Rank-1 Approximation of Inverse Fisher for Natural Policy Gradients in Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Natural gradients have long been studied in deep reinforcement learning due to their fast convergence properties and covariant weight updates. However, computing natural gradients requires inversion of the Fisher Information Matrix (FIM) at each iteration, which is computationally prohibitive in nature. In this paper, we present an efficient and scalable natural policy optimization technique that leverages a rank-1 approximation to full inverse-FIM. We theoretically show that under certain conditions, a rank-1 approximation to inverse-FIM converges faster than policy gradients and, under some conditions, enjoys the same sample complexity as stochastic policy gradient methods. We benchmark our method on a diverse set of environments and show that it achieves superior performance to standard actor-critic and trust-region baselines.

1 Introduction

Policy gradient methods (Sutton et al., 1999) have emerged as one of the most prominent candidates for modern reinforcement learning (RL). Specifically, they have seen widespread adoption in deep RL, where they incorporate differentiable policy parametrization and function approximators to help policy gradient methods scale up and learn robust controllers for high-dimensional state spaces (Arulkumaran et al., 2017; Wang et al., 2022). Despite this, a fundamental limitation of policy gradients is that RL objectives tend to be highly non-concave in nature. Consequently, algorithms based on standard gradient descent, such as REINFORCE (Williams, 1992), are prone to getting stuck at sub-optimal local minima when training high-dimensional policies. This is due to the gradient giving only the direction in which a policy's parameters should be optimized, without indicating the size of the optimization step (Martens, 2020). Step size selection is, therefore, non-trivial. Small steps lead to stable but slow and sample-inefficient training. On the contrary, large steps can cause unstable updates. This is particularly dangerous in RL, as unstable policies collect poor data, making it difficult for the agent to recover and escape local optima (van Heeswijk, 2022).

One approach to reduce this sub-optimality and achieve stable training with large steps is to incorporate the geometry of the loss manifold into account and thus consider natural gradient descent methods (Amari, 1998). Specifically, when applied to policy gradient RL, where policies are modeled as probability distributions, the Fisher information matrix (FIM) is used to precondition the policy gradients, which gives rise to natural policy gradient (NPG) methods (Kakade, 2001; Bagnell & Schneider, 2003; Peters et al., 2005). NPG algorithms are of interest because they generally tend to converge faster and have superior performance compared to their standard policy gradient counterparts (Grondman et al., 2012). More recently, NPGs established the roots of some of the most popular RL algorithms used in practice, including Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a) and Proximal Policy Optimization (PPO) (Schulman et al., 2017). Furthermore, the study of NPG methods is also motivated by their success in real-world applications (Richter et al., 2006; Su et al., 2019).

One of the main drawbacks of NPG algorithms is the high cost of computing and inverting the FIM. Storage is also an important concern, as for a parameterized policy, the FIM is a square matrix with dimensions equal to the number of parameters. Naturally, this issue becomes especially limiting when policies are parameterized by deep neural networks. In practice, NPG algorithms usually avoid computing the inverse Fisher directly and instead rely on approximations. A simple option is to only consider the diagonal elements of the FIM (Becker & Lecun, 1989; LeCun et al., 2002), leading to efficient inversion and low storage requirements. A more sophisticated approach is K-FAC (Martens & Grosse, 2015), which uses Kronecker products to approximate the FIM efficiently. As an alternative to the true FIM, the empirical FIM can be computed from samples, which is especially convenient when the required underlying probability distributions are unknown (Schraudolph, 2002). Hessian-free methods give a less direct approach to using the FIM without ever explicitly computing or storing it (Martens, 2010). As an example, TRPO avoids computing the FIM, calculating the Fisher-vector product directly instead. Despite the extensive literature on the subject, choosing the right way of approximating natural gradients is usually not straightforward. This choice often involves a trade-off between computational efficiency and approximation accuracy.

The goal of this work is to bridge the theoretical convergence properties of NPG with a practical, fast, and computationally feasible approach to invert FIM for high-dimensional reinforcement learning tasks. In this work, we use the empirical FIM as an alternative to the FIM and incorporate the Sherman-Morrison formula for efficient inversions. We propose a rank-1 approximation to avoid explicit matrix operations, keeping the time complexity at $\mathcal{O}(d)$ without the need for inner loops like conjugate gradient, thus balancing computational cost and accuracy of curvature information.

The main contributions are listed below:

1. We provide a fast and efficient way to compute the inverse FIM, which can be computed in $\mathcal{O}(d)$ complexity rather than $\mathcal{O}(d^3)$ for naive inversion of FIM, d denotes the dimension of the policy parameter vector θ.

2. We theoretically show that under certain conditions, our NPG approximation using the empirical FIM and Sherman-Morrison formula enjoys global convergence when using log-linear function approximators.

3. We also prove that the error induced in the policy update due to this rank-1 approximation is bounded and reduces with the increase in iterations.

4. We propose a novel policy gradient algorithm using this update: SM-ActorCritic (**SMAC**). We show that SMAC achieves faster convergence in diverse OpenAI Gym tasks (classic control and MuJoCo environments), compared to vanilla policy gradient, Adam, and trust-region methods that use Conjugate Gradient solvers to approximate the direction of natural policy gradient.

2 Related Work

Gradient descent updates all the parameters of a model equally, meaning that weights are considered to be inside a Euclidean space. However, the loss function of the model actually induces a Riemannian manifold space. Therefore, the size of a gradient step does not reflect the true scale of change of the model's loss. Consequently, standard gradient descent methods are highly sensitive to the choice of step size. To avoid this issue, Amari first framed gradient-based learning on a Riemannian manifold, introducing the natural gradient descent algorithm, which uses the update direction $F(\theta)^{-1}\nabla J(\theta)$, with the FIM $F(\theta)$ Amari (1998). Here, the FIM is a covariant measure of the amount of information contained in the parameters Amari (1998). Kakade extended this idea to reinforcement learning and coined the natural policy gradient (NPG) algorithm Kakade (2001). However, the computational cost associated with calculating the FIM, especially in models with a large number of parameters, can be prohibitively expensive. Therefore, practical implementations of natural gradients rely on approximations instead. We thus summarize the related work on these approximations as follows.

Kronecker-factored Approximate Curvature Block-diagonal approximations to the natural gradient, such as K-FAC (Martens & Grosse, 2015) and subsequent refinements (Zhang et al., 2023; Ren & Goldfarb, 2021; Bahamou et al., 2023; Benzing, 2022), have been widely adopted due to their computational efficiency. Related variants have also been developed for reinforcement learning (Wu et al., 2017).

Hessian-Free Methods Hessian-free optimization Martens (2010), another method for computing natural gradients, has seen widespread use in reinforcement learning, particularly due to its superior performance in TRPO Schulman et al. (2015a). This approach employs the conjugate gradient (CG) method to solve the associated linear system, thereby avoiding explicitly storing FIM, keeping memory usage comparable to a single forward–backward pass. However, CG requires several additional forward–backward passes, which reduces computational efficiency. In addition, due to the limited precision of CG estimation, practitioners usually perform an extra KL-based line search to adapt the step size (Schulman et al., 2015a).

Empirical Fisher Information The empirical Fisher (EF) estimates the Fisher information matrix by replacing the exact expectation over gradient outer products with the outer product of the gradients computed from individual samples Yang et al. (2022). However, the empirical Fisher's simple estimate has been frequently criticized for deviating significantly from the true Fisher, and its limitations have been analyzed in many studies Martens (2020); Kunstner et al. (2019). Nevertheless, its very low computational cost has enabled a broad range of successful applications Roux et al. (2007); Ren & Goldfarb (2019); Wu et al. (2024). Compared with the extensive use of K-FAC and Hessian-free methods in RL, the EF approach remains under-explored and is seldom applied in this domain.

3 Preliminaries

Markov decision process (MDP) We consider an episodic MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with a continuous or discrete state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $P(s'|s,a)$, bounded reward function $r : S \times A \mapsto [-R, R]$, and discount factor $\gamma \in (0,1)$. We also consider a stochastic policy $\pi_\theta(a|s)$ with parameters $\theta \in \mathbb{R}^d$ that induces a trajectory distribution $\pi_\theta(\tau_i)$ over $\tau_i = (s_0^i, a_0^i, s_1^i, a_1^i, \dots)$.

Performance objective and policy gradient Under the standard actor-critic framework, the RL objective is given by the expected discounted return of π_θ:

$$J(\theta) \;=\; \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t\, r(s_t,a_t)\right]. \tag{1}$$

To maximize Eq. 1, the advantage actor-critic (A2C) (Mnih et al., 2016) algorithm uses an advantage function $A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$, defined through the critic's state-value and action-value functions

$$V^{\pi_\theta}(s) = \mathbb{E}_{\tau\sim\pi_\theta}\left[G_t \mid s_t = s\right], \qquad Q^{\pi_\theta}(s,a) = \mathbb{E}_{\tau\sim\pi_\theta}\left[G_t \mid s_t = s, a_t = a\right],$$

where $G_t = \sum_{l=0}^{\infty}\gamma^l r(s_{t+l}, a_{t+l})$. Following the policy gradient theorem, the A2C gradient is then given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{\infty}\nabla_\theta \log\pi_\theta(a_t|s_t)\; A^{\pi_\theta}(s_t,a_t)\Big].$$

Fisher information matrix and Natural Policy Gradient For the exponential-family policy $\pi_\theta(\cdot \mid s)$, the natural Riemannian metric in parameter space is the Fisher information matrix

$$\boldsymbol{F}_s(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\big[\nabla_\theta \log\pi_\theta(a|s)\nabla_\theta\log\pi_\theta(a|s)^\top\big]. \tag{2}$$

For a preconditioning matrix $\boldsymbol{F}_\rho(\theta)$, this leads to the natural gradient update

$$\begin{aligned}\theta^{k+1} &= \theta^k + \eta \boldsymbol{F}_\rho(\theta^k)^{-1}\nabla_\theta J(\theta^k),\\ \boldsymbol{F}_\rho(\theta^k) &= \mathbb{E}_{s\sim d_\rho^{\pi_\theta}}[\boldsymbol{F}_s(\theta)],\end{aligned}$$

where $d_\rho^{\pi_\theta}$ represents the state visitation distribution under a policy π_θ and initial state distribution ρ:

$$d_\rho^{\pi_\theta}(s) := (1-\gamma)\,\mathbb{E}_{s_0\sim\rho}\sum_{t=0}^{\infty}\gamma^t\,\mathbb{P}(s_t = s \mid s_0, \pi_\theta) + \gamma\rho(s).$$

Empirical Fisher Information Replacing the expectation in $\boldsymbol{F}(\theta)$ with a sample average over a batch $\{\tau_i\}_{i=1}^N$ gives the empirical Fisher approximation

$$\hat{\boldsymbol{F}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top.$$

Sherman-Morrison formula The Sherman–Morrison formula efficiently computes the inverse of a matrix after a "rank-1 update" has been applied to it, provided that the inverse of the original matrix was already known Sherman & Morrison (1950). Suppose $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ is an **invertible square matrix** and $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ are **column vectors**. Then $\boldsymbol{X} + \boldsymbol{u}\boldsymbol{v}^\top$ is invertible **iff**

$$1 + \boldsymbol{v}^\top \boldsymbol{X}^{-1}\boldsymbol{u} \neq 0.$$

In this case, the Sherman-Morrison formula allows the computation of the inverse

$$(\boldsymbol{X} + \boldsymbol{u}\boldsymbol{v}^\top)^{-1} = \boldsymbol{X}^{-1} - \frac{\boldsymbol{X}^{-1}\boldsymbol{u}\boldsymbol{v}^\top\boldsymbol{X}^{-1}}{1 + \boldsymbol{v}^\top\boldsymbol{X}^{-1}\boldsymbol{u}}. \tag{3}$$

If $\boldsymbol{X}^{-1}$ is already known, this matrix inversion formula can bypass computing the inverse $(\boldsymbol{X} + \boldsymbol{u}\boldsymbol{v}^T)^{-1}$ directly, avoiding the cubic scaling with the dimension of $\boldsymbol{X}$. Instead, the update only requires three vector operations, which scale linearly with the dimension of $\boldsymbol{X}$. This update can use the full power of current GPU hardware platforms, since computing vector operations can be heavily parallelized. At the same time, it avoids the time-intensive matrix inversion processes. Specifically, methods like Singular Value Decomposition (SVD), which decomposes a matrix into its singular values and vectors, and LU decomposition, which factors a matrix into a lower triangular matrix (L) and an upper triangular matrix (U), are much slower than the simple operations used in our approach.

4 NPG Approximation

We adopt a matrix-free scheme for approximating the FIM to reduce memory and computational cost. At each step, we form a local empirical Fisher from the current batch, add a damping term λ, and apply the Sherman–Morrison formula to compute the inverse–vector product $\boldsymbol{F}(\theta)^{-1}g^k$ directly. This approach requires $\mathcal{O}(d)$ memory.

This formulation captures the local curvature of the loss landscape using only the current gradient direction. The damping term λ ensures numerical stability and prevents the update direction from being overly sensitive to individual gradients. Compared to the exact FIM, this method strikes a balance between computational efficiency and curvature-informed updates.

4.1 Matrix-Free Update

The Sherman-Morrison formula can be used to incrementally build the inverse of the EF $(\hat{\boldsymbol{F}}^k)^{-1}$ at step k by reusing the previous estimate $(\hat{\boldsymbol{F}}^{k-1})^{-1}$ (Singh & Alistarh, 2020):

$$\begin{aligned}(\hat{\boldsymbol{F}}^k)^{-1}g^k &= (\hat{\boldsymbol{F}}^{k-1} + \nabla_\theta \log \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)^\top)^{-1}g^k \\ &= (\hat{\boldsymbol{F}}^{k-1})^{-1}g^k - \frac{(\hat{\boldsymbol{F}}^{k-1})^{-1}\nabla_\theta \log \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)^\top(\hat{\boldsymbol{F}}^{k-1})^{-1}}{1 + \nabla_\theta \log \pi_\theta(a|s)^\top{(\hat{\boldsymbol{F}}^{k-1})}^{-1}\nabla_\theta \log \pi_\theta(a|s)}g^k.\end{aligned}$$

Other approaches compute a fresh EF at each iteration (Wu et al., 2024), which is used in the natural gradient

$$\Delta\theta = \eta(\lambda\boldsymbol{I} + \nabla_\theta \log \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)^\top)^{-1}\log \pi_\theta(a|s).$$

However, the time and space complexity of explicitly calculating and storing the FIM is $\mathcal{O}(d^2)$, which is infeasible for large-scale models. In this work, we instead use the Sherman–Morrison formula to directly compute the product of the inverse empirical Fisher and the policy gradient.

We consider a one-sample empirical Fisher with damping:

$$\hat{\boldsymbol{F}}^k = \lambda \boldsymbol{I} + \boldsymbol{F}^k(\theta) = \lambda \boldsymbol{I} + \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top$$

where $\lambda > 0$ is a fixed damping coefficient.

Applying the Sherman–Morrison formula yields:

$$\begin{aligned}
(\hat{\boldsymbol{F}}^k)^{-1} &= (\lambda \boldsymbol{I} + \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top)^{-1} \\
&= \frac{1}{\lambda}\boldsymbol{I} - \frac{\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top}{\lambda^2 + \lambda \nabla_\theta \log \pi_\theta(a|s)^\top \nabla_\theta \log \pi_\theta(a|s)} \\
&= \frac{1}{\lambda}\boldsymbol{I} - \frac{\boldsymbol{F}^k(\theta)}{\lambda^2 + \lambda Tr(\boldsymbol{F}^k(\theta))}. \qquad (4)
\end{aligned}$$

Multiplying the inverse Fisher with the gradient gives the natural policy gradient update direction:

$$\Delta\theta_k = \eta\,(\hat{\boldsymbol{F}}^k)^{-1} g^k = \eta \left[\frac{1}{\lambda} g^k - \frac{\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top g^k}{\lambda^2 + \lambda \nabla_\theta \log \pi_\theta(a|s)^\top \nabla_\theta \log \pi_\theta(a|s)} \right] \qquad (5)$$

where $g^k = \nabla_\theta \log \pi_\theta(a|s) A(s,a)$ is the one-sample policy gradient update direction, $A(s,a)$ is the advantage estimate and η is the step size.

This update involves only a matrix-vector operation and a vector outer product, making its time and memory complexity comparable to that of first-order methods, such as stochastic policy gradient.

4.2 Convergence Analysis

We take inspiration from the global convergence guarantees provided by Agarwal et al. (2021) and a follow-up work by Liu et al. (2020), which improved the global convergence results. To prove the global convergence, we take a standard Lipschitz assumption and take some previously proved bounds of the norm of exact policy gradient and the policy gradient defined till horizon H from Xu et al. (2020). We provide a theoretical analysis showing that the Sherman–Morrison rank-1 approximation to the inverse Fisher achieves global convergence. Under strong convexity assumptions (already taken in Kakade (2001), Peters et al. (2005)) of the FIM, we show that the average performance between the Sherman-Morrison update and the optimal policy π^* can be bounded. We defer the proof to the appendix, and state the final theorem here.

Theorem 1. *If we take the stochastic Sherman-Morrison policy update in Equation 4 and take* $\eta = \frac{1}{4L_J}$, $K = \mathcal{O}\left(\frac{1}{(1-\gamma)^2\varepsilon^2}\right)$, $N = \mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2}\right)$, *and* $H = \mathcal{O}\left(\log(\frac{1}{(1-\gamma)\varepsilon})\right)$. *Then, we have*

$$J(\pi^\star) - \frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[J(\theta^k)] \le \frac{\sqrt{\varepsilon_{bias}}}{1-\gamma} + \varepsilon.$$

In total, stochastic PG samples $\mathcal{O}\left(\frac{\sigma^2}{(1-\gamma)^2\varepsilon^4}\right)$ *trajectories, where* ε_{bias} *is the approximation error from the advantage function, and* ε *is an arbitrary precision level.*

Our proof relies on the fact that performance bound given by Lemma 4 Liu et al. (2020) and we prove that this result holds with a sample complexity of $KN = \mathcal{O}(\frac{\sigma^2}{(1-\gamma)^2\varepsilon^4})$, where N is the number of trajectories per update and K is the number of updates. This shows that to get an ε accurate policy, one needs $\mathcal{O}(\frac{1}{\varepsilon^2})$ samples, and when the horizon H and number of samples N are large enough, then σ can be reduced to arbitrary precision. Detailed derivations can be found in Appendix 1.

4.3 SM-ActorCritic

We plug our approximation into a standard A2C algorithm. Let π_θ be the policy (actor) and V_ϕ the value network (critic), with parameters θ and ϕ, respectively. At each iteration, we collect T transition tuples and

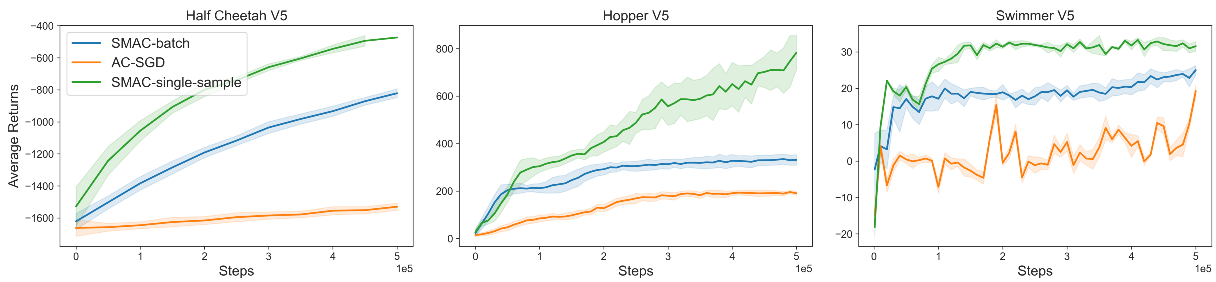

Figure 1: Effect of Batch Size.

compute the generalized advantage estimator (Schulman et al., 2015b, GAE) with a fixed $\lambda_{\text{GAE}} \in [0, 1]$:

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \tag{6}$$

$$A_t = \sum_{l=0}^{T-1-t} (\gamma \lambda_{\text{GAE}})^l \, \delta_{t+l}. \tag{7}$$

The critic is then trained by minimizing the mean squared error over the batch:

$$\mathcal{L}_{\text{critic}}(\phi) = \frac{1}{T} \sum_{t=0}^{T-1} \big(R_t - V_\phi(s_t)\big)^2,$$

using the Adam optimizer, where $R_t = A_t + V_\phi(s_t)$ is the Monte-Carlo return target. The actor is updated by a single natural policy gradient step obtained from the matrix-free Fisher inverse. The full SMAC update is given in Algorithm 1.

Algorithm 1 SM-ActorCritic

```
input: steps per update T, step size η, damping λ
initialise θ, φ
for iteration k = 0, 1, ... do
    rollout T steps, store (s^k, a^k, r^k)
    compute advantages A_k with GAE
    for all samples k do
        g^k ← ∇_θ log π_θ(a | s)
        θ^{k+1} ← θ^k + η (F̂^k)^{-1} g^k with equation 5
    end for
    update critic: φ^{k+1} ← φ^k − α∇_φ Σ_t (R^k − V_φ)^2
end for
```

5 Experimental Setup

To evaluate the performance of our proposed approach, SMAC, we provide experiments in several environments and compare against three baselines. We begin with three simple control environments, before moving on to six MuJoCo tasks (Todorov et al., 2012). For all experiments, we used the Gymnasium framework (Towers et al., 2024).

We implement SMAC on top of the A2C algorithm. In all environments, we evaluate SMAC against three baselines. To measure the benefits of using our NPG approximator, we consider an algorithm that is similar to SMAC in all aspects, except that it uses first-order updates instead of natural gradients. This is equivalent

	AC-SGD	AC-Adam	AC-CG	SMAC (Ours)
Acrobot	$\mathbf{-86.2 \pm 5.0}$	-88.4 ± 3.8	-93.3 ± 20.3	-94.1 ± 6.3
Cartpole	962.3 ± 49.0	$\mathbf{989.5 \pm 0.3}$	977.9 ± 0.0	973.4 ± 29.8
Pendulum	-2116 ± 564	-5731 ± 150	$\mathbf{-190 \pm 18}$	-2226 ± 859
Half Cheetah	-888.7 ± 15.4	-746.3 ± 24.6	-444.6 ± 12.1	$\mathbf{863.8 \pm 38.7}$
Hopper	180.5 ± 13.6	303.8 ± 27.1	172.7 ± 16.3	$\mathbf{331.2 \pm 29.7}$
Swimmer	12.4 ± 5.2	17.3 ± 0.5	23.8 ± 0.9	$\mathbf{28.5 \pm 0.8}$
Walker	213.4 ± 42.9	79.3 ± 95.0	$\mathbf{229.8 \pm 23.9}$	204.1 ± 87.9
Humanoid	4539 ± 110	3105 ± 299	3175 ± 1423	$\mathbf{4625 \pm 277}$
Pusher	-433.6 ± 54.9	-568.3 ± 12.8	-441.0 ± 34.5	$\mathbf{-408.2 \pm 32.2}$

Table 1: Average returns per episode, measured at the end of training, for each task in both classic control and MuJoCo environments. All results are averaged over 5 seeds.

to A2C with a simple stochastic gradient descent (SGD) optimizer, which we denote as AC-SGD. To ensure a fair comparison to vanilla A2C, we also compare against the AC-Adam baseline, where A2C is trained using the more complex Adam optimizer. Finally, we also test SMAC against a different NPG approximator. The AC-CG baseline uses the conjugate gradient algorithm to directly compute the NPG during the A2C update, without ever computing or storing the Fisher information matrix itself, similarly to TRPO (Schulman et al., 2015a). To better handle continuous control problems, SMAC and all the baselines we compare against compute advantages using the GAE (Schulman et al., 2015b).

For each of the four algorithms in each environment, we train five models using different random seeds. The results reported throughout training are averaged over these seeds, with standard deviations included. We evaluate agent performance using (undiscounted) returns, averaged across episodes. Besides the maximum average returns achieved, we are also interested in convergence speed. Furthermore, we discuss the average log-probabilities of the actions taken as a measure of the agents' confidence in the actions they take. We report these in detail in the supplementary material.

In all environments, episodes have a horizon of 1000 timesteps. To increase the readability of the plots, we group timesteps into 100 bins with no overlap and report the average of each bin. We apply additional smoothing by computing an exponentially weighted mean over the bins, with a smoothing factor of 0.1.

5.1 Ablation: Effect of Batch Size

The matrix-free updates operate on a single trajectory sample. Although the per-step cost is only $\mathcal{O}(d)$, many more updates are required to process a fixed number of transitions, so the computational cost can be very high.

To investigate this trade-off, we evaluate an alternative that uses the average gradient of log-probability:

$$\bar{\ell} = \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log \pi_\theta(a_i \mid s_i).$$

The regularized empirical Fisher then becomes

$$\hat{F}^k = \lambda \boldsymbol{I} + \bar{\ell}^k \, (\bar{\ell}^k)^\top,$$

and the update direction of SMAC is given by

$$\Delta\theta^k = \eta \left[\frac{1}{\lambda} g^k - \frac{\bar{\ell}^k \, (\bar{\ell}^k)^\top g^k}{\lambda^2 + \lambda (\bar{\ell}^k)^\top \bar{\ell}^k} \right].$$

Results. Fig. 1 compares the single-sample and batch-mean variants on Mujoco HalfCheetah and Hopper. Using $B = 1000$ reduces the overall optimization time by roughly 80% while achieving returns that remain

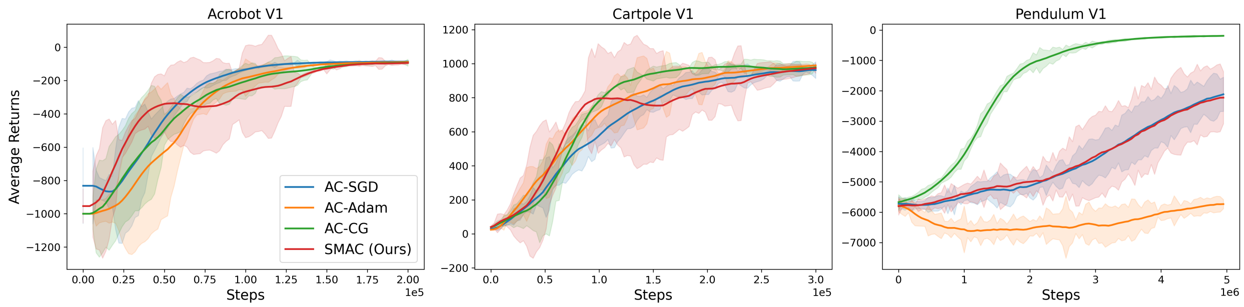

Figure 2: Results on three classic control tasks. We report the average return per episode. All results are averaged over 5 random seeds, with shaded areas showing standard deviation.

close to the single-sample baseline and clearly outperform the standard A2C optimized with SGD. Therefore, to further balance computational resources and performance, we implement the version with a batch size of 1000 in all subsequent experiments.

6 Results

We provide the final performance achieved by training each of the four algorithms on each of the nine tasks. These results, presented in Table 1, show that our method, SMAC, can achieve competitive results in two of the three simple control tasks. More importantly, the benefits of SMAC appear to increase in more complex environments, such as MuJoCo, where SMAC outperforms the other baselines in terms of final returns in all but one task. A more in-depth analysis of these results, with a special focus on convergence rate, is given in the following sections.

6.1 Classic Control

In the classic control environment, our method, SMAC, reaches a high initial average return faster than the baselines on two of the tasks, as shown in Fig. 2. This, however, comes at the cost of additional variance across seeds. The overall performance of SMAC is also competitive in those two environments.

In *Acrobot*, SMAC only requires 40000 timesteps to pass the average return threshold of -400. This is faster than AC-Adam, AC-SGD, and AC-CG, which require 70%, 35%, and 50% more timesteps, respectively. This accelerated learning is, however, followed by unstable training until approximately 125000 timesteps. Despite this, SMAC stabilizes and converges, reaching a final performance of -94.1 ± 6.3, which is similar to the baselines.

A similar behavior can be observed in *Cartpole*. SMAC agents achieve 75% of the maximum return possible in 87000 timesteps, slightly faster than AC-CG, which needs approximately 10% more timesteps. AC-Adam and AC-SGD are even slower, requiring 28% and 62% more timesteps, respectively, to reach the same threshold. SMAC becomes more unstable after this initial period, but stabilizes, reaching a competitive performance of 973.4 ± 29.8 by the end of training.

In contrast, in the last environment, *Pendulum*, our method fails to outperform AC-CG in terms of convergence speed and overall performance. AC-CG requires only 1.5 million timesteps to match the performance of SMAC, indicating that our proposed approximation of the FIM can struggle to improve learning in some environments. However, SMAC still achieves a final return of -2226 ± 859, outperforming AC-Adam and matching AC-SGD, with the former also displaying much slower and unstable learning.

The validity of our results is also confirmed by the action log-probabilities recorded. In *Acrobot* and *Cartpole*, the two tasks in which SMAC performs competitively, the log-probabilities converge faster than those of the baseline methods. Together with the increase in performance achieved by SMAC in these tasks, this

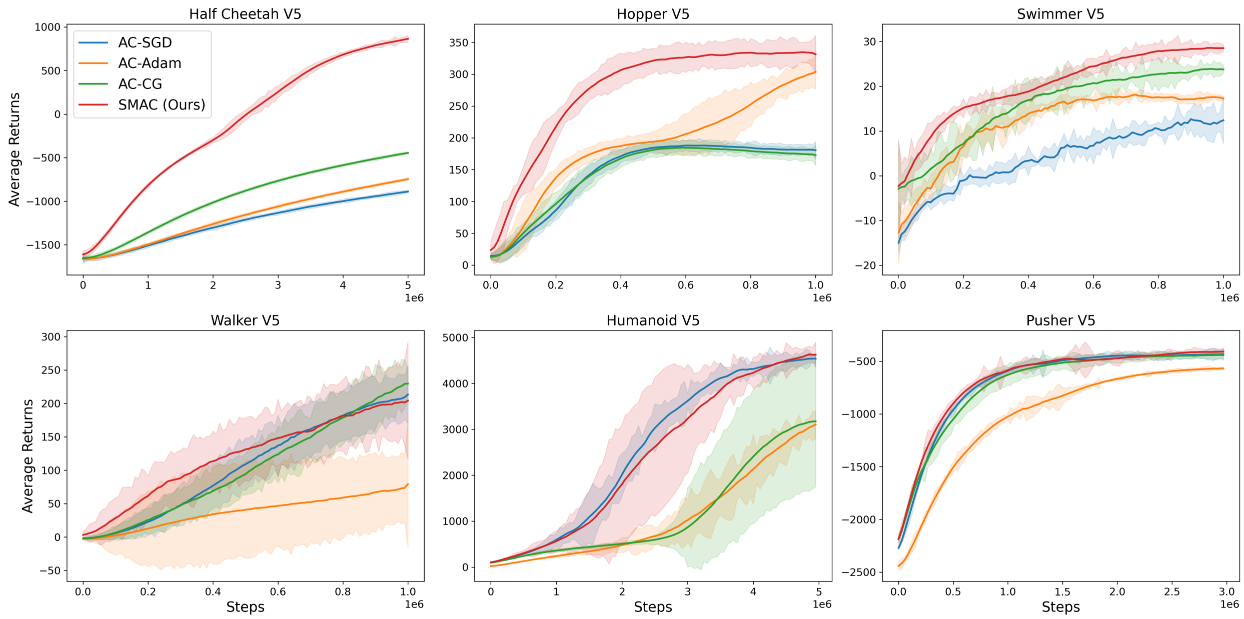

Figure 3: Results on six MuJoCo tasks. We report the average return per episode. All results are averaged over 5 random seeds, with shaded areas showing standard deviation.

may indicate that our agents can rapidly become confident in their policies when taking actions that yield increasingly higher returns. At the same time, in *Pendulum*, the log-probabilities of SMAC match those of AC-SGD and are bounded above by AC-CG. This could indicate that our method becomes overconfident in its actions too early, finally leading to a sub-optimal policy.

6.2 MuJoCo

In the more complex MuJoCo environment, we have found that throughout training, our method, SMAC, consistently outperforms or matches the baseline algorithms in terms of performance in 4 out of 6 tasks (Fig. 3). In the other two, SMAC still achieves comparable results. Furthermore, SMAC exhibits a noticeably higher sample efficiency in three tasks.

The largest improvement is in *Half Cheetah*, where SMAC achieves an average return of 863.8 ± 38.7 by the end of training. The next-best algorithm, AC-CG, achieves only -444.6 ± 12.1, being surpassed by our method by over 1300 points. Importantly, SMAC also learns three times faster, matching the final performance of AC-CG after only 1.7 million timesteps. Additionally, note that while SMAC has a higher variance across seeds mid-training, this reduces by the end of learning.

Similar trends can be seen in *Hopper*, where SMAC reaches the final performance of AC-Adam by using almost three times fewer timesteps. Our method achieves a final average return of 331.2 ± 29.7, outperforming the second-best performance of AC-Adam with 303.8 ± 27.1. The stability of SMAC appears to be comparable to that of AC-Adam but lower than that of the other two baselines.

Likewise, in *Swimer*, SMAC consistently outperforms the second-best baseline, with a final average return of 28.5 ± 0.8, compared to the second-best return of 23.8 ± 0.9. Moreover, it learns the fastest, matching the peak performance of AC-CG almost twice as fast. The stability of SMAC is also greater than that of AC-SGD, and competitive with that of AC-Adam and AC-CG.

In *Walker*, our method learns faster than the baselines in the first 60% of the timesteps. While its final performance of 204.1 ± 87.9 is lower than those of AC-CG and AC-SGD, which achieve 229.8 ± 23.9 and 213.4 ± 42.9, respectively, SMAC reaches the threshold of 115 (half of the final return of AC-CG) in only

410000 timesteps, while AC-CG itself needs 39% more. Additionally, our approach outperforms AC-Adam in both average return and stability.

For the *Humanoid* environment, SMAC achieves the top final performance, 4625 ± 277 average return, followed closely by AC-SGD with 4539 ± 110. It greatly outperforms AC-Adam and AC-CG in terms of both final returns and final stability, with only AC-SGD being more stable. While AC-SGD is also slightly faster than our method, SMAC still learns much faster than the other two baselines, matching their final performance after using just 59% of the total allocated training budget.

In *Pusher*, the performance of SMAC closely follows that of AC-SGD, slightly outperforming the latter with a final average return of -408.2 ± 32.2. Our method is also more sample efficient, passing the -441 return threshold (final performance of AC-CG) 21% faster than AC-CG and 9% faster than AC-SGD. While all agents stabilize by the end of training, only AC-Adam ends up being more stable than SMAC, with AC-SGD having the highest standard deviation.

Finally, we turn our attention to the average log-probabilities of the actions taken during training. The increase in log-probabilities shown by SMAC is stable and correlates with the increase in average returns given in Fig. 3. This may indicate that the policy quickly becomes confident in the actions it chooses, without becoming overconfident. Note that the log-probabilities of our method converge the fastest in 4 out of 6 tasks. While the log-probabilities of AC-CG converge first in *Humanoid* and *Pusher*, this does not lead to better performance. On the contrary, AC-CG agents become overconfident in their sub-optimal actions in the former task. The slow convergence of the other two agents' log-probabilities may indicate uncertainty in their actions, even when their performance matches that of SMAC.

7 Conclusion

In this work, we introduced a simple yet effective rank-1 approximation to the Natural Policy Gradient in the Actor-Critic framework. By using a regularized Empirical Fisher matrix and the Sherman–Morrison formula, we enable scalable and efficient natural policy gradient updates with only $\mathcal{O}(d)$ complexity. We theoretically prove the convergence properties of this approximation and integrate it into the standard Actor-Critic algorithm to demonstrate its practical utility. Experimental results on classic control and MuJoCo tasks show that our method achieves both fast and stable convergence in most environments, outperforms methods such as Advantage Actor-Critic and conjugate gradient in TRPO in several environments, offering a promising alternative for efficient NPG approximations.

References

Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22 (98):1–76, 2021. URL http://jmlr.org/papers/v22/19-736.html.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746. URL https://doi.org/10.1162/089976698300017746.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE signal processing magazine*, 34(6):26–38, 2017.

J. Andrew Bagnell and Jeff Schneider. Covariant policy search. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pp. 1019–1024, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

Achraf Bahamou, Donald Goldfarb, and Yi Ren. A mini-block fisher method for deep neural networks. In Francisco J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent (eds.), *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pp. 9191–9220. PMLR, 2023. URL https://proceedings.mlr.press/v206/bahamou23a.html.

Suzanna Becker and Yann Lecun. Improving the convergence of back-propagation learning with second-order methods. In *Proceedings of the 1988 Connectionist Models Summer School*, 01 1989.

Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1817–1853. PMLR, 2022. URL `https://proceedings.mlr.press/v162/benzing22a.html`.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 42(6):1291–1307, 2012.

Sham M. Kakade. A natural policy gradient. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani (eds.), *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pp. 1531–1538. MIT Press, 2001. URL `https://proceedings.neurips.cc/paper/2001/hash/4b86abe48d358ecf194c56c69108433e-Abstract.html`.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4158–4169, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/46a558d97954d0692411c861cf78ef79-Abstract.html`.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 2002.

Yanli Liu, Kaiqing Zhang, Tamer Basar, and Wotao Yin. An improved analysis of (variance-reduced) policy gradient and natural policy gradient methods. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7624–7636. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/56577889b3c1cd083b6d7b32d32f99d5-Paper.pdf`.

James Martens. Deep learning via hessian-free optimization. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 735–742. Omnipress, 2010. URL `https://icml.cc/Conferences/2010/papers/458.pdf`.

James Martens. New insights and perspectives on the natural gradient method. *J. Mach. Learn. Res.*, 21:146:1–146:76, 2020. URL `https://jmlr.org/papers/v21/17-678.html`.

James Martens and Roger B. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2408–2417. JMLR.org, 2015. URL `http://proceedings.mlr.press/v37/martens15.html`.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In *European Conference on Machine Learning*, pp. 280–291. Springer, 2005.

Yi Ren and Donald Goldfarb. Efficient subsampled gauss-newton and natural gradient methods for training neural networks. *CoRR*, abs/1906.02353, 2019. URL `http://arxiv.org/abs/1906.02353`.

Yi Ren and Donald Goldfarb. Tensor normal training for deep learning models. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 26040–26052, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/dae3312c4c6c7000a37ecfb7b0aeb0e4-Abstract.html`.

Silvia Richter, Douglas Aberdeen, and Jin Yu. Natural actor-critic for road traffic optimisation. *Advances in neural information processing systems*, 19, 2006.

Nicolas Le Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. Topmoumoute online natural gradient algorithm. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis (eds.), *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pp. 849–856. Curran Associates, Inc., 2007. URL `https://proceedings.neurips.cc/paper/2007/hash/9f61408e3afb633e50cdf1b20de6f466-Abstract.html`.

Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015a. URL `http://proceedings.mlr.press/v37/schulman15.html`.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

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

Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Annals of Mathematical Statistics*, 21(1):124–127, 1950. doi: 10.1214/aoms/1177729893.

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/d1ff1ec86b62cd5f3903ff19c3a326b2-Abstract.html`.

Zhiqiang Su, Meng Zhou, Fangfang Han, Yiwu Zhu, Dalei Song, and Tingting Guo. Attitude control of underwater glider combined reinforcement learning with active disturbance rejection control. *Journal of Marine Science and Technology*, 24(3):686–704, 2019.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL `https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Wouter JA van Heeswijk. Natural policy gradients in reinforcement learning explained. *arXiv preprint arXiv:2209.01820*, 2022.

Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2022.

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

Xiaodong Wu, Wenyi Yu, Chao Zhang, and Philip C. Woodland. An improved empirical fisher approximation for natural gradient descent. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL `http://papers.nips.cc/paper_files/paper/2024/hash/f23098fa0cfcdef0e743b134d380eeb9-Abstract-Conference.html`.

Yuhuai Wu, Elman Mansimov, Roger B. Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5279–5288, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/361440528766bbaaaa1901845cf4152b-Abstract.html`.

Pan Xu, Felicia Gao, and Quanquan Gu. Sample efficient policy gradient methods with recursive variance reduction. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HJlxIJBFDr`.

Minghan Yang, Dong Xu, Zaiwen Wen, Mengyun Chen, and Pengxiang Xu. Sketch-based empirical natural gradient methods for deep learning. *J. Sci. Comput.*, 92(3):94, 2022. doi: 10.1007/S10915-022-01911-X. URL `https://doi.org/10.1007/s10915-022-01911-x`.

Lin Zhang, Shaohuai Shi, and Bo Li. Eva: Practical second-order optimization with kronecker-vectorized approximation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=_Mic8V96Voy`.

A Appendix

A.1 Assumptions

Assumption 1. *The Fisher Information matrix induced by the policy π_θ is Positive Definite and satisfies*

$$\boldsymbol{F}_\rho(\theta) = \mathbb{E}_{s\sim d_\rho^{\pi_\theta}} \mathbb{E}_{a\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta \cdot \nabla_\theta \log \pi_\theta^T\right] \succcurlyeq \mu_F \mathbf{I} \tag{8}$$

Assumption 2. *The policy gradient estimates are bounded by a constant $G \geq 0$ and the change in policy gradient at different timestep are bounded by the change in policy parametrization weights. So,*

$$\|\nabla_\theta \log \pi_\theta\| \leq G \tag{9}$$

$$\|\nabla_\theta \log \pi_{\theta_1} - \nabla_\theta \log \pi_{\theta_2}\| \leq M \|\theta_1 - \theta_2\| \tag{10}$$

Assumption 3. ***Transferrable Compatible Function Approximator**: When we approximate the Advantage function using the policy π_θ it induces a transfer error defined as ϵ_{bias} which is zero for softmax parametrization and is very small for dense neural policy class.*

$$L_{\nu^\star}(w_\star^\theta; \theta) = \mathbb{E}_{(s,a)\sim\nu^\star} \left[\left(A^{\pi_\theta}(s,a) - (1-\gamma)(\boldsymbol{w}_\star^\theta)^\top \nabla_\theta \log \pi_\theta(a|s)\right)^2\right] \leq \varepsilon_{bias} \tag{11}$$

where $\boldsymbol{w}_\star^\theta = \operatorname{argmin}_w L_{\nu_\rho^{\pi_\theta}}(\boldsymbol{w}; \theta)$ is obtained via the full natural policy gradient direction at the parametrization θ.

Remark 1. *Regarding $\boldsymbol{F}(\theta)$, we know from Assumptions 1 and 2 that*

$$\mu_F I_d \preccurlyeq \boldsymbol{F}(\theta) \preccurlyeq G^2 I_d \text{ for any } \theta \in \mathbf{R}^d.$$

Remark 2. *Now we can get an upper bound to the second term of Equation 4 using Remark 1 as:*

$$\left\|\frac{\hat{\boldsymbol{F}}^k}{\lambda^2 + \lambda Tr(\hat{\boldsymbol{F}}^k)}\right\| \leq \frac{G^2}{\lambda^2 + \lambda\mu_F}.$$

A.2 Helper Lemmas

We will establish some helper lemma to prove the global convergence of Sherman-Morrison policy update.

Lemma 1. *Liu et al. (2020) In the stochastic PG update, by choosing $\eta = \frac{1}{4L_J}$, $K = \frac{32L_J(J^{H,\star} - J^H(\theta_0))}{\varepsilon}$, and $N = \frac{6\sigma^2}{\varepsilon}$, we have*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[\|\nabla J^H(\theta^k)\|^2] \leq \varepsilon.$$

In total, stochastic PG samples $\mathcal{O}\left(\frac{\sigma^2}{(1-\gamma)^2\varepsilon^2}\right)$ trajectories.

Lemma 2. *Xu et al. (2020) Let $J(\theta), J^H(\theta)$ are L_J-smooth, where $L_J = \frac{MR}{(1-\gamma)^2} + \frac{2G^2R}{(1-\gamma)^3}$ and*

$$\|\nabla J^H(\theta) - \nabla J(\theta)\| \leq GR\left(\frac{H+1}{1-\gamma} + \frac{\gamma}{(1-\gamma)^2}\right)\gamma^H$$

Lemma 3. *Liu et al. (2020) Let $g^k = \frac{1}{N}\sum_{i=1}^N g(\tau_i^H|\theta^k)$ be the policy gradient estimate for samples until horizon H. Then, we have*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[\|\nabla J^H(\theta^k)\|^2] \leq \frac{\frac{J^{H\star} - J^H(\theta_0)}{K} + (\frac{\eta}{2} + L_J\eta^2)\frac{\sigma^2}{N}}{\frac{\eta}{2} - L_J\eta^2} \tag{12}$$

Lemma 4. *Liu et al. (2020) Let $\boldsymbol{w}_\star^k = F_\rho^{-1}(\theta^k)\nabla J(\theta^k)$ be the exact NPG update direction at θ^k. Then, we have*

$$\begin{aligned} J(\pi^\star) - \frac{1}{K}\sum_{k=0}^{K-1} J(\theta^k) \leq \frac{\sqrt{\varepsilon_{bias}}}{1-\gamma} + \frac{1}{\eta K}\mathbb{E}_{s\sim d_\rho^{\pi^\star}}\left[KL\left(\pi^\star(\cdot|s)||\pi_{\theta^0}(\cdot|s)\right)\right] \\ + \frac{G}{K}\sum_{k=0}^{K-1}\|\boldsymbol{w}^k - \boldsymbol{w}_\star^k\| + \frac{M\eta}{2K}\sum_{k=0}^{K-1}\|\boldsymbol{w}^k\|^2. \end{aligned} \tag{13}$$

A.3 Global Convergence of SM Update

Let us take $\boldsymbol{w}^k$ as the update direction of Sherman-Morrison. To this end, we need to upper bound $\frac{1}{K}\sum_{k=0}^{K-1}\|\boldsymbol{w}^k - \boldsymbol{w}_\star^k\|$, $\frac{1}{K}\sum_{k=0}^{K-1}\|\boldsymbol{w}^k\|^2$, and $\frac{1}{K}\mathbb{E}_{s\sim d_\rho^{\pi^\star}}\left[\mathrm{KL}\left(\pi^\star(\cdot|s)||\pi_{\theta^0}(\cdot|s)\right)\right]$, where $\boldsymbol{w}_\star^k = F^{-1}(\theta^k)\nabla J(\theta^k)$ is the exact NPG update direction at θ^k.

- Bounding $\frac{1}{K}\sum_{k=0}^{K-1}\|\boldsymbol{w}^k - \boldsymbol{w}_\star^k\|$.

  We know from Jensen's inequality and $\left(\mathbb{E}[\|\boldsymbol{w}_t^{j+1} - \boldsymbol{w}_{t,\star}^{j+1}\|]\right)^2 \leq \mathbb{E}[\|\boldsymbol{w}_t^{j+1} - \boldsymbol{w}_{t,\star}^{j+1}\|^2]$ that

$$\begin{aligned} &\left(\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\boldsymbol{w}^k - \boldsymbol{w}_\star^k\|]\right)^2 \\ &\leq \frac{1}{K}\sum_{k=0}^{K-1}\left(\mathbb{E}[\|\boldsymbol{w}^k - \boldsymbol{w}_\star^k\|]\right)^2 \\ &\leq \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\boldsymbol{w}^k - \boldsymbol{w}_\star^k\|^2] \\ &\leq \frac{2}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\boldsymbol{w}^k - \nabla J(\theta^k)\|^2] + \frac{2}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla J(\theta^k) - \boldsymbol{w}_\star^k] \end{aligned} \tag{14}$$

  Let

$$\boldsymbol{g}^k = \frac{1}{N}\sum_{i=1}^{N} g(\tau_i^H|\theta^k),$$

  be the PG direction and

$$\boldsymbol{w}^k = \frac{1}{\lambda}\boldsymbol{g}^k - \frac{\hat{\boldsymbol{F}}^k}{\lambda^2 + \lambda Tr(\hat{\boldsymbol{F}}^k)}\boldsymbol{g}^k$$

be the weight update from the Sherman-Morrison update then we have from Lemma 3 and Assumption 2 that

$$
\begin{aligned}
&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\boldsymbol{w}^k-\nabla J(\theta^k)\|^2]\\
&=\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{\lambda}\boldsymbol{g}^k-\frac{\hat{\boldsymbol{F}}^k}{\lambda^2+\lambda Tr(\hat{\boldsymbol{F}}^k)}\boldsymbol{g}^k-\nabla J(\theta^k)\right\|^2\right]\\
&=\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{\lambda}(\boldsymbol{g}^k-\nabla J^H(\theta^k))-\frac{\hat{\boldsymbol{F}}^k}{\lambda^2+\lambda Tr(\hat{\boldsymbol{F}}^k)}(\boldsymbol{g}^k-\nabla J^H(\theta^k))-(\nabla J(\theta^k)-\nabla J^H(\theta^k))\right.\right.\\
&\quad\left.\left.+\frac{1}{\lambda}\nabla J^H(\theta^k)-\frac{\hat{\boldsymbol{F}}^k}{\lambda^2+\lambda Tr(\hat{\boldsymbol{F}}^k)}\nabla J^H(\theta^k)-\nabla J^H(\theta^k)\right\|^2\right]\\
&\leq 2\frac{\sigma^2}{N}\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)+2G^2R^2\left(\frac{H+1}{1-\gamma}+\frac{\gamma}{(1-\gamma)^2}\right)^2\gamma^{2H}\\
&\quad+2\left(1+\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla J^H(\theta^k)\|^2]
\end{aligned}
\tag{15}
$$

Furthermore, Assumption 1 tells us that

$$
\begin{aligned}
&\mathbb{E}[\|\nabla J(\theta^k)-\boldsymbol{w}_\star^k\|^2]\\
&=\mathbb{E}[\|\nabla J(\theta^k)-F^{-1}(\theta^k)\nabla J(\theta^k)\|^2]\\
&\leq\left(1+\frac{1}{\mu_F}\right)^2\mathbb{E}[\|\nabla J(\theta^k)\|^2]\\
&\leq\left(1+\frac{1}{\mu_F}\right)^2\left(2\mathbb{E}[\|\nabla J^H(\theta^k)\|^2]+2G^2R^2\left(\frac{H+1}{1-\gamma}+\frac{\gamma}{(1-\gamma)^2}\right)^2\gamma^{2H}\right).
\end{aligned}
\tag{16}
$$

Combining equation 15 and equation 16 with equation 14 gives

$$
\begin{aligned}
&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\boldsymbol{w}^k-\boldsymbol{w}_\star^k\|]\\
&\leq\left(4\frac{\sigma^2}{N}\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)+4G^2R^2\left(\frac{H+1}{1-\gamma}+\frac{\gamma}{(1-\gamma)^2}\right)^2\gamma^{2H}\left(1+\left(1+\frac{1}{\mu_F}\right)^2\right)\right.\\
&\quad\left.+4\left(1+\left(1+\frac{1}{\mu_F}\right)^2+\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla J^H(\theta^k)\|^2]\right)^{0.5}
\end{aligned}
\tag{17}
$$

And recall from equation 12 that

$$
\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla J^H(\theta^k)\|^2]\leq\frac{\frac{J^{H,\star}-J^H(\theta_0)}{K}+(\frac{\eta}{2}+L_J\eta^2)\frac{\sigma^2}{N}}{\frac{\eta}{2}-L_J\eta^2}.
$$

Let us take $\eta=\frac{1}{4L_J}$. Then we get

$$
\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla J^H(\theta^k)\|^2]\leq\frac{J^{H\star}-J^H(\theta_0)}{K}+\frac{3\sigma^2}{N}
$$

In addition, let H, N, and K satisfy

$$
\begin{aligned}
\frac{1}{3}(\frac{\varepsilon}{3G})^2 &\geq 4G^2R^2\left(\frac{H+1}{1-\gamma}+\frac{\gamma}{(1-\gamma)^2}\right)^2\gamma^{2H}\left(1+\left(1+\frac{1}{\mu_F}\right)^2\right)^2 \\
N &\geq \frac{4\sigma^2\left(3+\frac{4}{\lambda^2}+\frac{4G^2}{\lambda^2+\lambda\mu_F}+3\left(1+\frac{1}{\mu_F}\right)^2\right)}{\frac{1}{3}\left(\frac{\varepsilon}{3G}\right)^2}, \\
K &\geq \frac{64L_J(J^{H,\star}-J^H(\theta_0))\left(\left(1+\frac{1}{\mu_F}\right)^2+1+\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)}{\frac{1}{3}\left(\frac{\varepsilon}{3G}\right)^2}.
\end{aligned}
\tag{18}
$$

Then, we have

$$
\frac{G}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\boldsymbol{w}^k-\boldsymbol{w}_\star^k\|]\leq\frac{\varepsilon}{3}. \tag{19}
$$

- Bounding $\frac{1}{K}\sum_{k=0}^{K-1}\|\boldsymbol{w}^k\|^2$.

  We have from equation 15 and equation 12 that

$$
\begin{aligned}
&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\|\boldsymbol{w}^k\|^2 \\
&=\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{\lambda}\boldsymbol{g}^k-\frac{\hat{\boldsymbol{F}}^k}{\lambda^2+\lambda Tr(\hat{\boldsymbol{F}}^k)}\boldsymbol{g}^k\right\|^2\right] \\
&=\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\frac{1}{\lambda}(\boldsymbol{g}^k-\nabla J^H(\theta^k))-\frac{\hat{\boldsymbol{F}}^k}{\lambda^2+\lambda Tr(\hat{\boldsymbol{F}}^k)}(\boldsymbol{g}^k-\nabla J^H(\theta^k))+\frac{1}{\lambda}\nabla J^H(\theta^k)\right.\right. \\
&\quad\left.\left.-\frac{\hat{\boldsymbol{F}}^k}{\lambda^2+\lambda Tr(\hat{\boldsymbol{F}}^k)}\nabla J^H(\theta^k)\right\|^2\right] \\
&\leq 2\frac{\sigma^2}{N}\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)+2\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla J^H(\theta^k)\|^2] \\
&\leq 8\frac{\sigma^2}{N}\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)+\frac{16L_JJ^{H\star}-J^H(\theta_0)}{K}\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right).
\end{aligned}
$$

  Where to get to the last inequality we have taken $\eta=\frac{1}{4L_J}$ and applied Lemma 3.

$$
\begin{aligned}
N &\geq \frac{8\eta\sigma^2\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)}{\frac{\varepsilon}{6}}, \\
K &\geq \frac{16L_J\eta(J^{H,\star}-J^H(\theta_0))\left(\frac{1}{\lambda^2}+\frac{G^2}{\lambda^2+\lambda\mu_F}\right)}{\frac{\varepsilon}{6}},
\end{aligned}
\tag{20}
$$

  we arrive at

$$
\frac{\eta}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\boldsymbol{w}^k\|^2]\leq\frac{\varepsilon}{3}. \tag{21}
$$

- Bounding $\frac{1}{K}\mathbb{E}_{s\sim d_\rho^{\pi^\star}}\left[\mathrm{KL}\left(\pi^\star(\cdot|s)||\pi_{\theta^0}(\cdot|s)\right)\right]$.

By taking

$$K \geq \frac{3\mathbb{E}_{s\sim d_\rho^{\pi^\star}}\left[\mathrm{KL}\left(\pi^\star(\cdot|s)||\pi_{\theta^0}(\cdot|s)\right)\right]}{\eta\varepsilon} \tag{22}$$

we have

$$\frac{1}{\eta K}\mathbb{E}_{s\sim d_\rho^{\pi^\star}}\left[\mathrm{KL}\left(\pi^\star(\cdot|s)||\pi_{\theta^0}(\cdot|s)\right)\right] \leq \frac{\varepsilon}{3}, \tag{23}$$

In summary, we require N and K to satisfy equation 18, equation 20, and equation 22, which leads to

$$N = \mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2}\right), \qquad K = \mathcal{O}\left(\frac{1}{(1-\gamma)^2\varepsilon^2}\right), \qquad H = \mathcal{O}\left(\log((1-\gamma)^{-1}\varepsilon^{-1})\right).$$

By combining equation 19, equation 21, equation 23 and equation 13, we can conclude that

$$J(\pi^\star) - \frac{1}{K}\sum_{k=0}^{K-1} J(\theta^k) \leq \frac{\sqrt{\varepsilon_{\mathrm{bias}}}}{1-\gamma} + \varepsilon.$$

In total, stochastic Sherman-Morrison Policy gradient requires to sample $KN = \mathcal{O}\left(\frac{\sigma^2}{(1-\gamma)^2\varepsilon^4}\right)$ trajectories.

A.4 Additional Abalation Studies

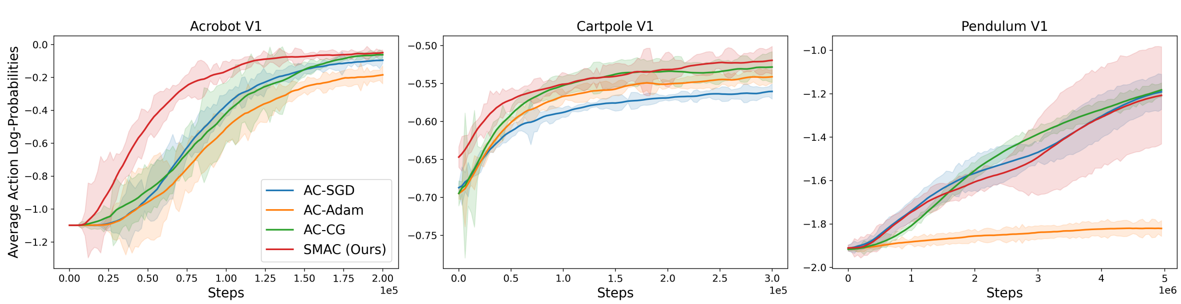

Figure 4: Average log-probabilities of the actions taken on three classic control tasks. All results are averaged over 5 random seeds, with shaded areas showing standard deviation.

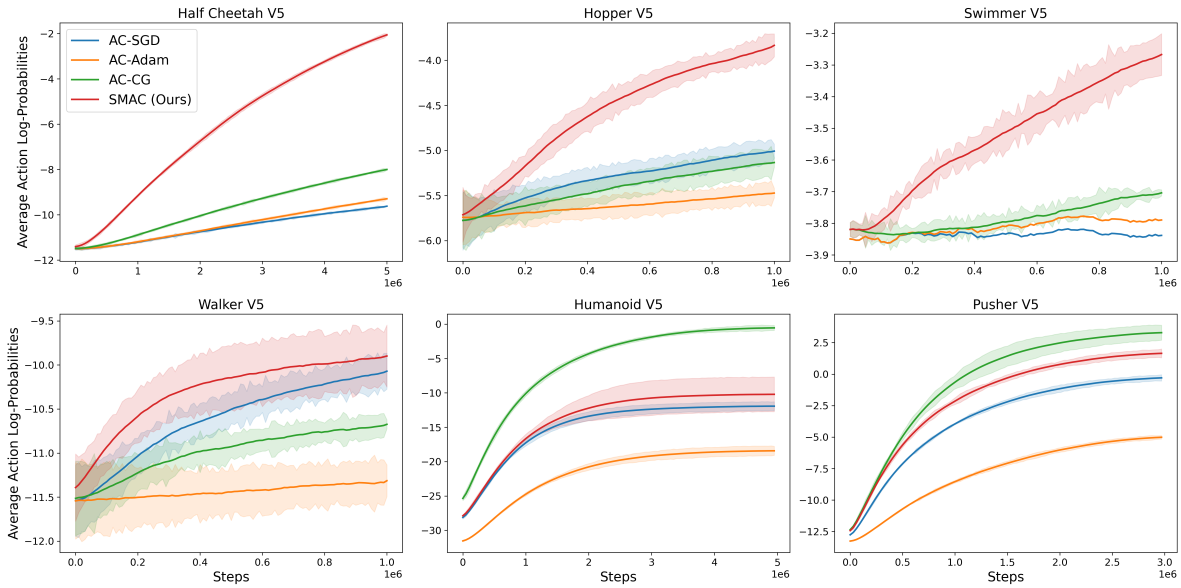

Figure 5: Average log-probabilities of the actions taken on six MuJoCo tasks. All results are averaged over 5 random seeds, with shaded areas showing standard deviation.

A.5 Hyper-parameters for Experiments

- η (actor) : step size for the policy-network update
- α (critic) : learning rate for the value network, optimised with Adam
- T : number of environment steps collected before each parameter update.
- γ : reward discount factor
- λ_{GAE} : trace-decay parameter used by Generalised Advantage Estimation when computing the advantage values A_t.
- λ : damping factor used in the matrix-free Sherman-Morrison approximation of the empirical Fisher.

Table 2: Training hyper-parameters

Environment	**Algorithm**	η (actor)	α (critic)	T	γ	λ_{GAE}	λ
Acrobot	SMAC	5×10^{-2}	1×10^{-3}	1000	0.99	0.9	0.1
	AC–Adam	6×10^{-4}	1×10^{-3}	1000	0.99	0.9	-
	AC–SGD	2×10^{-1}	1×10^{-3}	1000	0.99	0.9	-
	AC–CG	6×10^{-1}	1×10^{-3}	1000	0.99	0.9	-
Cartpole	SMAC	5×10^{-3}	1×10^{-3}	1000	0.99	0.9	0.1
	AC–Adam	7×10^{-5}	1×10^{-3}	1000	0.99	0.9	-
	AC–SGD	7×10^{-3}	1×10^{-3}	1000	0.99	0.9	-
	AC–CG	8×10^{-2}	1×10^{-3}	1000	0.99	0.9	-
Pendulum	SMAC	6×10^{-3}	1×10^{-3}	1000	0.99	0.9	0.1
	AC–Adam	7×10^{-4}	1×10^{-3}	1000	0.99	0.9	-
	AC–SGD	5×10^{-2}	1×10^{-3}	1000	0.99	0.9	-
	AC–CG	3×10^{-2}	1×10^{-3}	1000	0.99	0.9	-
Half Cheetah	SMAC	5×10^{-3}	1×10^{-4}	1000	0.99	0.9	1.0
	AC–Adam	1×10^{-2}	1×10^{-4}	1000	0.99	0.9	-
	AC–SGD	1×10^{-3}	1×10^{-4}	1000	0.99	0.9	-
	AC–CG	1×10^{-2}	1×10^{-4}	1000	0.99	0.9	-
Hopper	SMAC	1×10^{-2}	1×10^{-4}	1000	0.99	0.9	1.0
	AC–Adam	1×10^{-4}	1×10^{-4}	1000	0.99	0.9	-
	AC–SGD	1×10^{-3}	1×10^{-4}	1000	0.99	0.9	-
	AC–CG	1×10^{-2}	1×10^{-4}	1000	0.99	0.9	-
Swimmer	SMAC	3×10^{-2}	1×10^{-4}	1000	0.99	0.9	1.0
	AC–Adam	1×10^{-4}	1×10^{-4}	1000	0.99	0.9	-
	AC–SGD	1×10^{-3}	1×10^{-4}	1000	0.99	0.9	-
	AC–CG	1×10^{-2}	1×10^{-4}	1000	0.99	0.9	-
Walker	SMAC	3×10^{-2}	1×10^{-5}	1000	0.99	0.9	1.0
	AC–Adam	1×10^{-5}	1×10^{-5}	1000	0.99	0.9	-
	AC–SGD	1×10^{-4}	1×10^{-5}	1000	0.99	0.9	-
	AC–CG	1×10^{-2}	1×10^{-5}	1000	0.99	0.9	-
Humanoid	SMAC	1×10^{-4}	1×10^{-3}	1000	0.99	0.9	0.01
	AC–Adam	3×10^{-4}	1×10^{-3}	1000	0.99	0.9	-
	AC–SGD	1×10^{-3}	1×10^{-3}	1000	0.99	0.9	-
	AC–CG	9×10^{-1}	1×10^{-3}	1000	0.99	0.9	-
Pusher	SMAC	1.5×10^{-3}	1×10^{-3}	1000	0.99	0.9	0.01
	AC–Adam	2.5×10^{-3}	1×10^{-3}	1000	0.99	0.9	-
	AC–SGD	1×10^{-1}	1×10^{-3}	1000	0.99	0.9	-
	AC–CG	9×10^{-1}	1×10^{-3}	1000	0.99	0.9	-