Samples Are Useful? Not Always: denoising policy gradient updates using variance explained

Yannis Flet-Berliac, Philippe Preux

Samples are not all useful: Denoising policy gradient updates using variance

Yannis Flet-Berliac and Philippe Preux

Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL, F-59000 Lille, France
{yannis.flet-berliac,philippe.preux}@inria.fr

Abstract. Policy gradient algorithms in reinforcement learning rely on efficiently sampling an environment. Most sampling procedures are based solely on sampling the agent's policy. However, other measures made available through these algorithms could be used in order to improve the sampling prior to each policy update. Following this line of thoughts, we propose a method where a transition is used in the gradient update if it meets a particular criterion, and rejected otherwise. This criterion is the *fraction of variance explained* ($\mathcal{V}^{ex}$), a measure of the discrepancy between a model and actual samples. $\mathcal{V}^{ex}$ can be used to evaluate the impact each transition will have on the learning. This criterion refines sampling and improves the policy gradient algorithm. In this paper: (1) We introduce and explore $\mathcal{V}^{ex}$, the selection criterion used to improve the sampling procedure. (2) We conduct experiments across a variety of standard benchmark environments, including continuous control problems. Our results show better performance than if we did not use the $\mathcal{V}^{ex}$ criterion for the policy gradient update. (3) We investigate why $\mathcal{V}^{ex}$ gives a good evaluation for the selection of samples that will positively impact the learning. (4) We show how this criterion can be interpreted as a dynamic way to adjust the ratio between exploration and exploitation.

Keywords: Reinforcement learning · Policy gradient · Sampling.

1 Introduction

Learning to control agents in simulated environments has been a challenge for decades in reinforcement learning [21,41,28,26] and has recently led to a lot of research effort in this direction [19,12,3,39,8,34,6], notably in policy gradient methods [31,9,33,17,11,18]. Despite the definite progress made, the policy gradient algorithms still heavily suffer from sample efficiency [42,13,32,40].

In particular, many methods, on- and off-policy, are subject to use as much experience as possible in the most efficient way. We make the hypothesis that *not all experiences are good to consider*, at least not for use in the gradient update. In other words, perhaps trajectory simulation should indeed be as large and efficient as possible, but at the same time, some sampled transitions may have a negative impact on learning speed, causing a noisy gradient and hindering learning.

We will examine the impact of filtering the transitions, and how this affects the policy update on the learning performance. We exploit this for on-policy learning, primarily for its unbiasedness and stability compared to off-policy methods [20]. Moreover, on-policy is empirically known as being less sample efficient than off-policy learning; hence this issue emerged as an interesting research topic. However, our method can be applied to off-policy methods as well, and we leave this investigation open for future work.

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. A good final performance of the agent is conditioned on the sampling procedure. The method we introduce attempts to align the agent's aptitude in each environment (conditioned by a criterion, $\mathcal{V}^{ex}$) with the experiences that will affect its learning (the samples that will be used in the policy gradient update).

In section 3 we introduce our method. Then, in section 4 we investigate its performance across a variety of environments from the OpenAI Gym suite, Roboschool and the Atari domain. In the paper, we hypothesize that the agent's understanding of the environment can partially be measured through $\mathcal{V}^{ex}$. We explore this ability for the agent to evaluate its confidence or lack of knowledge about each environment state in section 4.3. In section 5, we consider the limitations of this criterion in the context of the policy gradient theorem and show how this can be a dynamic method for efficiently balancing exploration with exploitation.

The motivation for this research is to have a more efficient learning by taking into account states for which the agent predicts a critical learning impact. This method shifts from rewards-centered learning to learning taking into account the knowledge of the agent. The method takes advantage of $\mathcal{V}^{ex}$, allowing an alignment between the samples used to update the policy and the agent's progress.

2 Preliminaries

We consider a Markov Decision Process (MDP) which consists of a state space $\mathcal{S}$, an action space $\mathcal{A}$ and a 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 expected discounted reward be:

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

where $\gamma \in [0,1)$ is a discount factor [37] 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 [36]. The policy π_θ is generally modeled with a function parameterized with respect to θ. In deep reinforcement learning, the policy is parameterized by a neural network called the policy network.

2.1 Gradient-based method with clipped surrogate objective

We use Clipped-PPO [32], an on-policy gradient-based algorithm. This choice is mainly motivated by the fact that 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 (like TRPO [30] does), PPO formalizes the constraint as a penalty in the objective function.

In PPO, at each iteration, the new policy is updated with regards to the old policy:

$$\theta_{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} \underset{s,a\sim\pi_{\theta_k}}{\mathbb{E}} \left[J^{\mathrm{PPO}}\left(s, a, \theta_k, \theta\right) \right]. \tag{2}$$

The objective function is either clipped or subject to a KL-divergence term (log ratio between the old and new policy). In our case, we use the clipped version of PPO which objective function is:

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

where

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

By taking the minimum of the two terms in Eq. 3, the ratio $\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ is constrained to stay within a small interval around 1. The expected advantage function $A^{\pi_{\theta_{old}}}$ for the new policy is estimated by an old policy and then re-calibrated using the probability ratio between the new and the old policy.

2.2 Related Work

Our variance-directed policy gradient method incorporates three key ideas: i) policy and value function approximation with a neural network architecture combining or separating the actor and the critic, ii) an on-policy setting enabling more expected unbiasedness and stability than an off-policy formulation and iii) a variance-directed policy update to allow for better sampling and more efficient learning. Below, we consider previous work that builds on some of these approaches.

Actor-critic algorithms essentially use the value function to alternate between policy evaluation and policy improvement [37,1]. In order to update the actor, many methods adopt the on-policy formulation [25,7,18,32]. However, despite their important success, these methods suffer from sample complexity. In the literature, research has also been conducted in sampling prioritization. While [27] 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. [29] is related to our method in that it calculates the expected improvement in prediction error, but the objective is to maximize

the intrinsic reward through artificial curiosity while our method estimates the expected variance explained.

Lifelong learning literature [23,5,24] addresses the paradigm of how to reveal the ability to learn from continuous information flows when an agent's performance and represented state distribution change.

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 [31,16,17,10] that demonstrate how locomotion behavior and other skilled movements can emerge as the outcome of optimization problems.

3 Method

3.1 Variance explained

Let us define the criterion we use: the *fraction of variance explained* $\mathcal{V}^{ex}$. We compute the fraction of variance that the value function V_ϕ explains about the returns $\hat{R}$. It corresponds to the proportion of the variance in the dependent variable that is predictable from the independent variables. In statistics, it is also known as the coefficient of determination R^2 [14]. For clarity, we will not use this notation for the coefficient of determination, but we will refer to this criterion as:

$$\mathcal{V}^{ex} = 1 - \frac{\sum_{t=0}^{T} \left(\hat{R}_t - V_\phi(s_t)\right)^2}{\sum_{t=0}^{T} \left(\hat{R}_t - \overline{R}\right)^2} \tag{5}$$

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

Indeed, it should be noted that this criterion may be negative for non-linear models, indicating a severe lack of fit [14] of the corresponding function.

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. We will use this metric to select samples for which $\mathcal{V}^{ex}$ is high enough, to ensure the quality of the transition in terms of clairvoyance for the value function is good enough, but we also want to collect samples for which $\mathcal{V}^{ex}$ is low so that the value function has a wide margin for progress.

3.2 Variance explained applied to Clipped-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 tasks where the

input state is composed of pixels, we use a shared parameter network, primarily for faster convergence and computational reasons.

For shared parameters configurations, in addition to the Clipped-PPO objective function, an error term on the value estimation is added to the objective function as follows:

$$J^{\mathrm{PPO}'}(\theta) = \mathbb{E}\left[J^{\mathrm{PPO}}(\theta) - c_1\left(V_\theta(s) - \hat{R}\right)^2\right], \tag{6}$$

where c_1 is the coefficient for the squared-error loss of the value function. In addition, the $\mathcal{V}^{ex}$ value is approximated by the same shared network parameter to which we add a third head (the first two heads being the policy and the value function). Eq. 6 becomes:

$$J^{\mathrm{PPO}'}(\theta) = \mathbb{E}\left[J^{\mathrm{PPO}}(\theta) - c_1\left(V_\theta(s) - \hat{R}\right)^2 - c_2\left(\mathcal{V}_\theta^{ex} - \hat{\mathcal{V}}^{ex}\right)^2\right], \tag{7}$$

where c_2 is the coefficient for the squared-error loss of the variance explained function. In case the network is not shared between the policy and the value function, the head is added to the value function network.

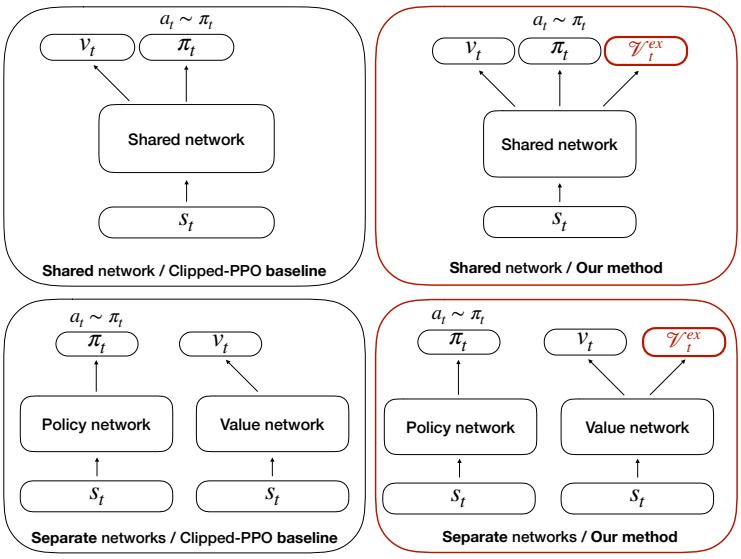

Fig. 1: Network-agnostic variance explained head.

Fig. 1 illustrates well how the head is embedded in the original architecture. By not changing the rest of the network, $\mathcal{V}^{ex}$ is very easy to incorporate into existing architectures without altering too much the complexity of the model.

3.3 Variance-directed update

At each time-step, if the state s_t complies with a condition subject to $\mathcal{V}^{ex}$, then the associated sampled transition is added to the current on-policy buffer (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}. We repeat the process until the episode is T steps long.

The condition is:

$$\frac{|\mathcal{V}_t^{ex}|}{|\tilde{\mathcal{V}}_{0:t-1}^{ex} + \epsilon_0|} \geq \textit{threshold}, \tag{8}$$

where $\tilde{\mathcal{V}}_{0:t-1}^{ex}$ is the median of $\mathcal{V}^{ex}$ between timesteps 0 and $t-1$ and ϵ_0[1] is to avoid division by zero. Fig. 2 illustrates the (grey) areas of accepted samples according to their $\mathcal{V}_t^{ex}$ and subject to the median of previous $\mathcal{V}^{ex}$ in the episode. We see the presence of a (white) area in which samples are excluded from the gradient update.

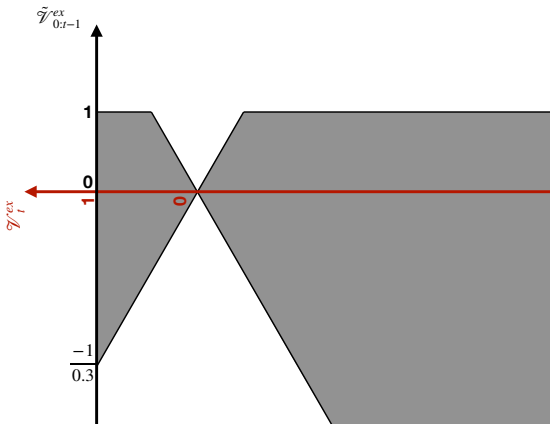

Fig. 2: Grey area samples are kept for the gradient update. Samples in the white area are discarded ($\textit{threshold} = 0.3$).

As depicted in the figure, this method selects the transitions that will impact the most the learning by using transitions where the explained variance $\mathcal{V}^{ex}$ is either high or low, but not in between. A high score means that the sample corresponds to a state which value (in the sense of value function) is well estimated, whereas a low score means that the value function does not fit well in this particular state. We believe that this helps to discard noisy samples, a hypothesis we will investigate.

Algorithm 1 illustrates how learning is achieved, in particular, the fitting of the $\mathcal{V}^{ex}$ function and how only collected samples are used for updates.

[1] ϵ_0 is set to 1^{-8}

Algorithm 1 Variance-directed update.

Initialise policy parameters θ_0 and value function parameters ϕ_0
Initialise $\mathcal{V}^{ex}$ function parameters ψ_0
for k = 0,1,2,... **do** ▷ For each update step
  Initialise *trajectory to capacity* T
  Sample the environment:
  while *size(trajectory)* $\leq T$ **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 $\frac{|\mathcal{V}_t^{ex}|}{|\bar{\mathcal{V}}_{0:t-1}^{ex}+\epsilon_0|} \geq$ *threshold* **then**
      collect transition $(s_t, a_t, r_t, v_t, s_{t+1}, \mathcal{V}_t^{ex})$ in *trajectory*
    else
      continue without collecting the transition
  Update policy with trajectory:
  $\theta_{k+1} \leftarrow \underset{\theta}{\operatorname{argmax}} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g\left(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)\right)\right)$
  Update value function with trajectory:
  $\phi_{k+1} \leftarrow \underset{\phi}{\operatorname{argmin}} \sum_{t=0}^{T} \left(V_{\phi_k}(s_t) - \hat{R}_t\right)^2$
  Update $\mathcal{V}^{ex}$ *function with trajectory*:
  $\psi_{k+1} \leftarrow \underset{\psi}{\operatorname{argmin}} \sum_{t=0}^{T} \left(\mathcal{V}_{\psi_k}^{ex}(s_t) - \hat{\mathcal{V}}_t^{ex}\right)^2$

4 Experiments

Unless otherwise stated, all curves correspond to the average of five runs with different seeds, and shaded areas are standard deviations. For ease of use and sharing, we have forked the original *baselines* repository from OpenAI and modified the code to incorporate our method [2]. The complete list of hyperparameters and details of our implementation are given in Appendix A and B respectively.

4.1 Comparison in the continuous domain: MuJoCo

We begin by comparing our variance-directed method (PPO_Vex in red) with its natural baseline Clipped-PPO introduced in section 2.1 (PPO in blue). We used 7 simulated robotic tasks [3] from OpenAI Gym [2], which use the MuJoCo [38] physics engine. Except for the two hyperparameters required by our method, namely *threshold* = 0.3 from algorithm 1 and $c_2 = 0.5$ from Eq. 7, all the others are exactly the same in both methods and identical to those in [32]. We made this choice within a clear and objective framework of comparison between the two methods. Thus, we have not optimized all hyperparameters for our method, and its reported performance is not necessarily the best that can be obtained with more intensive tuning, but it still exceeds the baseline.

[2] Code is available here: https://github.com/yfletberliac/denoising-gradient-updates
[3] 'HalfCheetah', 'Hopper', 'InvertedDoublePendulum', 'InvertedPendulum', 'Reacher', 'Swimmer' and 'Walker2d' (all '-v2')

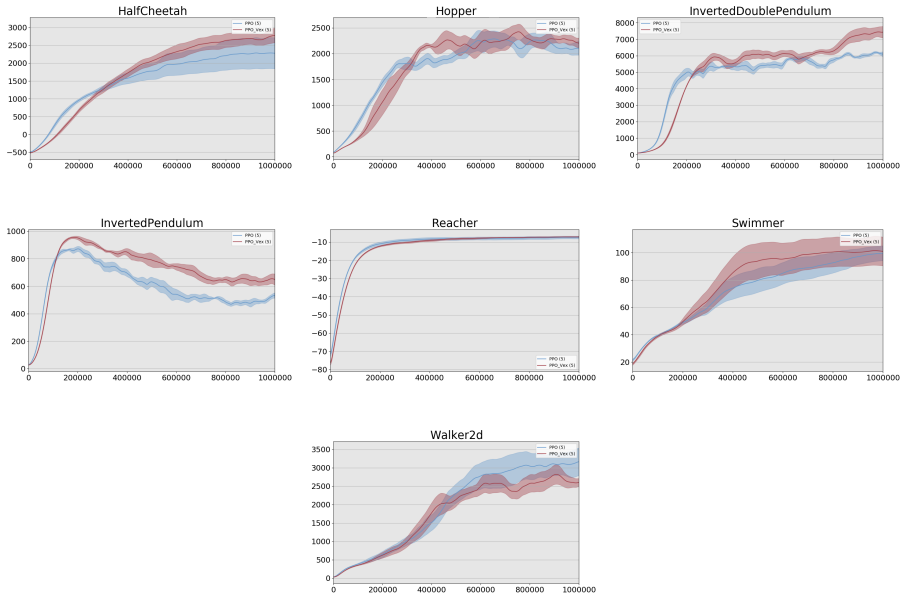

Fig. 3: Comparison of our method with Clipped-PPO on 7 MuJoCo environments (1M timesteps, 5 different seeds). Red is our method PPO_Vex. The line is the average performance, while the shaded area represents its standard deviation.

From the results reported in Fig. 6, we see that our method surpasses all continuous control tasks except 'Walker2d-v2'. We also present in table 1 the scores obtained for each task.

Table 1: Average total reward of the last 100 episodes over 5 runs on the 7 MuJoCo environments. **Boldface** $mean \pm std$ indicate better 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	$\mathbf{100.8 \pm 10.4}$
Walker2d	$\mathbf{3161 \pm 374}$	2593 ± 105

These experiments had a threefold goal: i) demonstrate the value of filtering the samples before the on-policy gradient update, ii) use the same configurations as for the reference method without additional hyperparameter tuning to support

the validity of the method only, iii) evaluate our method on a set of well-known continuous control environments.

4.2 Roboschool

We then experimented with on 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 left to fall and is continuously bombarded by white cubes that push it out of its path.

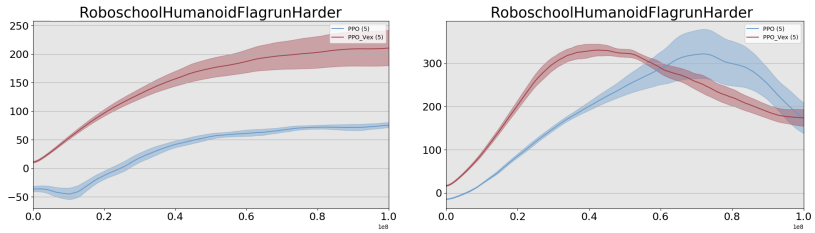

Fig. 4: Comparison of our method with Clipped-PPO on the more challenging Roboschool environment (100M timesteps, 5 different seeds). Red is our method PPO_Vex. The line is the average performance, while the shaded area represents its standard deviation. Left: 2-layer network. Right: 3-layer and larger network.

In Fig. 4, on the left, the same fully-connected network as for MuJoCo experiments (2 hidden layers each with 64 neurons) is used. On the right, the network is composed of 3 hidden layers with 512, 256 and 128 neurons. We trained those agents with 32 parallel actors. Interestingly in both experiments, our method performs better and faster at the beginning, but when the policy and value functions benefit from a larger network, the gap closes, and our contribution does as well as its baseline. When resources are limited parameter-wise, one can

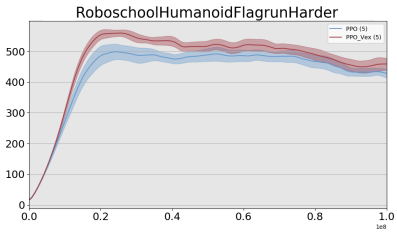

Fig. 5: 3-layer and larger network, now with 128 parallel actors instead of 32.

think of it as natural that filtering samples based on their predicted training impact seems to remove noise from the gradient update and accelerate learning.

Finally, we wanted to investigate further and conducted the same experiment as with the larger network (3 hidden layers with 512, 256 and 128 neurons), but with 128 actors in parallel instead of 32. The total number of timesteps is unchanged and is still equal to 100M. Results are reported in Fig. 5: our method is still faster and achieves better performance than the baseline.

4.3 Variance explained $\mathcal{V}^{ex}$: case study

While studying 'HalfCheetah-v2', we observed that for quite a few seeded environments, PPO was converging to a local minimum, leading the agent to move forward on its back. This is a well-known behavior [15]. However, we observed that our method 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. 6 where we can look at still frames of two agents trained for 1M timesteps on identically seeded environments.

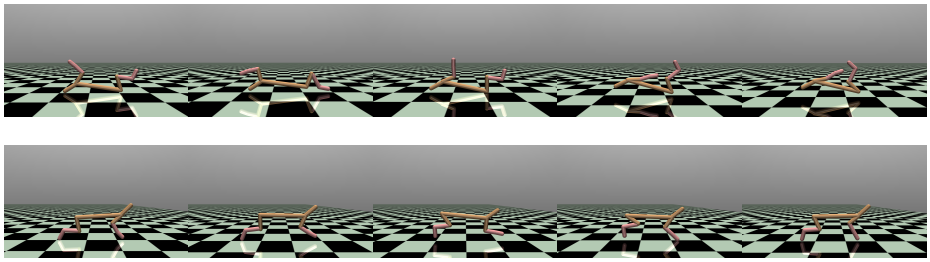

Fig. 6: Example of a deterministic environment where PPO converges towards a local minimum (top row) while our method does not (bottom row).

The behavior is entirely different for the agent trained with PPO and the agent trained with PPO combined with our method denoising the policy gradient updates. If we look at the explained variance in Fig. 7, we can see that the graphs differ in a very interesting way. 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, probably allowing it to explore more states than the former and eventually find the fastest way to move forward. Through this particular example, we can observe that PPO_Vex is better able to explore interesting states while exploiting with confidence the understanding of the states observed so far. The transitions of the white valley in Fig. 2 have been masked in favor of the more critical transitions of the grey valley.

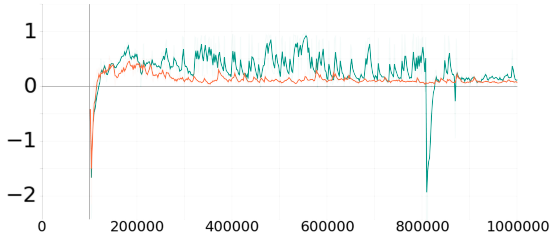

Fig. 7: $\mathcal{V}^{ex}$ comparison for Clipped-PPO (orange) and our method (green).

4.4 Worth mentioning 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 parameters where sampling filtering 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. 8. This is probably because the distribution of $\mathcal{V}^{ex}$ is not normal and includes outliers that will potentially produce misleading results [14].

Non-empirical $\mathcal{V}^{ex}$. We also experimented with using the real values of $\mathcal{V}^{ex}$ in Eq. 8 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. 8.

Adjusting state count. In order to stay in line with the policy gradient theorem [35], 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 we observed a decrease in performance.

5 Discussion

Intuitively, our method will consider for the policy update a combination of qualitative samples that provide the agent with i) solid and exercised behavior (which states yield a large $\mathcal{V}^{ex}$) and ii) samples that still represent a challenge for the agent from the point of view of understanding their value (whose states give a low $\mathcal{V}^{ex}$). The variance-directed algorithm catches samples with high learning impact, keeping the other noisy samples away from the gradient update.

5.1 What does this mean for the advantage term $A^{\pi_{\theta_{old}}}$?

Each episode is $T_{horizon}$ steps long. In an episode, there will be eluded (but executed) transitions. Naively, this means that those steps do not have a sufficient impact on learning according to the $\mathcal{V}_t^{ex}$ head. Hence, in line with the training, the transitions associated with those bad learning impact scores should not be included in the gradient update of the agent at timestep t. We can observe that as the head learns how to predict the expected $\mathcal{V}^{ex}$, the transitions which are taken into account for the gradient updates coincide better with the current experience of the agent, leading to better learning performances.

The advantage term is the difference between the total discounted sum of rewards from state s executing action a and the value of the state s.

$$A(s,a) = Q(s,a) - V(s) \tag{9}$$

This means that states where the variance explained $\mathcal{V}^{ex}$ is within a margin centered on 0 corresponding to the white valley in 2 will not be considered when computing the advantage function. The episode used in the update will have "holes" in it, allowing the algorithm to prioritize samples that are either well explained by the value function or poorly explained, but not vaguely in-between. We believe that the rejection of these samples has a denoising effect on the gradient update and a positive impact on the learning capacity.

5.2 What does this mean for the shared network parameters?

We are forcing the neural network to predict the variance explained $\mathcal{V}^{ex}$ in conjunction with the value function and the policy (cf. Fig. 1). Therefore, as its parameters are updated through gradient descent, they converge to one of the objective function minima (hopefully, global minimum). This state partially integrates $\hat{\mathcal{V}}^{ex}$, 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 forwards the network to adjust seeking a value relevant for the task. This is not giving domain knowledge to the task, but rather introducing problem knowledge by constraining the architecture of the network directly.

5.3 Denoising policy gradient updates and policy gradient theorem

Policy gradient algorithms are backed by the policy gradient theorem [35]:

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \sum_{s \in S} d^\pi(s) \sum Q^\pi(s,a) \pi_\theta(a|s) \\ &\propto \sum_{s \in S} d^\pi(s) \sum_{a \in \mathcal{A}} Q^\pi(s,a) \nabla_\theta \pi_\theta(a|s) \end{aligned} \tag{10}$$

Our method seems to make gradient updates more robust through denoising, especially when the update is low and the noise can be dominant. In addition,

not taking all samples reduces the bias in the state distribution d^π since as long as the asymptotic stationary regime is not reached it seems intuitively better to ignore some samples for a certain period of time, so as to allow the most efficient use of information. This way, it is more reasonable to consider the sampled states independent and identically distributed, which we theoretically need in order to use the policy gradient theorem.

6 Conclusion

We have introduced a new, lightweight and agnostic method readily applicable to any policy gradient method using a network architecture. Our variance explained criterion acts as a filter in-between the environment sampling and the policy update. The method 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 damaging learning but can, on the contrary, improve it. Finally, we studied the impact that learning from filtered samples has on both the exploitation of the states which the agent visits and on the exploration of those that are little or unknown to it. Many open questions warrant further study. First, although positive, the impact of sample filtering on the distribution of states is not formally understood. Second, there are numerous on- and off-policy methods that could benefit from denoising policy gradient updates using variance or other measures, we believe that the advantages of transition filtering before policy updates can be leveraged to improve a variety of policy gradient algorithms. Finally, we find the effort [22,4,43] to go further towards generalization in RL very interesting. We think our method could yield promising results in these problems.

7 Acknowledgements

The authors would like to acknowledge the support of Inria, and SequeL for providing a great environment for this research. The authors also acknowledge the support from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. This work was supported by the French Ministry of Higher Education and Research.

References

1. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics **13**(5), 834–846 (1983)
2. Brockman, G., et al.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
3. Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. arXiv preprint arXiv:1810.12894 (2018)

4. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. arXiv preprint arXiv:1812.02341 (2018)
5. Ditzler, G., Roveri, M., Alippi, C., Polikar, R.: Learning in nonstationary environments: A survey. IEEE Computational Intelligence Magazine **10**(4), 12–25 (2015)
6. Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al.: Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In: International Conference on Machine Learning. pp. 1406–1415 (2018)
7. Gruslys, A., Azar, M.G., Bellemare, M.G., Munos, R.: The reactor: A sample-efficient actor-critic architecture. In: International Conference on Learning Representations (2017)
8. Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution (2018)
9. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning. pp. 1856–1865 (2018)
10. Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., Tassa, Y.: Learning continuous control policies by stochastic value gradients. In: Advances in Neural Information Processing Systems. pp. 2944–2952 (2015)
11. Heess, N., et al.: Emergence of locomotion behaviours in rich environments. arXiv preprint arXiv:1707.02286 (2017)
12. Hessel, M., et al.: Rainbow: Combining improvements in deep reinforcement learning. In: AAAI Conference on Artificial Intelligence (2018)
13. Kakade, S.: On the sample complexity of reinforcement learning. Ph.D. thesis, University of London (2003)
14. Kvålseth, T.O.: Cautionary Note about R^2. The American Statistician **39**(4), 279–285 (1985)
15. Lapan, M.: 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)
16. Levine, S., Abbeel, P.: Learning neural network policies with guided policy search under unknown dynamics. In: Advances in Neural Information Processing Systems. pp. 1071–1079 (2014)
17. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
18. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning. pp. 1928–1937 (2016)
19. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
20. Nachum, O., Norouzi, M., Xu, K., Schuurmans, D.: Bridging the gap between value and policy based reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 2775–2785 (2017)
21. Nguyen, D., Widrow, B.: The truck backer-upper: An example of self-learning in neural networks. In: Advanced Neural Computers. pp. 11–19 (1990)
22. Packer, C., Gao, K., Kos, J., Krhenbhl, P., Koltun, V., Song, D.: Assessing generalization in deep reinforcement learning (2018)
23. Parisi, G.I., Tani, J., Weber, C., Wermter, S.: Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization. Frontiers in Neurorobotics **12**, 78 (2018)

24. Parisi, G., Kemker, R., Part, J., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural networks: the official journal of the International Neural Network Society **113**, 54–71 (2019)
25. Peters, J., Schaal, S.: Reinforcement learning of motor skills with policy gradients. Neural networks: the official journal of the International Neural Network Society **21**(4), 682–697 (2008)
26. Robinson, T., Fallside, F.: Dynamic reinforcement driven error propagation networks with application to game playing. In: Conference of the Cognitive Science Society. pp. 836–843 (1989)
27. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. In: International Conference on Learning Representations (2016)
28. Schmidhuber, J., Huber, R.: Learning to generate artificial fovea trajectories for target detection. International Journal of Neural Systems **2**(1/2), 135–141 (1991)
29. Schmidhuber, J.: Curious model-building control systems. In: IEEE International Joint Conference on Neural Networks. pp. 1458–1463 (1991)
30. Schulman, J., Levine, S., Abbeel, P., Jordan, M.I., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning. pp. 1928–1937 (2015)
31. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438 (2015)
32. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
33. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: International Conference on Machine Learning. pp. 387–395 (2014)
34. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**, 484 (2016)
35. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. Cambridge: MIT Press (2018)
36. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems. pp. 1057–1063 (2000)
37. Sutton, R.S., et al.: Introduction to reinforcement learning. Cambridge: MIT Press (1998)
38. Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5026–5033 (2012)
39. Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., Kavukcuoglu, K.: FeUdal networks for hierarchical reinforcement learning. In: International Conference on Machine Learning. pp. 3540–3549 (2017)
40. Wang, Z., et al.: Sample efficient actor-critic with experience replay. arXiv preprint arXiv:1611.01224 (2016)
41. Werbos, P.J.: Neural networks for control and system identification. In: IEEE Conference on Decision and Control,. pp. 260–265 (1989)
42. Wu, Y., Mansimov, E., Grosse, R.B., Liao, S., Ba, J.: Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In: Advances in Neural Information Processing Systems. pp. 5279–5288 (2017)
43. Zhang, A., Ballas, N., Pineau, J.: A dissection of overfitting and generalization in continuous reinforcement learning. arXiv preprint arXiv:1806.07937 (2018)

A Hyperparameters

Hyperparameter	Value
Horizon ($T_{horizon}$)	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 Clipped-PPO and in our method. The two last hyperparameters are only relevant for our method.

B Implementation details

Unless otherwise stated, the policy network used for MuJoCo and Roboschool tasks is a fully-connected multi-layer perceptron with two hidden layers of 64 units. For Atari, the network is shared between the policy and the value function and is the same as in [18]. The architecture for the $\mathcal{V}^{ex}$ function head is the same as for the value function head.

C Clipped surrogate objective: more details

In Eq. 3, we used the following standard definitions for the advantage function:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s), \tag{11}$$

where

$$Q^{\pi}\left(s_t, a_t\right) = \mathop{\mathbb{E}}_{\substack{s_{t+1:\infty} \\ a_{t+1:\infty}}} \left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right] \text{ and } V^{\pi}\left(s_t\right) = \mathop{\mathbb{E}}_{\substack{a_{t:\infty} \\ s_{t+1:\infty}}} \left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right]. \tag{12}$$