of a policy is denoted by π. RL algorithms aim to find a policy that maximizes the expected cumulative discounted reward from initial states,

$$\eta(\pi) = \mathbb{E}_{s_0,u_0,\dots}\Big[\sum_t \gamma^t r(s_t, u_t)\Big|\pi\Big],$$

where $\mathbb{E}_{s_0,u_0,\dots}[\cdot|\pi]$ denotes an expected value with respect to a state-action sequence $s_0 \sim \rho_0(\cdot), u_0 \sim \pi(\cdot|s_0), s_1 \sim P(\cdot|s_0, u_0), u_1 \sim \pi(\cdot|s_1), \dots$.

[1]Preferred Networks, Inc., Japan. Correspondence to: Yasuhiro Fujita <fujita@preferred.jp>, Shin-ichi Maeda <ichi@preferred.jp>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

The state-action value function of a policy π is defined as

$$Q^\pi(s,u) = \mathbb{E}_{s_1,u_1,\dots}\Big[\sum_t \gamma^t r(s_t,u_t)\Big|s_0=s,u_0=u,\pi\Big].$$

One way to find $\pi^* = \mathrm{argmax}_\pi \eta(\pi)$ is to adjust the parameters θ of a parameterized policy π_θ by following the gradient $\nabla_\theta\eta(\pi_\theta)$, which is referred to a policy gradient. The policy gradient theorem (Sutton et al., 1999) states that

$$\nabla_\theta\eta(\pi_\theta) = \mathbb{E}_s\Big[\mathbb{E}_u[Q^{\pi_\theta}(s,u)\psi(s,u)|s]\Big],$$

where $\psi(s,u) = \nabla_\theta \log \pi_\theta(u|s)$, $\mathbb{E}_u[\cdot|s]$ denotes a conditional expected value with respect to $\pi_\theta(\cdot|s)$, and $\mathbb{E}_s[\cdot]$ denotes an (improper) expected value with respect to the (improper) discounted state distribution $\rho^{\pi_\theta}(\cdot)$, which is defined as

$$\rho^\pi(s) = \sum_t \gamma^t \int \rho_0(s_0)p(s_t=s|s_0,\pi)ds_0.$$

In practice, the policy gradient is often estimated by a finite number of samples $\{(s^{(i)},u^{(i)})|u^{(i)}\sim\pi_\theta(\cdot|s^{(i)}), i=1,\dots,N\}$.

$$\nabla_\theta\eta(\pi_\theta) \approx \frac{1}{N}\sum_i Q^{\pi_\theta}(s^{(i)},u^{(i)})\psi(s^{(i)},u^{(i)}). \quad (1)$$

RL algorithms that rely on this estimation are referred to as policy gradient methods. While this estimation is unbiased, its variance is typically high and is considered as a crucial problem of policy gradient methods.

We address the problem by estimating $\nabla_\theta\eta(\pi_\theta)$ in an unbiased and lower-variance[1] manner than (1). To this end, we derive a random variable Y such that $\mathbb{V}[Y] \le \mathbb{V}[X]$ and $\mathbb{E}[Y] = \mathbb{E}[X]$, where $X = Q^{\pi_\theta}(s,u)\psi(s,u)$. Because $\mathbb{E}[X] = \mathbb{E}_s[\mathbb{E}_u[X|s]]$ and $\mathbb{V}[X] = \mathbb{V}_s[\mathbb{E}_u[X|s]] + \mathbb{E}_s[\mathbb{V}_u[X|s]]$, it is sufficient to show

$$\mathbb{E}_u[Y|s] = \mathbb{E}_u[X|s], \quad (2)$$
$$\mathbb{V}_u[Y|s] \le \mathbb{V}_u[X|s] \quad (3)$$

for all s. For notational simplicity, $\mathbb{E}_u[\cdot|s]$ and $\mathbb{V}_u[\cdot|s]$ are written as $\mathbb{E}_u[\cdot]$ and $\mathbb{V}_u[\cdot]$ below, respectively.

The exact value of $Q^{\pi_\theta}(s,u)$ is usually not available and needs to be estimated. It is often estimated using observed rewards after executing u at s, sometimes combined with function approximation to balance bias and variance (Schulman et al., 2016; Mnih et al., 2016), but this is possible only for u that is executed at s. Our algorithm assumes the estimates of $Q^{\pi_\theta}(s,u)$ only for such (s,u) pairs to be available, and thus is applicable to such cases.

[1] When θ is not a scalar, we consider the variance of gradients with respect to each element of θ throughout the paper.

3. Clipped Action Policy Gradient

We consider the case where any action $u\in\mathbb{R}^d$ ($d\ge1$) chosen by an agent is clipped by the environment into a range $[\alpha,\beta]\subset\mathbb{R}^d$. That is, the state-transition PDF and the reward function satisfy

$$P(s'|s,u) = P(s'|s,\mathrm{clip}(u,\alpha,\beta)), \quad (4)$$
$$r(s,u) = r(s,\mathrm{clip}(u,\alpha,\beta)), \quad (5)$$

respectively. The clip function is defined as $\mathrm{clip}(u,\alpha,\beta) = \max(\min(u,\beta),\alpha)$, where $\max$ and $\min$ are computed elementwise when u is a vector, i.e., $d\ge2$. Each of α and β can be a constant or a function of s. The case where the reward function depends on actions before clipping is discussed in Section 3.4.

Before explaining our algorithm, let us characterize the class of policies we consider in this study.

Definition 3.1 (compatible PDF). *Let $p_\theta(u)$ be a PDF of $u\in\mathbb{R}$ that has a parameter θ. If $p_\theta(u)$ is differentiable with respect to θ and allows the exchange of derivative and integral as $\int_{-\infty}^{\alpha}\nabla_\theta p_\theta(u)du = \nabla_\theta\int_{-\infty}^{\alpha}p_\theta(u)du$ and $\int_{\beta}^{\infty}\nabla_\theta p_\theta(u)du = \nabla_\theta\int_{\beta}^{\infty}p_\theta(u)du$, we call $p_\theta(u)$ a compatible PDF. If $p_\theta(u|s)$ is a conditional PDF that satisfies these conditions, we call it a compatible conditional PDF.*

3.1. Scalar actions

First, we derive an unbiased and lower-variance estimator of the policy gradient for scalar actions, i.e., $d=1$. The case of vector actions will be covered later in Section 3.2.

From (4) and (5), the state-action value function satisfies

$$\begin{aligned} Q^{\pi_\theta}(s,u) &= Q^{\pi_\theta}(s,\mathrm{clip}(u,\alpha,\beta)) \\ &= \begin{cases} Q^{\pi_\theta}(s,\alpha) & \text{if } u\le\alpha \\ Q^{\pi_\theta}(s,u) & \text{if } \alpha<u<\beta \\ Q^{\pi_\theta}(s,\beta) & \text{if } \beta\le u \end{cases}. \end{aligned} \quad (6)$$

Let X be a random variable that depends on u and $\mathbb{1}_{f(u)}$ be an indicator function that takes 1 when u satisfies the condition $f(u)$, otherwise 0. Because $X = \mathbb{1}_{u\le\alpha}X + \mathbb{1}_{\alpha<u<\beta}X + \mathbb{1}_{\beta\le u}X$, $\mathbb{E}_u[X]$ can be decomposed as

$$\mathbb{E}_u[X] = \mathbb{E}_u[\mathbb{1}_{u\le\alpha}X] + \mathbb{E}_u[\mathbb{1}_{\alpha<u<\beta}X] + \mathbb{E}_u[\mathbb{1}_{\beta\le u}X]. \quad (7)$$

From (6) and (7), we have

$$\begin{aligned} &\mathbb{E}_u[Q^{\pi_\theta}(s,u)\psi(s,u)] \\ &\quad = Q^{\pi_\theta}(s,\alpha)\mathbb{E}_u[\mathbb{1}_{u\le\alpha}\nabla_\theta\log\pi_\theta(u|s)] \\ &\qquad + \mathbb{E}_u[\mathbb{1}_{\alpha<u<\beta}Q^{\pi_\theta}(s,u)\nabla_\theta\log\pi_\theta(u|s)] \\ &\qquad + Q^{\pi_\theta}(s,\beta)\mathbb{E}_u[\mathbb{1}_{\beta\le u}\nabla_\theta\log\pi_\theta(u|s)]. \end{aligned} \quad (8)$$

Meanwhile, the following useful lemma holds.

Lemma 3.1. *Suppose $\pi_\theta(u|s)$ is a compatible conditional PDF of $u \in \mathbb{R}$ whose cumulative distribution function (CDF) is $\Pi_\theta(u|s)$. Then, the following equations hold:*

$$\mathbb{E}_u[\mathbb{1}_{u\le\alpha}\nabla_\theta \log \pi_\theta(u|s)] = \mathbb{E}_u[\mathbb{1}_{u\le\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)],$$
$$\mathbb{E}_u[\mathbb{1}_{\beta\le u}\nabla_\theta \log \pi_\theta(u|s)] = \mathbb{E}_u[\mathbb{1}_{\beta\le u}\nabla_\theta \log(1-\Pi_\theta(\beta|s))].$$

See the appendix for the proof.

By applying Lemma 3.1 to (8), we can construct an alternative estimator:

$$\begin{aligned}
&\mathbb{E}_u[Q^{\pi_\theta}(s,u)\psi(s,u)] \\
&\quad= Q^{\pi_\theta}(s,\alpha)\mathbb{E}_u[\mathbb{1}_{u\le\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)] \\
&\qquad+ \mathbb{E}_u[\mathbb{1}_{\alpha<u<\beta}Q^{\pi_\theta}(s,u)\nabla_\theta \log \pi_\theta(u|s)] \\
&\qquad+ Q^{\pi_\theta}(s,\beta)\mathbb{E}_u[\mathbb{1}_{\beta\le u}\nabla_\theta \log(1-\Pi_\theta(\beta|s))] \\
&\quad= \mathbb{E}_u[Q^{\pi_\theta}(s,u)\overline{\psi}(s,u)], \qquad (9)
\end{aligned}$$

where

$$\overline{\psi}(s,u) = \begin{cases} \nabla_\theta \log \Pi_\theta(\alpha|s) & \text{if } u \le \alpha \\ \nabla_\theta \log \pi_\theta(u|s) & \text{if } \alpha < u < \beta \\ \nabla_\theta \log(1-\Pi_\theta(\beta|s)) & \text{if } \beta \le u \end{cases}. \qquad (10)$$

By (9) the policy gradient can be estimated using the sample average of $Q^{\pi_\theta}(s,u)\overline{\psi}(s,u)$. This estimator, which we call clipped action policy gradient (CAPG), is better than the conventional estimator (1) in the sense that it has a lower variance while being unbiased.

The difference between the conventional estimator and CAPG comes from outside the action bounds. CAPG replaces $\pi_\theta(u|s)$ of $\nabla_\theta \log \pi_\theta(u|s)$ with $\Pi_\theta(\alpha|s)$ and $1 - \Pi_\theta(\beta|s)$ at $u \le \alpha$ and $\beta \le u$, respectively. Intuitively speaking, because both $\Pi_\theta(\alpha|s)$ and $1-\Pi_\theta(\beta|s)$ are deterministic given s, the variance should decrease. In fact, this observation is true.

To show this, we need to decompose the variance. The variance of a random variable X that depends on u can be decomposed as

$$\begin{aligned}
\mathbb{V}_u[X] = {}& \mathbb{V}_u[\mathbb{1}_{u\le\alpha}X] + \mathbb{V}_u[\mathbb{1}_{\alpha<u<\beta}X] + \mathbb{V}_u[\mathbb{1}_{\beta\le u}X] \\
&- 2\mathbb{E}_u[\mathbb{1}_{u\le\alpha}X]\mathbb{E}_u[\mathbb{1}_{\alpha<u<\beta}X] \\
&- 2\mathbb{E}_u[\mathbb{1}_{\alpha<u<\beta}X]\mathbb{E}_u[\mathbb{1}_{\beta\le u}X] \\
&- 2\mathbb{E}_u[\mathbb{1}_{\beta\le u}X]\mathbb{E}_u[\mathbb{1}_{u\le\alpha}X].
\end{aligned}$$

Let us compare each term of the right-hand side between the cases $X = Q^{\pi_\theta}(s,u)\psi(s,u)$ and $X = Q^{\pi_\theta}(s,u)\overline{\psi}(s,u)$. From Lemma 3.1, we can see that the terms $\mathbb{V}_u[\mathbb{1}_{\alpha<u<\beta}X]$, $\mathbb{E}_u[\mathbb{1}_{u\le\alpha}X], \mathbb{E}_u[\mathbb{1}_{\alpha<u<\beta}X]$, and $\mathbb{E}_u[\mathbb{1}_{\beta\le u}X]$ do not make any differences. The following lemma shows that the difference arises from the terms $\mathbb{V}_u[\mathbb{1}_{u\le\alpha}X]$ and $\mathbb{V}_u[\mathbb{1}_{\beta\le u}X]$.

Lemma 3.2. *Suppose $\pi_\theta(u|s)$ is a compatible conditional PDF of $u \in \mathbb{R}$ whose CDF is $\Pi_\theta(u|s)$. Then, the following inequalities hold:*

$$\mathbb{V}_u[\mathbb{1}_{u\le\alpha}\nabla_\theta \log \pi_\theta(u|s)] \ge \mathbb{V}_u[\mathbb{1}_{u\le\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)],$$
$$\mathbb{V}_u[\mathbb{1}_{\beta\le u}\nabla_\theta \log \pi_\theta(u|s)] \ge \mathbb{V}_u[\mathbb{1}_{\beta\le u}\nabla_\theta \log(1-\Pi_\theta(\beta|s))].$$

The equalities hold only when $\nabla_\theta \log \pi_\theta(u|s)$ is constant over $u \le \alpha$ and $\beta \le u$, respectively.

See the appendix for the proof.

Combining Lemma 3.1 and Lemma 3.2 leads to the following result.

Lemma 3.3. *Suppose $\pi_\theta(u|s)$ is a compatible conditional PDF of $u \in \mathbb{R}$ whose CDF is $\Pi_\theta(u|s)$. Let $f(s,u)$ be a real-valued function such that*

$$f(s,u) = \begin{cases} f(s,\alpha) & \textit{if } u \le \alpha \\ f(s,u) & \textit{if } \alpha < u < \beta \\ f(s,\beta) & \textit{if } \beta \le u \end{cases}.$$

Define $\psi(s,u) = \nabla_\theta \log \pi_\theta(u|s)$ and $\overline{\psi}(s,u)$ as (10). Then, the following equality and inequality hold:

$$\mathbb{E}_u[f(s,u)\overline{\psi}(s,u)] = \mathbb{E}_u[f(s,u)\psi(s,u)],$$
$$\mathbb{V}_u[f(s,u)\overline{\psi}(s,u)] \le \mathbb{V}_u[f(s,u)\psi(s,u)].$$

The equality of the variances holds only when $\psi(s,u)$ is constant over both $u \le \alpha$ and $\beta \le u$.

Lemma 3.3 shows that both (2) and (3) are satisfied when $Y = Q^{\pi_\theta}(s,u)\overline{\psi}(s,u)$ and $X = Q^{\pi_\theta}(s,u)\psi(s,u)$. Therefore, we can conclude that CAPG has a lower variance than the conventional estimator while being unbiased.

3.2. Vector actions

The results in the previous subsection can be extended to the case of vector actions, $\mathbf{u} \in \mathbb{R}^d$ where $d \ge 2$, as long as the elements of $\mathbf{u}$ are conditionally independent given s, i.e., the PDF can be factored as

$$\pi(\mathbf{u}|s) = \pi_\theta^{(1)}(u_1|s)\pi_\theta^{(2)}(u_2|s)\cdots\pi_\theta^{(d)}(u_d|s),$$

where u_i denotes the i-th element of $\mathbf{u}$, and $\pi_\theta^{(i)}$ denotes its corresponding conditional PDF. A typical example of such a policy is a multivariate Gaussian policy with a diagonal covariance.

Lemma 3.4. *Suppose $\pi_\theta(\mathbf{u}|s)$ is a conditional PDF of $\mathbf{u} \in \mathbb{R}^d$ ($d \ge 2$) whose CDF is $\Pi_\theta(\mathbf{u}|s)$. The conditional PDF and CDF of u_i are denoted by $\pi_\theta^{(i)}$ and $\Pi_\theta^{(i)}$, respectively. Suppose each $\pi_\theta^{(i)}$ is compatible and the elements of $\mathbf{u}$ are conditionally independent given s. Let*

$f(s, \mathbf{u})$ be a real-valued function such that $f(s, \mathbf{u}) = f(s, \mathrm{clip}(\mathbf{u}, \alpha, \beta))$. Define $\psi(s, \mathbf{u}) = \sum_i \psi^{(i)}(s, u_i)$, where $\psi^{(i)}(s, u) = \nabla_\theta \log \pi_\theta^{(i)}(u|s)$. Similarly, define $\overline{\psi}(s, \mathbf{u}) = \sum_i \overline{\psi}^{(i)}(s, u_i)$, where

$$\overline{\psi}^{(i)}(s, u) = \begin{cases} \nabla_\theta \log \Pi_\theta^{(i)}(\alpha|s) & \textit{if } u \leq \alpha \\ \nabla_\theta \log \pi_\theta^{(i)}(u|s) & \textit{if } \alpha < u < \beta \\ \nabla_\theta \log(1 - \Pi_\theta^{(i)}(\beta|s)) & \textit{if } \beta \leq u \end{cases}.$$

Then, the following equality and inequality hold:

$$\mathbb{E}_{\mathbf{u}}[f(s, \mathbf{u})\overline{\psi}(s, \mathbf{u})] = \mathbb{E}_{\mathbf{u}}[f(s, \mathbf{u})\psi(s, \mathbf{u})], \tag{11}$$

$$\mathbb{V}_{\mathbf{u}}[f(s, \mathbf{u})\overline{\psi}(s, \mathbf{u})] \leq \mathbb{V}_{\mathbf{u}}[f(s, \mathbf{u})\psi(s, \mathbf{u})]. \tag{12}$$

The equality of the variances holds only when $\psi^{(i)}(s, u)$ is constant over both $u \leq \alpha$ and $\beta \leq u$ for all $1 \leq i \leq d$.

See the appendix for the proof.

3.3. Implementation

CAPG can be easily incorporated into existing policy gradient-based algorithms. We only have to replace the computation of $\psi(s, u)$ with that of $\overline{\psi}(s, u)$ to use CAPG. When $\psi(s, u)$ is computed using an automatic differentiation tool, we can instead replace $\log \pi_\theta(u|s)$ with

$$\begin{cases} \log \Pi_\theta(\alpha|s) & \text{if } u \leq \alpha \\ \log \pi_\theta(u|s) & \text{if } \alpha < u < \beta \\ \log(1 - \Pi_\theta(\beta|s)) & \text{if } \beta \leq u \end{cases}.$$

3.4. Extensions

Although we have used standard notations of MDPs, our results do not rely on the Markov property. CAPG works as an unbiased and lower-variance policy gradient estimator in non-Markovian environments as well, in the same way that the REINFORCE algorithm (Williams, 1992) works in such environments.

We assumed (5) so that $Q^{\pi_\theta}(s, u)$ becomes constant outside the action bounds. However, sometimes it makes sense to use a reward function that depends on out-of-bound actions even when the state-transition dynamics does not, e.g., to penalize the norm of actions to prevent the policy from going too far out of the bounds. With such a reward function, (6) no longer holds. Instead, we can use the recursive structure of $Q^{\pi_\theta}(s, u)$ to obtain

$$\begin{aligned} &\mathbb{E}_u[Q^{\pi_\theta}(s, u)\psi(s, u)] \\ &\quad= \mathbb{E}_u[r(s, u)\psi(s, u)] \\ &\qquad+ \mathbb{E}_u[\gamma \mathbb{E}_{s', u'}[Q^{\pi_\theta}(s', u')]\psi(s, u)], \end{aligned} \tag{13}$$

where $\mathbb{E}_{s', u'}[\cdot]$ denotes an expected value with respect to $s' \sim P(\cdot|s, \mathrm{clip}(u, \alpha, \beta))$, $u' \sim \pi_\theta(\cdot|s')$. We can apply CAPG to the second term of the right-hand side of (13) because $\gamma \mathbb{E}_{s', u'}[Q^{\pi_\theta}(s', u')]$ only depends on u via $\mathrm{clip}(\cdot, \alpha, \beta)$.

3.5. Clipped distribution

So far we have derived CAPG as a better policy gradient estimator. We now argue that CAPG can be interpreted as estimating the policy gradient of a transformed policy.

Given a policy π_θ and action bounds $[\alpha, \beta]$, we can consider a policy $\overline{\pi}_\theta$ modeled as a probability distribution with bounded support whose CDF is defined as $\overline{\Pi}_\theta(u|s) = \mathbb{1}_{\alpha \leq u < \beta}\Pi_\theta(u|s) + \mathbb{1}_{\beta \leq u}$, which is a mixture of two degenerate distributions at $\{\alpha, \beta\}$ and a truncated version of π_θ. The corresponding PDF with respect to the measure generated by the mixture [2] is given by

$$\overline{\pi}_\theta(u|s) = \begin{cases} \Pi_\theta(\alpha|s) & \text{if } u = \alpha \\ \pi_\theta(u|s) & \text{if } \alpha < u < \beta \\ 1 - \Pi_\theta(\beta|s) & \text{if } u = \beta \end{cases}.$$

We call this distribution a clipped distribution. Seeing that $\nabla_\theta \log \overline{\pi}_\theta(u|s) = \overline{\psi}(s, u)$ for $u \in [\alpha, \beta]$, CAPG applied to π_θ is, in fact, estimating the policy gradient of $\overline{\pi}_\theta$. If we see Gaussian policies used with action bounds as clipped Gaussian policies, then CAPG is the straightforward policy gradient estimator for them, whereas the conventional estimator has an unnecessarily high variance.

While a clipped distribution resembles a truncated distribution, they are different. A clipped distribution can be multimodal even when its underlying distribution is unimodal because it puts the probability mass at the action bounds. In contrast, a truncated distribution is always unimodal when its underlying distribution is unimodal. This makes a difference in their representational powers to model policies.

4. Experiments

In this section, we evaluate the performance of CAPG compared to the conventional policy gradient estimator, which we call PG, in problems with action bounds.

4.1. Continuum-armed bandit problems

To demonstrate how CAPG works and how it interacts with each aspect of problems separately, we used continuum-armed bandit problems (Agrawal, 1995), i.e., MDPs with continuous action spaces and no state transitions. State-independent policies were optimized by policy gradients to maximize action-dependent immediate rewards.

[2] The probability measure P corresponding to $\overline{\Pi}_\theta(u|s)$, defined over the measurable space $([\alpha, \beta], \mathcal{B}([\alpha, \beta]))$, is such that $P \ll \lambda + \delta_\alpha + \delta_\beta$, where $\mathcal{B}$ is the Borel σ-algebra, λ is the Lebesgue measure, and δ_x is a Dirac measure at x.

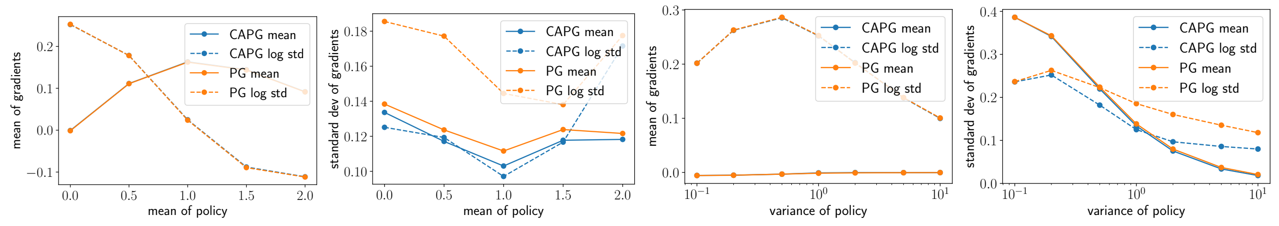

Figure 1. Means and standard deviations of policy gradient estimates obtained using CAPG and PG on a continuum-armed bandit problem with a fixed policy of varying means (left half) and variances (right half). For each data point, policy gradients with respect to θ_μ and θ_Σ are estimated 10,000 times using 10,000 different batches of 5 (action, reward) pairs. The CAPG and PG plots of the means of gradients almost overlap each other, and hence, only the PG plots are visible.

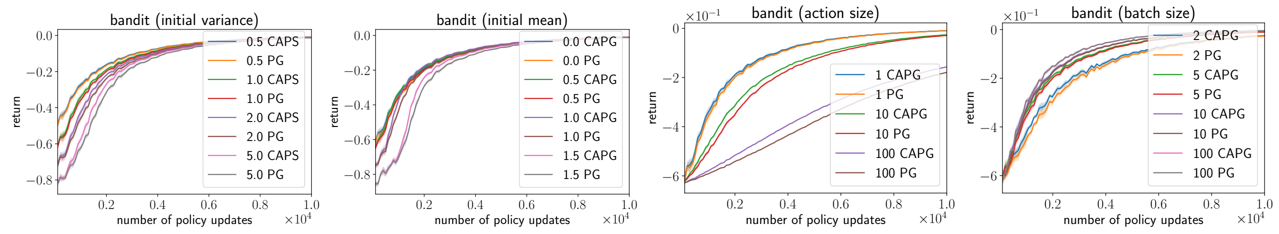

Figure 2. Training curves on continuum-armed bandit problems with four different aspects separately controlled: (from left to right) variance of the initial policy, mean of the initial policy, number of dimensions of actions, and batch size. For each run, the last reward before every policy update is sampled and then averaged over the previous 100 updates to obtain a smoothed curve. The smoothed curves are then averaged to compute the mean curves with 68% and 95% bootstrapped confidence intervals, which are indicated by the shaded areas.

The action space was $[-1, 1]^d, d \geq 1$ and the reward function was defined as $r(u) = -\frac{1}{d}\sum_i |u_i|$ so that only choosing the optimal action of zeros achieves the maximum, zero reward.

Each policy was modeled as a multivariate Gaussian distribution with a diagonal covariance matrix and parameterized by $\theta = \{\theta_\mu, \theta_\Sigma\}$, where $\theta_\mu \in \mathbb{R}^d$ is the mean vector and $\theta_\Sigma \in \mathbb{R}^d$ is the main diagonal of the covariance matrix.

The following experimental settings were used unless otherwise stated. Actions were scalars, i.e., $d = 1$. The parameters of a policy were initialized as zero mean and unit variance for each dimension. Each policy update used a batch of 5 (action, reward) pairs. The average reward in a batch was used as a baseline that was subtracted from each reward. Adam (Kingma & Ba, 2015) with its default hyperparameters was used to update the parameters.

To quantify the variance reduction achieved by CAPG, we repeatedly estimated policy gradients using new samples without updating a policy. Figure 1 shows the mean and standard deviation of policy gradient estimates obtained by CAPG and PG with a fixed policy of varying means and variances. For both θ_μ and θ_Σ in all settings, CAPG consistently achieved lower variance than PG without introducing visible bias. These results numerically corroborate CAPG's variance reduction ability as well as its unbiasedness. The efficacy of CAPG diminished at $\sigma^2 = 0.1$, where sampled actions rarely go outside the bounds.

Figure 2 shows the training curves of CAPG and PG with four different aspects separately controlled: variance of the initial policy, mean of the initial policy, number of dimensions of actions, and batch size. Each configuration is evaluated with 10 different random seeds. CAPG consistently achieved faster learning across the settings. A larger initial variance and a more distant initial mean tend to make the gap more visible. CAPG's gain scales even for 100 dimensions, implying its utility for more challenging, complex continuous control tasks. Using smaller batch sizes benefits more from CAPG, and this is expected because smaller batch sizes are more affected by the variance of gradient estimation. With the batch size of 100, the training curve of CAPG is difficult to distinguish from that of PG. It should be noted that in these experiments all the actions are sampled from the same state. In practical model-free RL scenarios, more than one action cannot be sampled from the same state.

4.2. Simulated control problems

To evaluate CAPG's effectiveness in more practical settings, we used the following two popular deep RL algorithms for continuous control:

- Proximal policy optimization (PPO) with clipped sur-

	Obs. space	Action space
InvertedPendulum-v1	$\mathbb{R}^4$	$[-3.0, 3.0]^1$
InvertedDoublePendulum-v1	$\mathbb{R}^{11}$	$[-1.0, 1.0]^1$
Reacher-v1	$\mathbb{R}^{11}$	$[-1.0, 1.0]^2$
Hopper-v1	$\mathbb{R}^{11}$	$[-1.0, 1.0]^3$
HalfCheetah-v1	$\mathbb{R}^{17}$	$[-1.0, 1.0]^6$
Swimmer-v1	$\mathbb{R}^8$	$[-1.0, 1.0]^2$
Walker2d-v1	$\mathbb{R}^{17}$	$[-1.0, 1.0]^6$
Ant-v1	$\mathbb{R}^{111}$	$[-1.0, 1.0]^8$
Humanoid-v1	$\mathbb{R}^{376}$	$[-0.4, 0.4]^{17}$
HumanoidStandup-v1	$\mathbb{R}^{376}$	$[-0.4, 0.4]^{17}$

Table 1. MuJoCo-simulated environments used in the experiments and their observation and action spaces.

rogate objective (Schulman et al., 2017)

- Trust region policy optimization (TRPO) (Schulman et al., 2015) with generalized advantage estimation (GAE) (Schulman et al., 2016).

For each of the two algorithms, we implemented the variant that uses CAPG as well as the original one that uses PG. The only difference between these two is whether CAPG or PG is used.

For our experiments, we used 10 MuJoCo-simulated environments implemented in OpenAI Gym that are widely used as benchmark tasks for deep RL algorithms (Schulman et al., 2017; Henderson et al., 2018; Ciosek & Whiteson, 2018; Gu et al., 2017b; Duan et al., 2016; Dhariwal et al., 2017). The names of the environments are listed along with their observation and action spaces in Table 1. All the environments have bounded action spaces; hence, actions are clipped before being sent to the environments.

We considered all the combinations of $\{\text{PPO, TRPO}\} \times \{\text{CAPG, PG}\} \times 10$ environments, each of which is trained for 1 million timesteps. Each combination is tried 50 times with different random seeds. Because we found it difficult to obtain reasonable performance within 1 million timesteps on Ant-v1, Humanoid-v1, and HumanoidStandup-v1, we also tried training for 10 million timesteps on these environments.

We followed the hyperparameter settings used in (Henderson et al., 2018), except that the learning rate of Adam used by PPO was reduced to 3e-5 for 10 million timesteps training to obtain reasonable performance with PG. We used separate neural networks with two hidden layers, each of which has 64 hidden units with tanh nonlinearities, for both a policy and a state value function. The policy network outputs the mean of a multivariate Gaussian distribution. The main diagonal of the covariance matrix was separately parameterized as a logarithm of the standard deviation for each dimension.

Table 2 summarizes the comparison between CAPG and PG, combined with TRPO and PPO. We used areas under the learning curves (AUCs) as evaluation measures because they can measure not only the final performance but also the learning speed and stability.

For PPO and TRPO with 1 million training timesteps, CAPG significantly ($p < 0.025$, i.e., $> 95\%$ significance) improved AUCs on 3 and 7 out of the 10 environments, respectively. It also significantly helped in training for 10 million timesteps on two out of the three harder environments for both PPO and TRPO. On other environments, it kept almost the same level of AUCs on other tasks, although there seemed to be slight decreases in some environments. These results indicate that CAPG can safely replace PG in many cases.

Figures 3 and 4 show the smoothed learning curves of all the experiments. In some cases, the improvements were small but consistent, e.g., TRPO on InvertedDoublePendulum-v1 and TRPO on HumanoidStandup-v1 (10 million). In some other cases, large improvements were achieved, e.g., PPO on Swimmer-v1 and TRPO on Humanoid-v1 (10 million).

Although we used the same hyperparameters from (Henderson et al., 2018) for both PG and CAPG, the best hyperparameters for CAPG can be different. It is possible that separate hyperparameter tuning can further improve the performance of CAPG.

Comparing the results of PPO and TRPO, PPO was more affected than TRPO by the difference in estimators, suggesting that PPO is more vulnerable to high variance in gradient estimation. TRPO is likely to be more robust against variance for the following reasons.

- TRPO uses a large batch of 5000 actions for every policy update. PPO uses minibatches of 64 actions, resulting in noisier updates.
- TRPO solves a constrained optimization problem for every policy update so that the change in KL divergence is close to a constant; thus, it is robust to changes in the scale of gradients. PPO also adapts its step size using Adam, but this adaptation is slower and based on the statistics of accumulated past gradients.

Because we observe that even TRPO can benefit from CAPG, we expect the benefits address other algorithms with noisier updates as well.

5. Related Work

A variety of techniques has been proposed to reduce the variance of policy gradient estimation since its introduction. The control variate method, namely subtracting some baseline from approximate returns, is widely used to reduce the

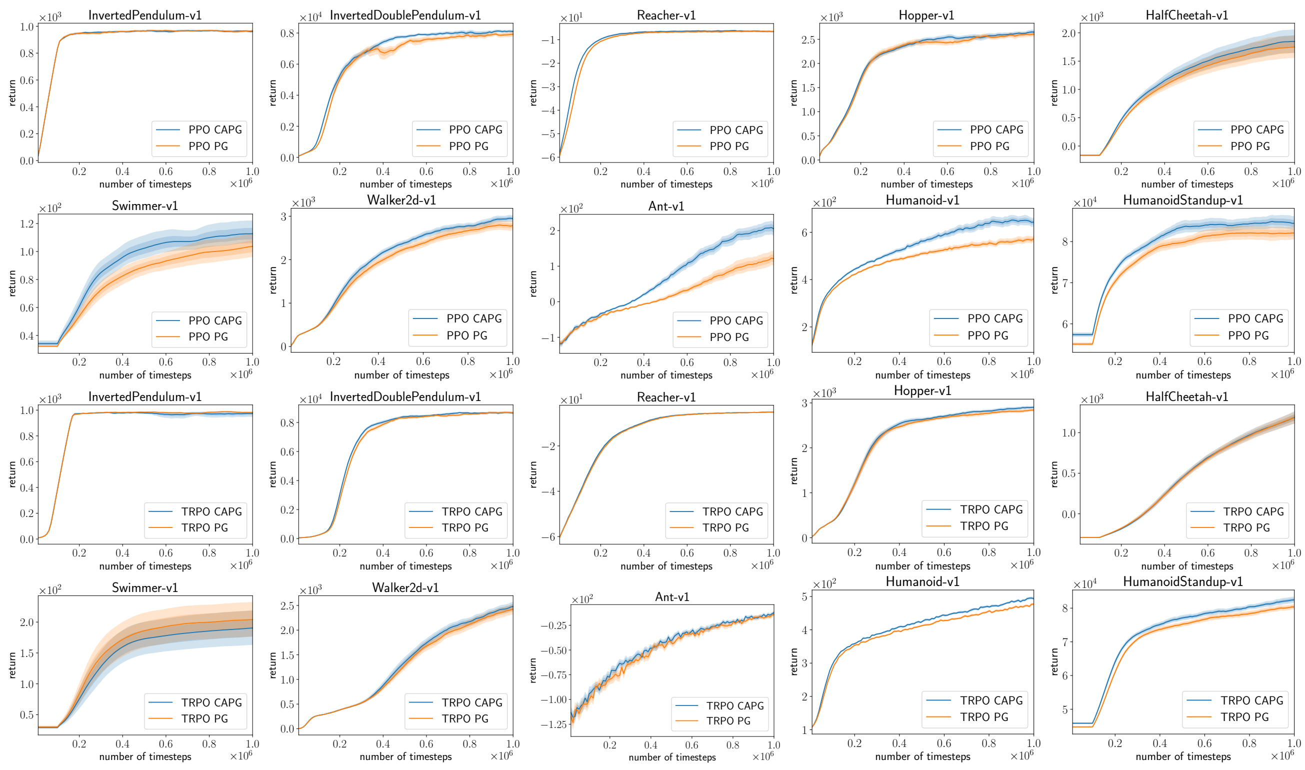

Figure 3. Training curves of PPO (upper half) and TRPO (lower half) on the 10 MuJoCo-simulated environments. For each run, after every training episode, the average return of the previous 100 training episodes is computed and linearly interpolated between the episodes to obtain a smoothed curve. The smoothed curves are then averaged to compute the mean curves with 68% and 95% bootstrapped confidence intervals, which are indicated by the shaded areas.

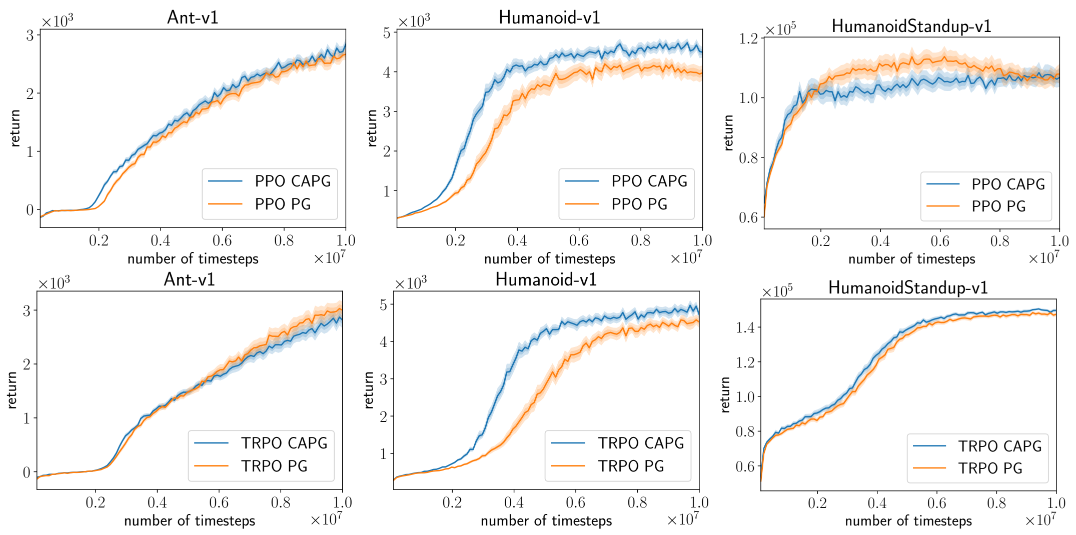

Figure 4. Training curves of PPO (upper half) and TRPO (lower half) on the three harder MuJoCo-simulated environments. For each run, after every training episode, the average return of the previous 100 training episodes is computed and linearly interpolated between the episodes to obtain a smoothed curve. The smoothed curves are then averaged to compute the mean curves with 68% and 95% bootstrapped confidence intervals, which are indicated by the shaded areas.

	PPO CAPG	PPO PG	p-value	TRPO CAPG	TRPO PG	p-value
InvertedPendulum-v1	955.30$\pm$1.12	955.68$\pm$0.84	7.88e-01	915.08$\pm$5.23	919.94$\pm$0.79	3.63e-01
InvertedDoublePendulum-v1	**7239.24$\pm$23.01**	6991.40$\pm$43.01	2.67e-06	**7108.54$\pm$18.17**	7007.32$\pm$18.95	2.07e-04
Reacher-v1	**-10.67$\pm$0.15**	-11.60$\pm$0.17	8.71e-05	-14.66$\pm$0.13	-14.93$\pm$0.13	1.41e-01
Hopper-v1	2320.49$\pm$11.49	2288.50$\pm$17.91	1.37e-01	2313.33$\pm$16.14	2283.55$\pm$16.03	1.94e-01
HalfCheetah-v1	1219.54$\pm$60.94	1144.53$\pm$58.81	3.78e-01	502.05$\pm$18.36	499.99$\pm$18.57	9.37e-01
Swimmer-v1	**92.56$\pm$3.48**	82.45$\pm$2.75	2.49e-02	148.86$\pm$11.44	161.18$\pm$11.92	4.58e-01
Walker2d-v1	**2185.63$\pm$26.23**	2060.95$\pm$38.92	9.41e-03	1436.38$\pm$30.31	1390.69$\pm$27.60	2.68e-01
Ant-v1	**56.85$\pm$5.19**	-33.32$\pm$7.26	2.01e-16	**-204.68$\pm$1.84**	-212.15$\pm$1.92	6.04e-03
Humanoid-v1	**547.64$\pm$5.90**	493.39$\pm$3.89	2.49e-11	**415.88$\pm$0.79**	402.19$\pm$0.75	3.96e-22
HumanoidStandup-v1	**79414.10$\pm$496.59**	76845.37$\pm$512.67	5.03e-04	**73592.94$\pm$292.50**	71796.93$\pm$265.64	1.58e-05
Ant-v1 (10m)	**1579.54$\pm$10.64**	1476.51$\pm$15.21	2.98e-07	1395.50$\pm$28.44	1449.61$\pm$32.05	2.10e-01
Humanoid-v1 (10m)	**3650.00$\pm$33.98**	3107.34$\pm$59.01	1.06e-11	**3353.08$\pm$23.57**	2743.53$\pm$40.29	1.99e-21
HumanoidStandup-v1 (10m)	101826.33$\pm$1012.21	105289.56$\pm$1173.48	2.78e-02	**123777.22$\pm$383.77**	120994.09$\pm$403.91	2.57e-06

Table 2. Performance comparison of CAPG and PG on the 10 MuJoCo-simulated environments. Performance is evaluated with the average area under the learning curve (AUC) $\pm$ standard error over 1 million timesteps. For each training run, its AUC is computed by linearly interpolating returns between training episodes. For each combination of $\{\text{TRPO, PPO}\} \times \{\text{CAPG, PG}\} \times 10$ environments, from 50 training runs with different random seeds, the average AUC and standard error are computed. p-values are also computed between CAPG and PG versions using Welch's t-test. Bold numbers indicate that they are better than their counterparts by 95% significance.

variance while avoiding the introduction of bias into the estimation (Williams, 1992; Sutton et al., 1999; Greensmith et al., 2004; Gu et al., 2017a;b). Relying on predicted values instead of sampled returns is also popular despite the bias it often introduces (Degris et al., 2012; Mnih et al., 2016; Schulman et al., 2016; Ciosek & Whiteson, 2018). Our approach reduces the variance differently from these two common approaches. Therefore, it can be easily combined with the existing techniques to reduce the variance further while not introducing additional bias.

The problem of using probability distributions with unbounded support for control problems with bounded action spaces was pointed out in (Chou et al., 2017), which proposed modeling policies as beta distributions as a solution. While they reported performance improvements by using beta policies across multiple continuous control environments, Gaussian policies still nearly dominate the deep RL literature (Dhariwal et al., 2017; Henderson et al., 2018; Tassa et al., 2018). Truncated distributions have also been used to deal with bounded action spaces in prior work (Nakano et al., 2012; Shariff & Dick, 2013; Zimmer et al., 2016). In contrast, our approach allows us to keep using the same policy parameterizations, typically Gaussians, and still exploit action bounds. It is also possible to see CAPG as using a multimodal distribution with bounded support, whereas beta policies and truncated Gaussian policies are unimodal. For example, a clipped Gaussian policy can easily learn to choose end-values of the action bounds with a high probability by moving its mean toward the corresponding end, while beta and truncated Gaussian policies need to be near-deterministic to choose near-end values with a high probability.

Exploiting the integral form of policy gradients to reduce the variance has been proposed in (Ciosek & Whiteson, 2018; Asadi et al., 2017). They directly evaluated the integral over the whole action space, which can be analytically computed for limited classes of action value approximators and policies. Their method can reduce the variance by eliminating the need for Monte-Carlo estimation of policy gradients while introducing bias from action value approximation. Our method only evaluates the integral outside the action bounds, i.e., where action values are constant, and thus is unbiased.

6. Discussion

We have shown that the variance of policy gradient estimation can be reduced by exploiting the fact that actions are clipped before they are sent to the environment. An unbiased and lower-variance policy gradient estimator, named CAPG, has been proposed based on our analysis. CAPG is easy to implement and can be combined with existing variance reduction techniques, such as control variates and value function approximations.

We numerically analyzed CAPG's behavior on simple continuum-armed bandit problems, confirming its efficacy in variance reduction. When incorporated into existing deep RL algorithms, CAPG generally achieved the same or better performance on challenging simulated control benchmark tasks, indicating its promise as an alternative to the conventional estimator.

While a Gaussian policy is the most common choice in policy gradient-based continuous control, distributions with bounded support may be more suitable for bounded action spaces. Prior work has proposed beta and truncated distributions to explore this direction. We argued that CAPG can also be seen as estimating the policy gradient of a transformed distribution with bounded support, termed a clipped distribution. Further studies are needed on the behaviors of different kinds of distributions as policy representations.

Acknowledgments

We thank Toshiki Kataoka, Kenta Oono, Masaki Watanabe and others at Preferred Networks for insightful comments and discussions.

References

Agrawal, R. The Continuum-Armed Bandit Problem. *SIAM Journal on Control and Optimization*, 33(6):1926–1951, 1995.

Asadi, K., Allen, C., Roderick, M., Mohamed, A.-r., Konidaris, G., and Littman, M. Mean Actor Critic. *ArXiv e-prints*, 2017.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *ArXiv e-prints*, 2016.

Chou, P.-W., Maturana, D., and Scherer, S. Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution. In *ICML*, 2017.

Ciosek, K. and Whiteson, S. Expected Policy Gradients. In *AAAI*, 2018.

Degris, T., White, M., and Sutton, R. S. Off-Policy Actor-Critic. In *ICML*, 2012.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. OpenAI Baselines. https://github.com/openai/baselines, 2017.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking Deep Reinforcement Learning for Continuous Control. In *ICML*, 2016.

Greensmith, E., Bartlett, P., and Baxter, J. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *The Journal of Machine Learning Research*, 5: 1471–1530, 2004.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-Prop: Sample-Efficient Policy Gradient with an Off-Policy Critic. In *ICLR*, 2017a.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., Schölkopf, B., and Levine, S. Interpolated Policy Gradient : Merging On-Policy and Off-Policy Gradient Estimation for Deep. In *NIPS*, 2017b.

Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M., and Silver, D. Emergence of Locomotion Behaviours in Rich Environments. *ArXiv e-prints*, 2017.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep Reinforcement Learning that Matters. In *AAAI*, 2018.

Kingma, D. P. and Ba, J. L. Adam: a Method for Stochastic Optimization. In *ICLR*, 2015.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-End Training of Deep Visuomotor Policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. a., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*, 2016.

Nakano, D., Maeda, S.-i., and Ishii, S. Control of a Free-Falling Cat by Policy-Based Reinforcement Learning. In *ICANN*, 2012.

Schulman, J., Levine, S., Moritz, P., Jordan, M., and Abbeel, P. Trust Region Policy Optimization. In *ICML*, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *ICLR*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *ArXiv e-prints*, 2017.

Shariff, R. and Dick, T. Lunar Lander : A Continous-Action Case Study for Policy-Gradient Actor-Critic Algorithms. In *RLDM*, 2013.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Driessche, G. V. D., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., and Kavukcuoglu, K. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7585):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., and Sifre, L. Mastering the game of Go without human knowledge. *Nature Publishing Group*, 550(7676):354–359, 2017.

Sutton, R. S., Mcallester, D., Singh, S., and Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*, 1999.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., De, D., Casas, L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind Control Suite. *ArXiv e-prints*, 2018.

Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *IROS*, 2012.

Williams, R. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992.

Zimmer, M., Boniface, Y., and Dutech, A. Off-policy Neural Fitted Actor-Critic. In *NIPS Deep Reinforcement Learning Workshop*, 2016.