SAMPLES ARE USEFUL? NOT ALWAYS: DENOISING POLICY GRADIENT UPDATES USING VARIANCE EXPLAINED

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy gradient algorithms in reinforcement learning optimize the policy directly and rely on efficiently sampling an environment. However, while most sampling procedures are based solely on sampling the agent's policy, other measures directly accessible through these algorithms could be used to improve sampling before each policy update. Following this line of thoughts, we propose the use of SAUNA, a method where transitions are rejected from the gradient updates if they do not meet a particular criterion, and kept otherwise. This criterion, the fraction of variance explained $\mathcal{V}^{ex}$, is a measure of the discrepancy between a model and actual samples. In this work, $\mathcal{V}^{ex}$ is used to evaluate the impact each transition will have on learning: this criterion refines sampling and improves the policy gradient algorithm. In this paper: (a) We introduce and explore $\mathcal{V}^{ex}$, the criterion used for denoising policy gradient updates. (b) We conduct experiments across a variety of benchmark environments, including standard continuous control problems. Our results show better performance with SAUNA. (c) We investigate why $\mathcal{V}^{ex}$ provides a reliable assessment for the selection of samples that will positively impact learning. (d) We show how this criterion can work as a dynamic tool to adjust the ratio between exploration and exploitation.

1 INTRODUCTION

Learning to control agents in simulated environments has been a challenge for decades in reinforcement learning (Nguyen & Widrow, 1990; Werbos, 1989; Schmidhuber & Huber, 1991; Robinson & Fallside, 1989) and has recently led to a lot of research efforts in this direction (Mnih et al., 2013; Burda et al., 2019; Ha & Schmidhuber, 2018; Silver et al., 2016; Espeholt et al., 2018), notably in policy gradient methods (Schulman et al., 2016; Silver et al., 2014; Lillicrap et al., 2016; Mnih et al., 2016). Despite the definite progress made, policy gradient algorithms still heavily suffer from sample inefficiency (Kakade, 2003; Wu et al., 2017; Schulman et al., 2017; Wang et al., 2017).

In particular, many policy gradient methods are subject to use as much experience as possible in the most efficient way. We make the hypothesis that *not all experiences are worth* to use in the gradient update. In other words, while perhaps trajectory simulations should be as rich as possible, some samples may instead add noise to the gradient update and hinder learning. Both the number of samples and the quality of the sampled transitions have a critical impact on the behavior of the agent: the better the experience, the better the resulting policy and the better the environment sampling. In essence, the quality of the sampling procedure conditions the final performance of the agent.

SAUNA aligns the agent's immediate ability in each environment with the experiences that will affect its learning: the fraction of variance explained $\mathcal{V}^{ex}$ will condition the rejection of samples before the policy update. We will examine the impact of filtering some of the transitions out and study how SAUNA affects the learning performance across a variety of tasks from *MuJoCo* (Todorov et al., 2012), Roboschool, and the *Atari 2600* domain (Bellemare et al., 2013). We also discuss the limitations of our method in the context of the policy gradient theorem and show how SAUNA can work as a dynamic tool for efficiently balancing exploration with exploitation.

We exploit this for on-policy learning: first for its unbiasedness and stability compared to off-policy methods (Nachum et al., 2017), second because on-policy is empirically known as being less sample

efficient than off-policy learning and therefore increased interest in this research topic. However, our method can be applied to off-policy methods as well, and we leave this investigation open for future work.

The contributions of this paper are summarized as follows:

1. We propose to move from reward-centered learning to learning that takes into account the agent's knowledge. We hypothesize that the agent's ability in an environment can partially be measured through $\mathcal{V}^{ex}$. We explore how the use of this criterion can drive the alignment between the samples used to update the policy and the agent's progress.
2. We provide a method that transforms policy gradient algorithms by assuming that not all samples are useful for learning and that these disturbing samples should, therefore, be rejected. While our method is a simple extension of policy gradient algorithms, it adds a variance criterion to the optimization problem and introduces a novel rejection sampling procedure.
3. By combining (1) and (2), we obtain a learning algorithm that is empirically effective in learning neural network policies for challenging control tasks. In addition to showing that all samples are not useful and that some should be rejected, our results extend the state-of-the-art in using reinforcement learning for high-dimensional continuous control.

2 PRELIMINARIES

We consider a Markov Decision Process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$ and reward function $r(s, a)$ where $s \in \mathcal{S}$, $a \in \mathcal{A}$. Let $\pi = \{\pi(a|s), s \in \mathcal{S}, a \in \mathcal{A}\}$ denote a stochastic policy and let the objective function be the traditional expected discounted reward:

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

where $\gamma \in [0, 1)$ is a discount factor (Puterman, 1994) and $\tau = (s_0, a_0, s_1, \dots)$ is a trajectory sampled from the environment.

Policy gradient methods aim at modelling and optimizing the policy directly (Williams, 1992). The policy π is generally implemented with a function parameterized by θ. In the sequel, we will use θ to denote the parameters as well as the policy (assuming the architecture of the neural net is fixed and well defined). In deep reinforcement learning (DRL), the policy is represented in a neural network called the policy network and is assumed to be continuously differentiable with respect to its parameters θ.

2.1 POLICY GRADIENT METHOD WITH CLIPPED SURROGATE OBJECTIVE

We use PPO (Schulman et al., 2017), an on-policy gradient-based algorithm. In previous work, PPO has been tested on a set of benchmark tasks and has proven to produce impressive results in many cases despite a relatively simple implementation. For instance, instead of imposing a hard constraint as does TRPO (Schulman et al., 2015), PPO formalizes the constraint as a penalty in the objective function. In PPO, at each iteration, the new policy θ_{new} is obtained from the old policy θ_{old}:

$$\theta_{new} \leftarrow \mathop{\operatorname{argmax}}_{\theta} \mathop{\mathbb{E}}_{s_t, a_t\sim\pi_{\theta_{old}}}\left[L^{\mathrm{PPO}}\left(s_t, a_t, \theta_{old}, \theta\right)\right]. \tag{2}$$

We use the clipped version of PPO whose objective function is:

$$L^{\mathrm{PPO}}(s_t, a_t, \theta_{old}, \theta) = \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A^{\pi_{\theta_{old}}}(s_t, a_t),\ g(\epsilon, A^{\pi_{\theta_{old}}}(s_t, a_t))\right), \tag{3}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A, A \geq 0 \\ (1-\epsilon)A, A < 0. \end{cases} \tag{4}$$

A is the advantage function, $A(s,a) \triangleq Q(s,a) - V(s)$ (see Appendix F). The expected advantage function $A^{\pi_{\theta_{old}}}$ is estimated by an old policy and then re-calibrated using the probability ratio between the new and the old policies. By taking the minimum of the two terms in Eq. (3), the ratio is constrained to stay within a small interval around 1, making the training updates more stable.

2.2 Related Work

Our method incorporates three key ideas: (a) policy and value function approximation with a neural network architecture combining or separating the actor and the critic, (b) an on-policy setting enabling more expected unbiasedness and stability than an off-policy formulation and (c) a policy gradient update improved by conditioning the use of samples with the fraction of variance explained to allow for better sampling and more efficient learning. Below, we consider previous works that build on some of these approaches.

Actor-critic algorithms essentially use the value function to alternate between policy evaluation and policy improvement (Sutton & Barto, 1998; Barto et al., 1983). In order to update the actor, many methods adopt the on-policy formulation (Peters & Schaal, 2008; Mnih et al., 2016; Schulman et al., 2017). However, despite their important successes, these methods suffer from sample complexity.

In the literature, research has also been conducted in prioritization sampling. While Schaul et al. (2016) makes the learning from experience replay more efficient by using the TD error as a measure of these priorities in an off-policy setting, our method directly selects the samples on-policy. Schmidhuber (1991) is related to our method in that it calculates the expected improvement in prediction error, but with the objective to maximize the intrinsic reward through artificial curiosity. Instead, our method estimates the expected fraction of variance explained and filters out some of the samples to improve the learning efficiency.

Finally, motion control in physics-based environments is a long-standing and active research field. In particular, there are many prior works on continuous action spaces (Schulman et al., 2016; Levine & Abbeel, 2014; Lillicrap et al., 2016; Heess et al., 2015) that demonstrate how locomotion behavior and other skilled movements can emerge as the outcome of optimization problems.

3 Method

3.1 Variance explained: $\mathcal{V}^{ex}$

For a trajectory τ, we define $\mathcal{V}^{ex}_{\tau}$ as the *fraction of variance explained*. It is the fraction of variance that the value function explains about the returns and corresponds to the proportion of the variance in the dependent variable V that is predictable from the independent variable s_t. We compute $\mathcal{V}^{ex}_{\tau}$ at each policy gradient update with the samples used for the gradient computation. In statistics, this quantity is also known as the coefficient of determination R^2 (Kvålseth, 1985). For the sake of clarity, instead of using the notation R^2 we will refer to this criterion as $\mathcal{V}^{ex}_{\tau}$:

$$\mathcal{V}^{ex}_{\tau} \triangleq 1 - \frac{\sum_{t\in\tau}\left(\hat{R}_t - V(s_t)\right)^2}{\sum_{t\in\tau}\left(\hat{R}_t - \overline{R}\right)^2}, \tag{5}$$

where $\hat{R}_t$ and $V(s_t)$ are respectively the return and the expected return from state $s_t \in \tau$, and $\overline{R}$ is the mean of all returns in trajectory τ. It should be noted that this criterion may be negative for non-linear models, indicating a severe lack of fit (Kvålseth, 1985) of the corresponding function:

- $\mathcal{V}^{ex}_{\tau} = 1$ if the fitted value function V perfectly explains the returns;
- $\mathcal{V}^{ex}_{\tau} = 0$ corresponds to a simple average prediction;
- $\mathcal{V}^{ex}_{\tau} < 0$ if the value function provides a worse fit to the outcomes than the mean of the discounted rewards.

Interpretation. $\mathcal{V}^{ex}$ measures the ability of the value function to fit the returns. $\mathcal{V}^{ex} = 0.43$ implies that 43% of the variability of the dependent variable $\hat{R}$ has been accounted for, and the remaining 57% of the variability is still unaccounted for. By its definition, this quantity is a highly relevant indicator for assessing self-performance in reinforcement learning.

3.2 $\mathcal{V}^{ex}$ APPLIED TO PPO

When applying policy gradient methods using a neural network for function approximation, we use either shared parameters for the policy (actor) and value (critic) function or a copy of the same architecture for both. For shared parameters configurations, an error term on the value estimation is added to the PPO objective. In addition to the policy and the value functions, our method adds a third head to the shared network. Let $\mathcal{V}^{ex}_{\theta}(s_t)$ be the prediction of $\mathcal{V}^{ex}_{\tau}$ under parameters θ at state $s_t \in \tau$. The final objective becomes:

$$L(s_t, a_t, \theta_{old}, \theta) = \mathbb{E}\left[L^{\text{PPO}}(s_t, a_t, \theta_{old}, \theta) - c_1\left(V_\theta(s_t) - \hat{R}_t\right)^2 - c_2\left(\mathcal{V}^{ex}_{\theta}(s_t) - \mathcal{V}^{ex}_{\tau}\right)^2\right], \quad (6)$$

where c_1 and c_2 are respectively the coefficient for the squared-error loss of the value function and of the fraction of variance explained function. It is important to note that although $\mathcal{V}^{ex}_{\tau}$ is defined for a sampled trajectory τ, the model predicts its value at each state $s_t \in \tau$. For cases where the network is not shared between the policy and the value function, $\mathcal{V}^{ex}_{\tau}$ is added to the value function network. Appendix A illustrates well how the new head is embedded in the original architecture. The rest of the network is unchanged, making it very easy to use SAUNA without altering the complexity of existing policy gradient methods.

3.3 SAUNA: $\mathcal{V}^{ex}$-DIRECTED UPDATE

For simplicity, we rewrite $\mathcal{V}^{ex}_{\theta}(s_t)$ as $\mathcal{V}^{ex}_{t}$. Let $\tilde{\mathcal{V}}^{ex}_{0:t-1}$ be the median of $\mathcal{V}^{ex}_{\theta}$ between timesteps 0 and $t-1$. The filtering condition is:

$$\frac{|\mathcal{V}^{ex}_{t}|}{|\tilde{\mathcal{V}}^{ex}_{0:t-1}| + \epsilon_0} \geq threshold, \quad (7)$$

where $\epsilon_0 = 10^{-8}$ is to avoid division by zero. At each timestep t, if the state s_t complies with this condition, then its associated transition is added to the current on-policy buffer (i.e. added as a training sample for the on-policy gradient update). If not, the action is simply executed and the model considers the next state s_{t+1}. The process continues until the trajectory is T-steps long.

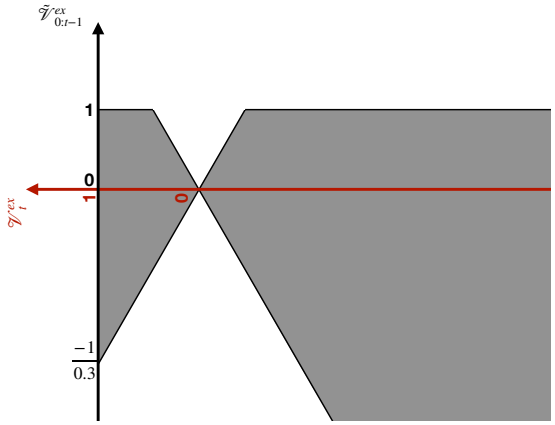


Figure 1: Grey samples: kept for the gradient update. White samples: discarded ($threshold$ = 0.3).

Interpretation. Fig. 1 illustrates where the accepted (grey area) and excluded (white area) samples stand with respect to their $\mathcal{V}^{ex}_{t}$, and relative to the median of the previous $\mathcal{V}^{ex}_{0:t-1}$ in the trajectory. The figure depicts how the filtering condition dynamically selects the transitions for which $\mathcal{V}^{ex}_{t}$ is either high or low, but not in between: those are the transitions that will impact the most the learning. Indeed, a high score means that the sample corresponds to a state for which the value function estimates well its utility. On the contrary, a low score means that the value function does not fit well in this particular state. Finally, a score near zero means that the value function is performing just as good as taking the empirical mean of the returns.

Algorithm 1 SAUNA: $\mathcal{V}^{ex}$-directed update.

Initialize policy parameters θ_0
Initialize value function parameters ϕ_0 and $\mathcal{V}^{ex}$ function parameters ψ_0

for $k = 0, 1, 2, \ldots$ **do** ▷ For each update step

  Initialize trajectory τ to capacity T
  while size(τ) $\leq T$ **do** ▷ For each timestep t
    $a_t \sim \pi_{\theta_k}(s_t)$, $v_t = V_{\phi_k}(s_t)$, $\mathcal{V}_t^{ex} = \mathcal{V}_{\psi_k}^{ex}(s_t)$
    execute action a_t and observe reward r_{t+1} and next state s_{t+1}
    if $\frac{|\mathcal{V}_t^{ex}|}{|\tilde{\mathcal{V}}_{0:t-1}^{ex}|+\epsilon_0} \geq$ *threshold* **then**
      collect transition $(s_t, a_t, r_t, v_t, s_{t+1}, \mathcal{V}_t^{ex})$ in τ
    else
      continue without collecting the transition

  Gradient Update

$$\theta_{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} \sum_{t \in \tau} \min \left(\frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_k}\left(a_t|s_t\right)} A^{\pi_{\theta_k}}\left(s_t, a_t\right), g\left(\epsilon, A^{\pi_{\theta_k}}\left(s_t, a_t\right)\right) \right) \tag{8}$$

$$\phi_{k+1} \leftarrow \underset{\phi}{\operatorname{argmin}} \sum_{t \in \tau} \left(V_{\phi_k}\left(s_t\right) - \hat{R}_t \right)^2 \tag{9}$$

$$\psi_{k+1} \leftarrow \underset{\psi}{\operatorname{argmin}} \sum_{t \in \tau} \left(\mathcal{V}_{\psi_k}^{ex}\left(s_t\right) - \hat{\mathcal{V}}_{\tau}^{ex} \right)^2 \tag{10}$$

Algorithm 1 illustrates how learning is achieved, in particular, the fitting of the $\mathcal{V}^{ex}$ function in Eq. (10) and how only collected samples are used for updates in the *if* statement. We have chosen to depict a configuration where the parameters between the policy network, the value function and the $\mathcal{V}^{ex}$ function are not shared, since from this configuration the shared parameter case is direct.

4 Experiments

In this section, unless otherwise stated, all curves correspond to the average of 6 runs with different seeds, and shaded areas are standard deviations. For ease of reproducibility and sharing, we have forked the original *baselines* repository from OpenAI and modified the code to incorporate our method[1]. The complete list of hyperparameters and details of our implementation are given in Appendix B and C respectively. A discussion about additional experiments whose results are non-positive, but which we think contribute positively to this paper, can be found in Appendix E.

4.1 Comparison in the continuous domain: MuJoCo

We begin by comparing SAUNA (PPO+Vex in red) with its natural baseline PPO introduced in section 2 (PPO in blue). We use 6 simulated robotic tasks from OpenAI Gym (Brockman et al., 2016) using the MuJoCo physics engine. Except for the two hyperparameters required by our method, namely *threshold* $= 0.3$ from Eq. (5) and $c_2 = 0.5$ from Eq. (6), all the others are exactly the same in both methods and identical to those in Schulman et al. (2017). We made this choice within a clear and objective framework of comparison between the two methods. Thus, we have not optimized the rest of the hyperparameters for SAUNA, and its reported performance is not necessarily the best that could be obtained with more intensive tuning. From the results reported in Fig. 2, we see that our method surpasses all continuous control tasks. We also present in Table 1 the scores obtained for each task.

[1]Code is available here: https://github.com/iclr2020-submission/denoising-gradient-updates

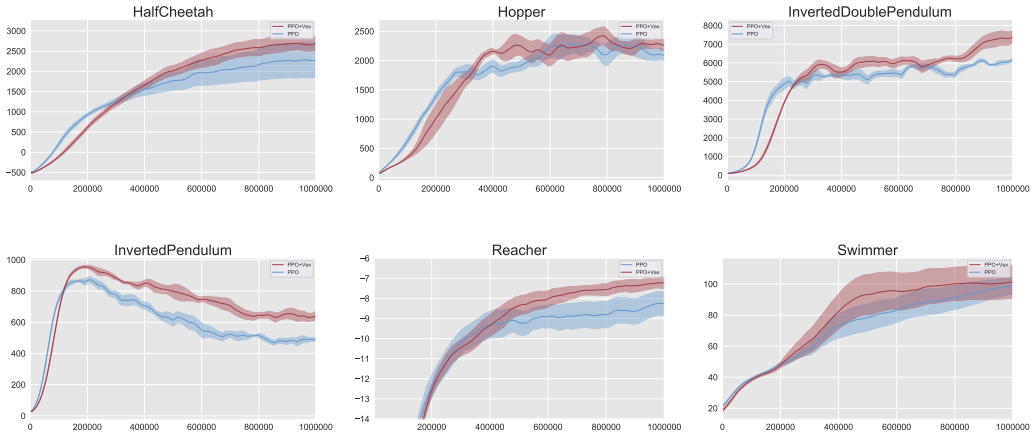


Figure 2: Comparison of SAUNA with PPO on 6 MuJoCo environments (10^6 timesteps, 6 different seeds). Red is our method PPO+Vex. Line: average performance. Shaded area: standard deviation.

Table 1: Average total reward of the last 100 episodes over 6 runs on the 6 MuJoCo environments. **Boldface** $mean \pm std$ indicate better mean performance.

Task	PPO	Ours
HalfCheetah	2277 ± 432	$\mathbf{2929 \pm 169}$
Hopper	2106 ± 133	$\mathbf{2250 \pm 73}$
InvertedDoublePendulum	6100 ± 143	$\mathbf{6893 \pm 350}$
InvertedPendulum	532 ± 19	$\mathbf{609 \pm 24}$
Reacher	-7.5 ± 0.8	$\mathbf{-7.2 \pm 0.3}$
Swimmer	99.5 ± 5.4	$\mathbf{100.8 \pm 10.4}$

4.2 The advantage of filtering out samples

We further study the impact of filtering out noisy samples by conducting additional experiments in predicting $\mathcal{V}^{ex}$ while omitting the filtering step before the gradient update: the *if* statement in Algorithm 1 is removed and all transitions are collected in τ. Indeed, the SAUNA algorithm could improve the agent's performance by simply training the shared network to optimize the variance explained head. Fig. 3 (full results are provided in Appendix D) demonstrates the positive effects of filtering out the samples.

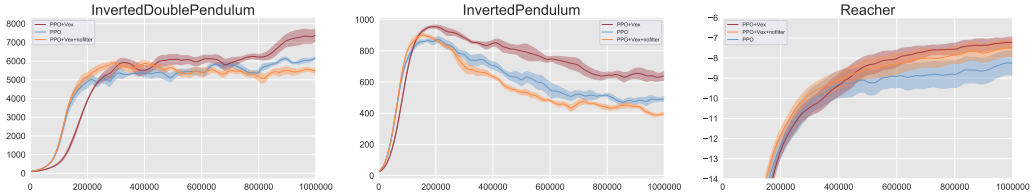
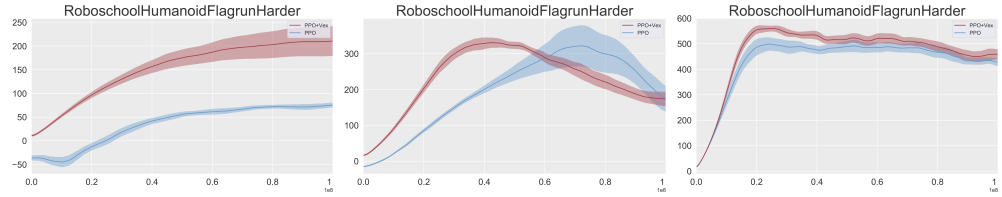


Figure 3: Comparison of SAUNA with PPO on 3 MuJoCo environments (10^6 timesteps, 6 different seeds). Red is our method PPO+Vex, Orange is PPO+Vex without the filtering out of noisy samples. Line: average performance. Shaded area: standard deviation.

The previous experiments have a threefold goal: (a) demonstrate the value of filtering the samples before the policy gradient update, (b) use the same configurations as for the reference method without additional hyperparameter tuning to support the validity of the method only, (c) evaluate SAUNA on a set of well-known continuous control environments.

4.3 ROBOSCHOOL

We then experiment with the more difficult, high-dimensional continuous domain environment of Roboschool: *RoboschoolHumanoidFlagrunHarder-v1*. The purpose of this task is to allow the agent to run towards a flag whose position varies randomly over time. It is continuously bombarded by white cubes that push it out of its path, and if it does not hold itself up it is left to fall.



(a) (b) (c)

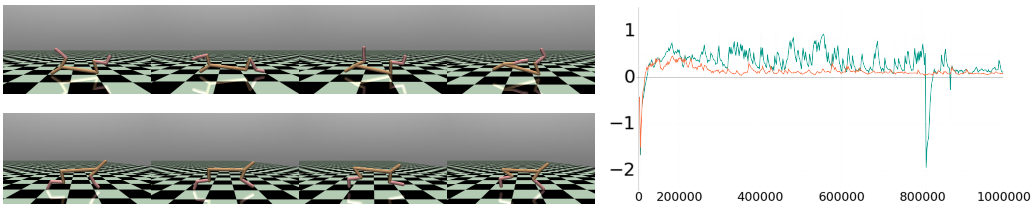
Figure 4: Comparison of SAUNA with PPO on the more challenging Roboschool environment (10^8 timesteps, 6 different seeds). Red is our method PPO+Vex. Line: average performance. Shaded area: standard deviation.

In Fig. 4a, the same fully-connected network as for the MuJoCo experiments (2 hidden layers each with 64 neurons) is used. In Fig. 4b, the network is composed of a larger 3 hidden layers with 512, 256 and 128 neurons. We trained those agents with 32 parallel actors. In both experiments, SAUNA performs better and faster at the beginning. Then, only when the policy and value functions benefit from a larger network, the gap closes, and our method does as well as the baseline. When resources are limited in terms of number of parameters, it seems natural that filtering out samples based on their predicted training impact allows to remove noise from the gradient update and accelerate learning.

Finally, we investigated further and conducted the same experiment with the larger network (3 hidden layers with 512, 256 and 128 neurons), but with 128 actors in parallel instead of 32. Results are reported in Fig. 4c: our method is still faster and achieves better performance than the baseline.

4.4 SAUNA: CASE STUDY

While studying *HalfCheetah-v2*, we observed that for a number of seeds, PPO was converging to a local minimum forcing the agent to move forward on its back. This is a well-known behavior (Lapan, 2018). However, we observed that SAUNA made it possible to leave from, or at least to avoid these local minima. Those particular deterministic environments can be generated reproducibly with specific seeds. This is illustrated in Fig. 5a where we can see still frames of two agents trained for 10^6 timesteps on identically seeded environments.



(a) (b)

Figure 5: (a) Example of a deterministic environment where PPO gets trapped in a local minimum (top row) while our method reaches a better optimum (bottom row). (b) $\mathcal{V}^{ex}$ score for PPO (orange) and SAUNA (green).

The behavior is entirely different for the agent trained with PPO and the agent trained with SAUNA denoising the policy gradient updates. If we look at the explained variance in Fig. 5b, we can see that the graphs differ quite interestingly. The orange agent seems to find very quickly a stable state in which it will put itself on its back while the green agent's variance explained varies much more.

This seems to allow the latter to explore more states than the former and finally find the fastest way forward. Through this particular example, we can observe that SAUNA is better able to explore interesting states while exploiting with confidence the value given to the states observed so far. The transitions of the white valley in Fig. 1 have been discarded in favor of the more critical transitions of the grey valley.

5 DISCUSSION

Intuitively, for the policy update, our method will only use qualitative samples that provide the agent with (a) reliable and exercised behavior (high $\mathcal{V}^{ex}$) and (b) challenging states from the point of view of correctly predicting their value (low $\mathcal{V}^{ex}$). The SAUNA algorithm keeps samples with high learning impact, rejecting other noisy samples from the gradient update.

5.1 DENOISING POLICY GRADIENT UPDATES AND THE POLICY GRADIENT THEOREM

Policy gradient algorithms are backed by the policy gradient theorem (Sutton et al., 2000):

$$\begin{aligned} \nabla_\theta L(\theta) &= \nabla_\theta \sum_{s \in S} d^\pi(s) \sum Q^\pi(s,a) \pi_\theta(a|s) \\ &\propto \sum_{s \in S} d^\pi(s) \sum_{a \in \mathcal{A}} Q^\pi(s,a) \nabla_\theta \pi_\theta(a|s). \end{aligned} \tag{11}$$

As long as the asymptotic stationary regime is not reached, it is not reasonable to assume the sampled states to be independent and identically distributed (i.i.d.). Hence, it seems intuitively better to ignore some of the samples for a certain period, to allow the most efficient use of information. One can understand SAUNA as making gradient updates more robust through denoising, especially when the update is low and the noise can be dominant. Besides, not taking all samples reduces the bias in the state distribution d^π. Therefore, it now seems more reasonable to consider the sampled states i.i.d., which we theoretically need for the policy gradient theorem.

5.2 IMPACT OF $\mathcal{V}^{ex}$ ON THE SHARED NETWORK PARAMETERS

The shared network predicts $\mathcal{V}^{ex}$ in conjunction with the value function and the policy. Therefore, as its parameters are updated through gradient descent, they converge to one of the objective function minima (hopefully, a global minimum). This parameter configuration integrates $\mathcal{V}^{ex}$, predicting how much the value function has fit the observed samples, or informally speaking how well the value function is doing for state s_t. This new head tends to lead the network to adjust predicting a quantity relevant for the task. Instead of using domain knowledge for the task, the method rather introduces problem knowledge by constraining the parameters directly.

6 CONCLUSION

We have introduced a new, lightweight and agnostic method readily applicable to any policy gradient method using a neural network as function approximation. Our variance explained criterion acts as a filter in-between the environment sampling and the policy update. SAUNA removes noise from the policy gradient updates to make learning more robust, ultimately leading to improved performance. We demonstrated its effectiveness on several standard benchmark environments and showcased that samples can be removed from the gradient update without breaking learning but can, on the opposite, improve it. We additionally studied the impact that learning from filtered samples has on both the exploitation of states visited by the agent and on the exploration of those that are little or unknown to it. Several open topics warrant further study. First, in this work, the influence of sample filtering on the distribution of states has been demonstrated to be beneficial but has not been theoretically studied. Second, numerous on- and off-policy methods could benefit from denoising policy gradient updates using variance explained or other measures, and we believe that the advantages of using SAUNA could be leveraged to improve a variety of other policy gradient algorithms. Finally, we find the effort (Cobbe et al., 2019; Zhang et al., 2018) to go further towards generalization in RL very promising, and we think SAUNA could be useful in these problems as a way to regularize policy gradient methods.

REFERENCES

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 1282–1289, 2019.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1406–1415, 2018.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pp. 2450–2462, 2018.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of London, 2003.

Tarald O Kvålseth. Cautionary Note about R^2. *The American Statistician*, 39(4):279–285, 1985.

Max Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing, 2018.

Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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, 2016.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

Derrick Nguyen and Bernard Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Advanced Neural Computers*, pp. 11–19, 1990.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks: the official journal of the International Neural Network Society*, 21(4):682–697, 2008.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471619779.

Anthony J. Robinson and F Fallside. Dynamic reinforcement driven error propagation networks with application to game playing. In *Conference of the Cognitive Science Society*, pp. 836–843, 1989.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

Jürgen Schmidhuber. Curious model-building control systems. In *IEEE International Joint Conference on Neural Networks*, pp. 1458–1463, 1991.

Jürgen Schmidhuber and Rudolf Huber. Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems*, 2(1/2):135–141, 1991.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1928–1937, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

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

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pp. 387–395, 2014.

David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484, 2016.

Richard S. Sutton and Andrew G. Barto. *Introduction to reinforcement learning*. Cambridge: MIT Press, 1998.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

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.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017.

Paul J. Werbos. Neural networks for control and system identification. In *IEEE Conference on Decision and Control,*, pp. 260–265, 1989.

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

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 *Advances in Neural Information Processing Systems*, pp. 5279–5288, 2017.

Amy Zhang, Nicolas Ballas, and Joëlle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

A ILLUSTRATION OF THE SAUNA ARCHITECTURE

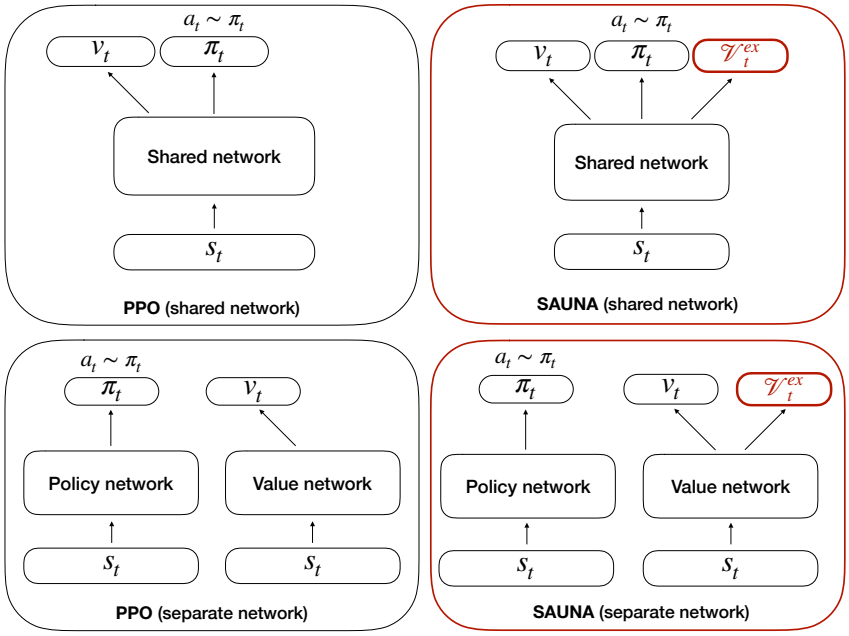


Figure 6: Network-agnostic variance explained head.

In Fig. 6 we rewrite $\mathcal{V}_\theta^{ex}(s_t)$ as $\mathcal{V}_t^{ex}$. On the right side of the figure is illustrated the $\mathcal{V}_t^{ex}$ head that is added to the shared or the separate network configurations. Note that even if $\mathcal{V}_\tau^{ex}$ is defined for a sampled trajectory τ, the model predicts its value at each state $s_t \in \tau$.

B HYPERPARAMETERS

Hyperparameter	Value
Horizon (T)	2048 (MuJoCo), 512 (Roboschool)
Adam stepsize	$3 \cdot 10^{-4}$
Nb. epochs	10 (MuJoCo), 15 (Roboschool)
Minibatch size	64 (MuJoCo), 4096 (Roboschool)
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.2
VF coef (c_1)	0.5
$\mathcal{V}^{ex}$ coef (c_2)	0.5
$\mathcal{V}^{ex}$ $threshold$	0.3

Table 2: Hyperparameters used both in PPO and SAUNA. The two last hyperparameters are only relevant for our method.

C IMPLEMENTATION DETAILS

Unless otherwise stated, the policy network used for MuJoCo and Roboschool tasks is a fully-connected multi-layer perceptron with 2 hidden layers of 64 units. For Atari, the network is shared between the policy and the value function and is the same as in Mnih et al. (2016). The architecture for the $\mathcal{V}^{ex}$ function head is the same as for the value function head.

D THE ADVANTAGE OF FILTERING OUT SAMPLES

In order to identify the effects of the training of the $\mathcal{V}^{ex}$ head and the filtering out of sample, we verify the hypothesis that filtering out noisy samples does improves the performance. To do so, in Section 4.2, we conduct experiments where the network predicts $\mathcal{V}^{ex}$ but where the noisy samples are not filtered out: the *if* statement in Algorithm 1 is removed and all transitions are collected in τ.

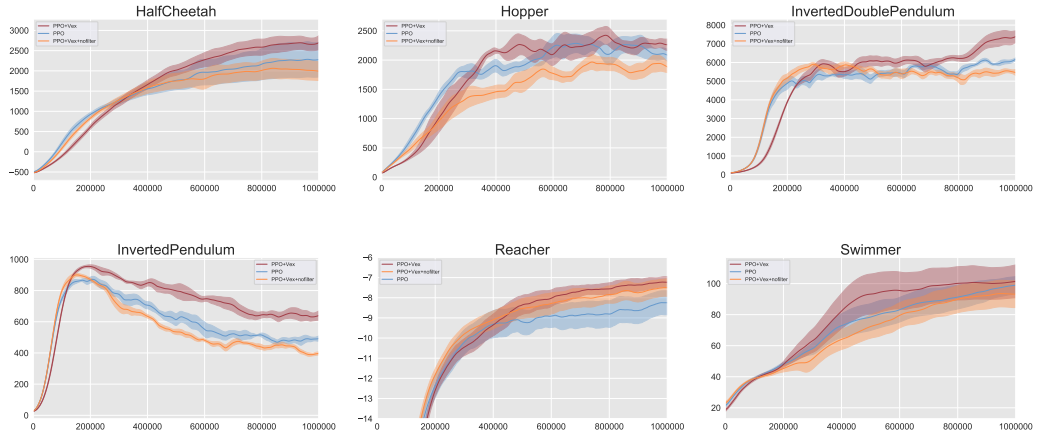


Figure 7: Comparison of our method with PPO on 6 MuJoCo environments (10^6 timesteps, 6 different seeds). Red is our method PPO+Vex, Orange is PPO+Vex without the filtering out of noisy samples. Line: average performance. Shaded area: standard deviation.

In Fig. 7, we see that when the noisy samples are not filtered out the performance is worst than the baseline, confirming the positive denoising impact of filtering out the variance-selected samples.

E ADDITIONAL EXPERIMENTS WITH NON-POSITIVE RESULTS

Atari domain. We tested our method on the Atari 2600 domain without observing any improvement in learning. By comparing the two algorithms where the method of filtering the samples is used or not, we could not observe any difference, as some tasks were better performed by one method and others by the other.

Mean of $\mathcal{V}^{ex}$. Although $\tilde{\mathcal{V}}^{ex}$, the median of $\mathcal{V}^{ex}$, is more expensive to calculate, we observe that it gives much better results than if we use its mean in Eq. (7). Using the median helps (Kvålseth, 1985) because the distribution of $\mathcal{V}^{ex}$ is not normal and includes outliers that will potentially produce misleading results.

Non-empirical $\mathcal{V}^{ex}$. We also experimented with using the real values of $\mathcal{V}^{ex}$ in Eq. (7) when calculating $\tilde{\mathcal{V}}^{ex}_{0:t-1}$, instead of the predicted ones. This has yielded less positive results, and it is likely that this is due to the difference between the predicted and actual values at the beginning of learning, which has the effect of distorting the ratio in Eq. (7).

Adjusting state count. In order to stay in line with the policy gradient theorem (Sutton et al., 2000), we have worked to adjust the distribution of states d^π to what it really is, since some states that the agent has visited are not included in the gradient update. We adjusted it using the ratio between the number of states visited and the actual number of transitions used in the gradient update, but this did not improve the learning, and instead, we observed a decrease in performance.

F CLIPPED SURROGATE OBJECTIVE DETAILS

In Eq. (3), we use the following standard definitions for the advantage function:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s), \tag{12}$$

where

$$Q^{\pi}\left(s_t, a_t\right) = \mathop{\mathbb{E}}_{\substack{s_{t+1:\infty} \\ a_{t+1:\infty}}}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right] \text{ and } V^{\pi}\left(s_t\right) = \mathop{\mathbb{E}}_{\substack{a_{t:\infty} \\ s_{t+1:\infty}}}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right]. \tag{13}$$

G AN ANALOGY WITH SAUNAS

Saunas originated in Northern Europe and are thought to date back to 7000 BC. Their use helps to release impurities [filtered out noisy samples] and improves the regeneration of cells [improved policy gradient updates]. Their temperatures could be fatal if not regulated by humidity [$\mathcal{V}^{ex}$ criterion].