# POLICY OPTIMIZATION VIA STOCHASTIC RECURSIVE GRADIENT ALGORITHM

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

This paper proposes the StochAstic Recursive graAdient Policy Optimization (SARAPO) algorithm based on the SARAH algorithm, which is a novel variance reduction algorithm in machine learning. We provide first some theoretical analysis using stochastic differential equations. Furthermore, rich experiments further exemplify the advantage of SARAPO over existing policy gradient methods such as SVRPO and TRPO.

## 1 INTRODUCTION

Reinforcement Learning (RL) (Sutton & Barto, 1998) is a dynamic learning approach to interact with the environment and make the best actions. It achieves remarkable performance in several tasks including control and Atari games. Among all existing RL algorithms, policy gradient (Sutton et al., 1999) is the most fundamental method for RL, and we could use the REINFORCE(Williams, 1992), among many baselines, to estimate the policy gradient.

However, policy gradient suffers from sample inefficiency and high variance during the training phase. To remedy this problem, trust region and line search optimization method (Nocedal & Wright, 2006) [Numerical Optimization] are introduced to policy gradient algorithm (Kakade, 2001). Trust region policy optimization (TRPO) was first proposed by (Schulman et al., 2015). It merges the trust region method and the natural gradient theory (Amari, 1998) into the policy gradient framework. TRPO achieves a more robust performance on continuous control tasks. The basic idea is to make the policy move toward the direction that improves mean episode rewards under the constraint that it *cannot* be too far away from the old policy distribution.

Compared to traditional gradient descent which corresponds to the $L^2$ constraint, constraints that uses KL divergence has been widely adopted as an alternative choice. The method of KL constraint was firstly proposed by Amari (1998). Then in 2001, Kakade merges the natural gradient into the policy gradient (Kakade, 2001) (See Martens (2014) for further extensions). Natural gradient solves the problem of zero gradients in plateau areas, in which KL constraint is highly related with the ellipsoid constraint that is associate with the population Fisher information matrix. Furthermore, TRPO also uses line search to avoid reaching a point outside feasible set/constraint.

SVRG (Johnson & Zhang, 2013) is a kind of stochastic variance reduced method. It overcomes the fallback of SGD of large variance and potentially accelerates the RL training. Previous works used SVRG to accelerate policy gradient (SVRPG) and its TRPO variants (SVRPO) (Xu et al., 2017; Papini et al., 2018). The variance reduced version of policy gradient allows a multi-step optimization for each batch of data, while the vanilla policy gradient optimizes the policy once each iteration. In theory, SVRG accommodates with larger stepsize for each iteration and observably outperforms traditional PG and TRPO, respectively. Furthermore, the introduction of stochastic variance reduction not only improves sample efficiency but also stabilizes the training process by reducing the variance.

In this paper, we focus on an alternative variance reduction method called SARAH, recently proposed by Nguyen et al. (2017) whose convergence rate matches that of SVRG in the convex case and has the stable convergence property that SVRG does not possess. We propose a new algorithm named SARAPO which hybrids the SARAH method with TRPO, and analyze the dynamics via its approximating stochastic differential equation (SDE). In experiments, we show the performance and

sample efficiency of SARAPO, SVRPO, and TRPO and find that SARAPO outperforms TRPO and matches the performance of SVRPO in most of the tasks.

The rest of this paper is organized as the following: we introduce some background, propose and analyze our algorithm in Section 2. In Section 3, we compare SARAH and SVRG from a dynamical system theoretical viewpoint. We present the numerical results of experiments in Section 4 and conclude our papers in Section 5.

## 2 METHODOLOGY

The reinforcement learning task can be considered as solving a discrete time Markov Decision Process (MDP) $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho\}$. Where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma$ is the discount factor and $\rho$ the initial state distribution. The policy gradient method operate by directly maximizing the expected sum of rewarads:

$$L(\pi_w) = \mathbb{E}_{\pi_w}\left[\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t)\right]$$

with respect to the parameters $w$ of the stochastic policy $\pi_w(a|s)$. The gradient of the objective function $\nabla_w L$ is given by:

$$\nabla_w L(w) = \mathbb{E}_{\pi_w}\left[\sum_{t=0}^{\infty} \nabla_w \log \pi(a_t|s_t) Q^{\pi}(s_t, a_t)\right] \tag{1}$$

The expectatioin is calculated by Monte-Carlo simulation by sampling n timesteps $\{s_t, a_t, r_t\}_{t=1}^{n}$ of policy $\pi_w$.

### 2.1 TRUST REGION POLICY GRADIENT

As the policy gradient is highly unstable and sample inefficient, TRPO introduces the trust region method to policy gradient to limit how far the policy go away in each updates. Instead of the $l^2$ norm in trust region method, TRPO takes KL as the measure of distance. After turning the constraint problem into the penalized problem, TRPO optimizes the surrogate loss in every iterations:

$$\max_{w} \quad L_{w_{old}}(w) = \mathbb{E}_{\pi_{w_{old}}}\left[\frac{\pi_w(a|s)}{\pi_{w_{old}}(a|s)} A^{\pi_{w_{old}}}(s, a)\right] - \beta \overline{D}_{KL}(\pi_{w_{old}}, \pi_w) \tag{2}$$

Use first order approximation to the first term of the objective and second order approximation to the second term of the objective. In every iteration, TRPO takes an update of:

$$v_t = \nabla_w L \tag{3}$$

$$w_{t+1} = w_t + \eta_t F^{-1} v_t \tag{4}$$

Where $F^{-1}$ is the Fisher Information Matrix (FIM). The initial step size $\eta$ is calculated by limiting the kl distance within the range $\delta$:

$$\eta_t = \sqrt{2\delta/(v_t F^{-1} v_t)} \tag{5}$$

Then a line search is needed to decide the final step size.

### 2.2 STOCHASTIC GRADIENT RECURSIVE POLICY OPTIMIZATION

SARAH and SVRG are all stochastic variance reduction gradient algorithms in optimization theory. They reduce the variance during optimization process and reaches a faster convergence rate. Indstead of calculating the full gradient $L$. SVRG calculate a mini batch of size $m$ in each inner loop, it gets an estimate of the full gradient with less complexity:

$$v_t = \tilde{v} + \nabla_w L_m - \nabla_w L_m$$

where $\tilde{v}$ and $\tilde{w}$ are saved gradients and saved parameters in the previous outer loop. And $L_m$ is the objective function estimated using $m$ samples.

In Sarah, it estimate each gradients using the gradient estimated one step before and the difference between the gradients of current parameter and the previous parameter on the mini batch.

$$v_t = v_{t-1} + \nabla_w L_m(w_t) - \nabla_w L_m(w_{t-1})$$

The details of calculating the estimated gradients are listed in Algorithm 1

**Algorithm 1** SARAPO

**Initialization:**
Initialize $w_0 = w_1$ to be a randomized parameter.
**for** $t = 1$ to $T$ **do**
  **if** t%m == 0 **then**
    Set $\tilde{w} = w_t$
    Run policy $\pi_{w_t}$ for N timesteps. Store a batch of trajectories $\mathcal{D}$. Calculate the baseline gradients:

$$\tilde{v}_0 = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \nabla L_x(w_t)$$

  **else**
    Draw a mini-batch $\mathcal{D}_t$ of size $M$ from $\mathcal{D}$
    Calculate the estimated gradient $\hat{v}(w)$:

$$\tilde{v}_t = \tilde{v}_{t-1} + \frac{1}{|\mathcal{D}_t|} \sum_{x \in \mathcal{D}_t} (\nabla L_x(w_t) - \nabla L_x(w_{t-1}))$$

  **end if**
  Update policy parameter with Algorithm 2
**end for**
**Output:** $w \leftarrow \tilde{w}^T$

After calculating the estimate of $v$. We do a step of the TRPO update, where the Fisher information matrix is estimated by the second derivative of the KL distance. Notice that the second derivative of KL can approximate the FIM only at the point of the old distribution. So we use the KL distance between the current parameters and the parameters one step before to calculate $\hat{H}$. Futher, the FIM is estimated with only a mini batch of data. In the last step of TRPO, we search for $\eta$ that can improve the value of the objective function.

**Algorithm 2** Inner loop details

estimate the Fisher Information Matrix $\hat{H} = \frac{\partial^2}{\partial w_n \partial w_m} \overline{KL}(\pi_{w_t}(\cdot \| s_n), \pi_w(\cdot \| s_n))|_{w_t}$
Update policy parameter with mini-batch $I_j$:

$$w_{t+1} \leftarrow w_t + \eta_t \hat{H}^{-1}(w_t) \cdot \hat{v}_t.$$

Initialize $\eta_t$ according to equation 5
linesearch $\eta_j, \eta_j/2, \eta_j/4, \cdots$ ,for improving $L_{\pi_{\tilde{w}}(\pi_w)}$

## 3 Theory of Dynamical Systems

The core inner-loop update rule of Stochastic Recursive Gradient Method (SARAH) (Nguyen et al., 2017) is, assuming the minibatch size is 1,

$$\begin{aligned} v_{t+1} =& v_t + \nabla L_{i_{t+1}}(w_t) - \nabla L_{i_{t+1}}(w_{t-1}), \\ w_{t+1} =& w_t - \eta v_{t+1}. \end{aligned}$$

Here we provide a simple alternative viewpoint of (random) dynamical systems. As we could see from Algorithm 1, we applied a *scaled-form* of SARAH in combination with Trust Region Policy

Optimization by the inverse Fisher Information matrix $F^{-1}$. Seeing this, the inner loop update has the essential form of

$$\begin{aligned} v_{t+1} =& v_t + \nabla L_{i_{t+1}}(w_t) - \nabla L_{i_{t+1}}(w_{t-1}), \\ w_{t+1} =& w_t - \eta F^{-1} v_{t+1}. \end{aligned} \tag{6}$$

For simplicity of analysis, we consider the *quadratic approximation* approach. Let the quadratic objective function $L(w) := \frac{1}{n}\sum_{i=1}^{n} L_i(w)$, with

$$L_i(w) = \frac{1}{2} w^\top H_i w + g_i^\top w.$$

We sketch the analysis using ODE and SDE, in order to better understand the dynamics.

### 3.1 ODE ANALYSIS

By rescaling the time $t = \eta^{-1} s$ according to the stepsize $\eta$, we define a new process

$$V(s) = v_{\lfloor \eta^{-1} s \rfloor} \quad \text{and} \quad X(s) = w_{\lfloor \eta^{-1} s \rfloor}.$$

Then the newly defined process is converted from equation 6 and has the form

$$\begin{aligned} V(s+\eta) =& V(s) + \nabla L_{i_{s+\eta}}(X(s)) - \nabla L_{i_{s+\eta}}(X(s-\eta)) \\ =& V(s) + HX(s) - HX(s-\eta) + \eta e_{s+\eta}, \end{aligned}$$

where $e_{s+\eta}$ is a martingale difference noise term of $O(1)$, and

$$X(s+\eta) = X(s) - \eta F^{-1} V(s+\eta).$$

Substituting $X(s) - X(s-\eta)$ with $-\eta F^{-1} V(s)$, we derive the update rule for $V(s)$

$$\eta^{-1}(V(s+\eta) - V(s)) = -HF^{-1}V(s) + e_{s+\eta}.$$

From the standard weak convergence theory, one easily conclude the following:

**Theorem 1** *When $\eta \to 0^+$, the scaled SARAH update rule (6) can be approximated by the following ODE system*

$$\begin{aligned} \frac{d}{ds} V(s) &= -HF^{-1}V(s), \\ \frac{d}{ds} X(s) &= F^{-1}V(s). \end{aligned} \tag{7}$$

*Note in (7), the dynamics of $V(s)$ forms an autonomous ODE, which is independent of the dynamics of $X(s)$.*

Left multiplying both sides of the first line of equation 7 by $F^{-1}$, one obtains

$$\frac{d}{ds}(F^{-1}V(s)) = -F^{-1}H(F^{-1}V(s)).$$

Noticing $V(0) = HX_0$, we can easily verify that the $V(s)$ can be solved as

$$V(s) = F(F^{-1}V(s)) = F\exp(-s \cdot F^{-1}H)F^{-1}HX_0 = H\exp(-s \cdot F^{-1}H)X_0. \tag{8}$$

By combining $V(s)$'s solution with the update rule for $X(s)$, we have

$$\frac{d}{ds} X(s) = -\exp(-s \cdot F^{-1}H)F^{-1}HX_0,$$

and has the expressed solution

$$\begin{aligned} X(s) &= X_0 - \int_0^s \exp(-r \cdot F^{-1}H)dr\, F^{-1}HX_0 = X_0 - \int_0^s \sum_{k=0}^{\infty} \frac{(-r)^k}{k!}(F^{-1}H)^{k+1} dr X_0 \\ &= X_0 - \sum_{k=0}^{\infty} \frac{(-r)^{k+1}}{(k+1)!}(F^{-1}H)^{k+1} \cdot X_0 = X_0 - (I - \exp(-s \cdot F^{-1}H))X_0. \end{aligned}$$

As a result, we reach the following solution to (7):

$$X(s) = \exp(-s \cdot F^{-1}H)X_0. \tag{9}$$

From (8) and (9), both the vectors $X(s)$ and $V(s)$ exponentially decays with rate matrix $F^{-1}H$ which is consistent with the theory of natural gradient ascent (Kakade, 2001).

### 3.2 SDE ANALYSIS

The ODE analysis simply neglects the effect of noise. Here we turns to develop a stochastic differential equation approximation tool to exploit the effect of noise. Denote $H_i = H + E_i$, $H$ being the Hessian for $L(w)$. Further assume that the noise term $E_i$ is $\sigma diag(\chi^1, \ldots, \chi^d)$ where $\chi^i$ are i.i.d. standard normal, for simplicity.

Define the coordinate-wise product of two vectors $(x_1, \ldots, x_d)^\top \bullet (y_1, \ldots, y_d)^\top = (x_1 y_1, \ldots, x_d y_d)^\top$. Let $W(s) = (W_1(s), \ldots, W_d(s))^\top$ be a $d$-dimensional standard Wiener processes. Then under standard assumptions, the inner loop has an update rule for $V(s)$ which can be written as

$$\begin{aligned} V(s+\eta) - V(s) =& -\eta H F^{-1} V(s) - \eta\sigma\, \chi \bullet (F^{-1} V(s)) \\ =& -\eta H F^{-1} V(s) - \eta^{1/2}\sigma\, (W(s+\eta) - W(s)) \bullet (F^{-1} V(s)). \end{aligned}$$

As $\eta \to 0^+$, it becomes a SDE system

$$\frac{d}{ds} V(s) = -H F^{-1} V(s) - \eta^{1/2}\sigma \left( \frac{d}{ds} W(s) \right) \bullet (F^{-1} V(s)),$$

which can be simplified as

$$dV(s) = -H F^{-1} V(s) ds - \eta^{1/2}\sigma (F^{-1} V(s)) \bullet dW(s). \tag{10}$$

From the above analysis, one can verify that the approximating SDE in (10) introduces a small (size of $O(\eta^{1/2})$) noise deviation from its ODE curve, and hence enjoys similar property as the natural gradient descent update. In the case where $H$ is convex, the dynamics validates the property of inner-loop convergence in (Nguyen et al., 2017) [Theorem 1a & 1b], and hence the multi-loop geometric convergence [Theorem 2]. The per-step cost, however, turns to be evaluations of two individual gradients instead of the whole-batched gradients.

**Remark 1** *One may also conduct a similar analysis for SVRG and conclude the same approximating ODE for scaled SVRG. The approximating SDE, nevertheless, shall differ by an extra noise term that is proportional to the distance between the current iterate and the control point. This well-explains the inner-loop convergence and stability properties that SARAH enjoy upon SVRG.*

## 4 EXPERIMENTS

We designed experiments to test the performance of SARAPO and SVRPO. Our implementation is based on the $modular_rl$. We used the commonly used Mujoco testing environment on several continuous control tasks: Swimmer, Half-Cheetah, Walker and Hopper. We compared our implementation of SARAPO with SVRPO and the original TRPO. The result is shown in Figure 1

We fix part of our parameters according to Xu et al. (2017) and Schulman et al. (2015). We set the Number of steps per Iteration to be 50000 and Discount to be 0.995. We use a Gaussian Policy with a diagonal covariance matrix, whose mean is parameterized by a multi-layer perceptron (MLP). For the Mujoco tasks, we use two hidden layers of size $(64, 64)$ with tanh activation function. We fine tuned other parameters as follows, minibatch size is tuned between 1000 or 5000, inner loop iterations is between 50 or 20, inner loop max kl between 0.005 and 0.01, and the cg damping factor. We list the detailed parameters of each experiments in the appendix.

As the result varies a lot accross different runs, we chose five different random seed and plot our result as the average of the five runs. As the outer loop iterations are fixed, the x-axis is proportional to the number of samples we have sampled, the curve corresponds to the sample efficiency.

In Figure 1 We compared the learning curve of three algorithms mentioned above: SARAPO, SVRPO and TRPO. We see that in Half-Cheetah environment, SARAPO observably performs better than other two algorithms and achieves high score in 500 epochs. In Walker and Hopper, the three algorithms have minor difference. SARAPO rises faster in Swimmer but SVRPO reaches a better score.

To get a clearer view of our result, we list two tables.

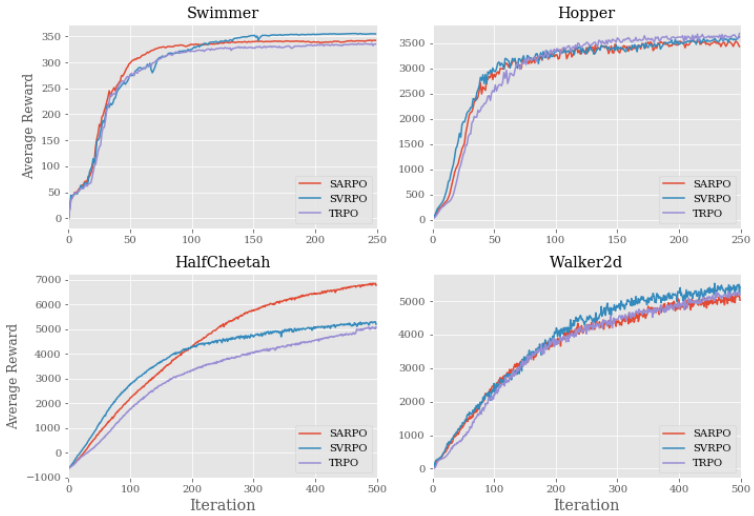


Figure 1: A comparison of variance reduced policy optimization and traditional TRPO. The result is the mean of five runs on different random seeds.

Table 4 presents the number of iterations needed to cross the threshold which is 90 percent of the best score. In two of the tasks, SARAPO reaches the threshold the fastest and in the other two tasks, SVRPO outperforms the other two.

| Tasks | Threshold | SARAPO | SVRPO | TRPO |
|---|---|---|---|---|
| Swimmer | 302 | **52** | 72 | 71 |
| Half-Cheetch | 4561 | **216** | 251 | 402 |
| Walker | 4669 | 343 | **274** | 330 |
| Hopper | 3198 | 74 | **65** | 78 |

Table 1: Number of Iteration Below the Threshold

| Tasks | SARAPO | SVRPO | TRPO |
|---|---|---|---|
| Swimmer | 343.0 | **355.7** | 336.5 |
| Half-Cheetch | **6845.1** | 5291.4 | 5080.9 |
| Walker | 5232.9 | **5529.2** | 5265.2 |
| Hopper | 3578.1 | 3625.8 | **3692.8** |

Table 2: Performance of Policy

Table 2 shows the final performance of policy which is the average of best reward over five different seeds.

We did more experiments on the phenomenons during the training process. We record three statistics in each mini iterations: the norm of estimatedronghe gradients, the improvement of the policy loss, and the kl distance between the old policy and the current policy. Figure /refstats mix the result of all three statistcs of three different tasks. The first row shows the training process of Swimmer, the second row is the Hopper task and the third row is Half-Cheetah. To see the details, we choose an interval that the policy is still improving and limit our x-axis to be in that range and only one random seed is chosen to plot the data.

In the first column, we plot the $l^2$ norm of the estimated gradient of SVRG and SARAH.

Then we calculate the loss function both before an update and after an update. For SARAPO and SVRPO, we sum the improvement in each mini iterations to get the improvement of the outer it-

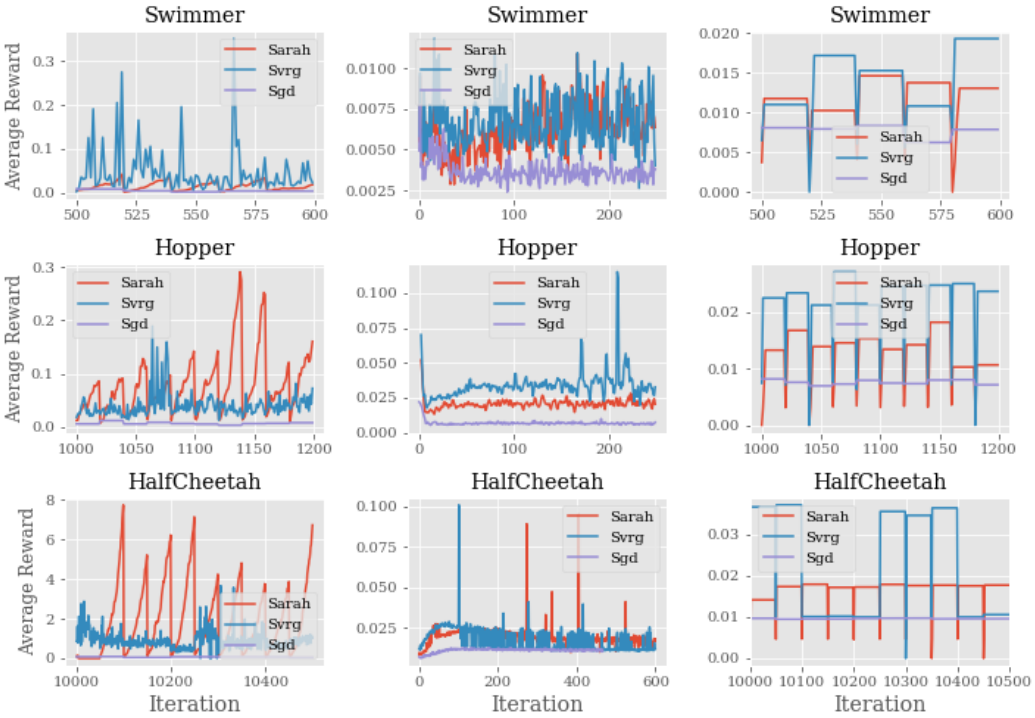


Figure 2: An illustration of the policy loss improvement, the estimated gradients norm and the kl during training. The first row is the results on the Swimmer task. The second row is the results on the Hopper task. The third row is the results on the HalfCheetah. The pictures are zoomed in to show more details and the x-axis is normalized to be the number of outer loop iterations.

erations. We compare the result with that in TRPO. From the second colum of Figure **??**, we see that both SARAH and SVRG reach a better optima than SGD. Generally, a larger improvement corresponds to high mean episode rewards. For the random seed chosen to plot Figure **??**, SVRPO outperforms SARAPO in the first few iterations in Swimmer, and SARAPO outperforms SVRPO at last in Half-Cheetah.

The third column shows the kl divergence between the old policy and the updated policy. We see that when $\delta$ is set to be 0.01, the kl seldom exceed 0.01 in TRPO. SARAPO stays under 0.02 and SVRPO stays under 0.03. The variance reduced gradient methods are able to find a better solution slightly outside the kl range.

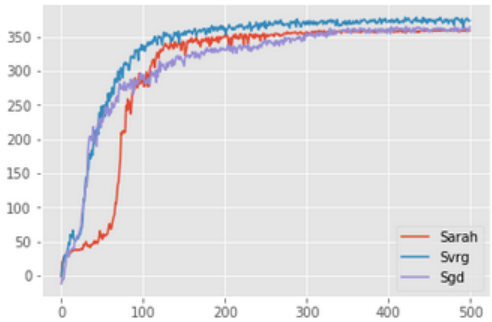


Figure 3: Comparison of Stochastic Recursive Proximal Policy Gradient to PPO

Finally, we do an experiment on Swimmer of the comparison between SARAPPO, SVRPPO and PPO. Both SARAPPO and SVRPPO reaches a better reward than PPO.

## 5 CONCLUSION

In this work, we propose the Stochastic Recursive Gradient Policy Optimization (SARAPO) algorithm based on the novel SARAH algorithm for variance reduction (Nguyen et al., 2017). We provide from both theory and practical experiments on its advantage over existing policy gradient methods. We hope this work can inspire future works and ultimately obtain higher performance in the reinforcement learning tasks.

## REFERENCES

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10(2):251–276, February 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL http://dx.doi.org/10.1162/089976698300017746.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Sham Kakade. A natural policy gradient. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 1531–1538, Cambridge, MA, USA, 2001. MIT Press. URL http://dl.acm.org/citation.cfm?id=2980539.2980738.

James Martens. New perspectives on the natural gradient method. *CoRR*, abs/1412.1193, 2014.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pp. 2613–2621, 2017.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4026–4035, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/papini18a.html.

John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1889–1897. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045319.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press. URL http://dl.acm.org/citation.cfm?id=3009657.3009806.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Tianbing Xu, Qiang Liu, and Jian Peng. Stochastic variance reduction for policy gradient estimation. *CoRR*, abs/1710.06034, 2017.

# A SVRGPPO

**Algorithm 3** SVRG Proximal Policy Gradient

**Input:**
  **Initialization:**
  Initialize $\tilde{w}^0$ randomly
  **for** $\ell = 1$ to $L$ **do**
    Generate $N$ transitions $\mathcal{D}_\ell$ by executing the current policy $\pi_{\tilde{w}^\ell}$.
    $L_{\tilde{w}^\ell} = \frac{1}{N}\sum_t [\frac{\pi_w(a_t|s_t)}{\pi_{\tilde{w}^\ell}(a_t|s_t)}\hat{A}_t - \beta KL[\pi_{\tilde{w}^\ell}(\cdot|s_t), \pi_w(\cdot|s_t)]]$
    Compute whole gradient $\tilde{g} = \nabla L(\tilde{w}^\ell)$
    $w_0^\ell = \tilde{w}^\ell$
    **for** $j = 1$ to $J$ **do**
      Draw a minibatch $\mathcal{D}_j$ from $\mathcal{D}_\ell$
      Compute gradient $g = \tilde{g} + \nabla L_j(w_j^\ell) - \nabla L_j(\tilde{w}^\ell)$
      $w_{j+1}^\ell = Adam(w_j^\ell, g)$
    **end for**
    Compute $d = \frac{1}{N}\sum_t KL[\pi_{\tilde{w}^\ell}(\cdot|s_t), \pi_{w_J^\ell}(\cdot|s_t)]$
      if $d < d_{targ}/1.5, \beta \leftarrow \beta/2$
      if $d > d_{targ} \times 1.5, \beta \leftarrow \beta \times 2$
    $\tilde{w}^{\ell+1} = w_J^\ell$
    Update value estimation using $\mathcal{D}_\ell$
  **end for**
**Output:** $w \leftarrow w^L$

# B HYPERPARAMETERS

| Tasks | Minibatch Size | Inner Loop Iteration | Max KL | Cg Damping Factor |
|---|---|---|---|---|
| Swimmer | 5000 | 20 | 0.005 | 0.1 |
| Half-Cheetch | 5000 | 50 | 0.005 | 0.1 |
| Walker | 1000 | 20 | 0.005 | 0.1 |
| Hopper | 1000 | 20 | 0.005 | 0.1 |

Table 3: Best Hyper-Parameter of SARAPO

| Tasks | Minibatch Size | Inner Loop Iteration | Max KL | Cg Damping Factor |
|---|---|---|---|---|
| Swimmer | 1000 | 20 | 0.01 | 0.1 |
| Half-Cheetch | 5000 | 50 | 0.01 | 0.1 |
| Walker | 5000 | 50 | 0.01 | 0.1 |
| Hopper | 5000 | 20 | 0.01 | 0.1 |

Table 4: Best Hyper-Parameter of SVRPO

# C THE MEAN EPISODE REWARD OF THE EXPERIMENTS IN FIGURE ??

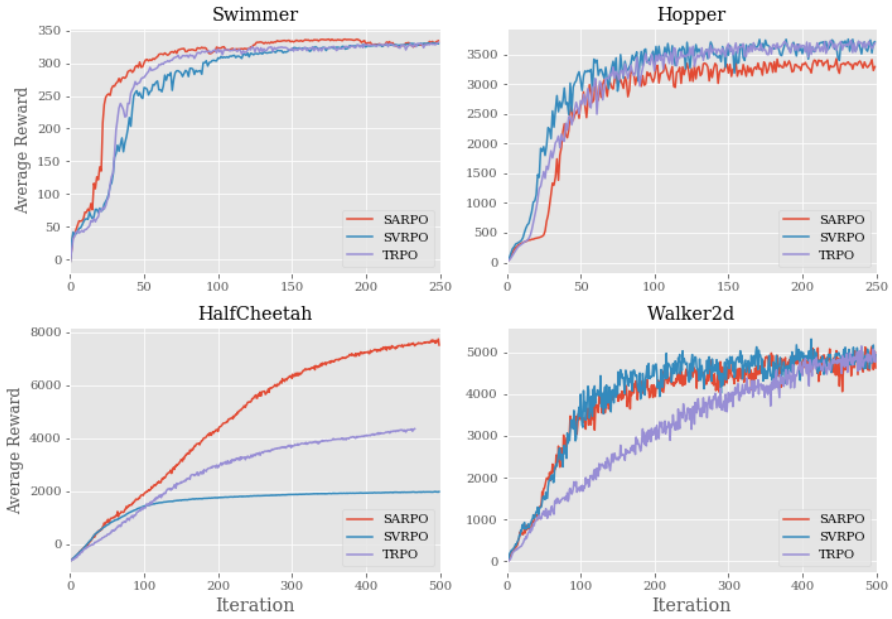


Figure 4: The Mean Episode Reward of the experiments in Figure **??**