A Functional Clipping Approach for Policy Optimization Algorithms

WANGSHU ZHU AND ANDRE ROSENDO, (Member, IEEE)
Living Machines Laboratories, Department of Computer Science, School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
Corresponding author: Wangshu Zhu (zhuwsh@shanghaitech.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Project 61850410527, and in part by the Shanghai Young Oriental Scholars under Project 0830000081.

ABSTRACT Proximal policy optimization (PPO) has yielded state-of-the-art results in policy search, a subfield of reinforcement learning, with one of its key points being the use of a surrogate objective function to restrict the step size at each policy update. Although such restriction is helpful, the algorithm still suffers from performance instability and optimization inefficiency from the sudden flattening of the curve. To address this issue we present a novel functional clipping policy optimization algorithm, named Proximal Policy Optimization Smoothed Algorithm (PPOS), and its critical improvement is the use of a functional clipping method instead of a flat clipping method. We compare our approach with PPO and PPORB, which adopts a rollback clipping method, and prove that our approach can conduct more accurate updates than other PPO methods. We show that it outperforms the latest PPO variants on both performance and stability in challenging continuous control tasks. Moreover, we provide an instructive guideline for tuning the main hyperparameter in our algorithm.

INDEX TERMS Machine learning, robot control, deep reinforcement learning, policy search algorithm.

I. INTRODUCTION
Reinforcement learning, especially deep model-free reinforcement learning, has achieved great progress in recent years. The application of reinforcement learning ranges from video games [9], [14], board games [21] to robotics [12], [13], and challenging control tasks [5], [19]. Within these approaches, Policy gradient (PG) methods are widely used in model-free policy search algorithms, contributing significantly to the fields of robotics and high-dimensional control. PG methods update the policy with a step estimator of the gradient of the expected return and gradually converge to the optimal policy [16]. One of the most significant problems of PG-based methods is to determine the suitable step size for the policy update while an improper step size will either prolong the policy searching process or even degrade the policy, since the sampling data highly depends on the current policy. As a result, the trade-off between learning speed and robustness is a challenging problem for PG methods.

The trust region policy optimization (TRPO) method [18] proposes an objective function (also called the “surrogate” objective) and aims to maximize the objective function while subjecting to a constraint on the size of the policy update by KL divergence. However, the subjecting item is computationally inefficient and difficult to scale up for high-dimensional problems when extending to complex network architectures. To address this problem, the proximal policy Optimization (PPO), which adopts a clipping mechanism on the probability of likelihood ratio, was introduced [20]. This clipping mechanism significantly reduces the complexity of the optimizing process. To deal with learning robustness PPO tries to remove the incentive for pushing the policy away from the old one when the ratio is out of a clipping range, being more efficient for both implementation and computation than the trust region-based TRPO.

Despite the success of PPO it also has some flaws in restricting the update step size, which means that PPO cannot truly bind the likelihood ratio within the clipping range [25]. Aiming towards a better step size update, the proximal policy optimization with rollback (PPORB) applied a rollback operation on the clipping function to inhibit the policy from
being pushed away during training [26]. PPORB adopts a straight downward-slope function instead of the original flat function when the ratio is out of the clipping range, which hinders the natural incentive from PPO to seek a large policy update. However, this solution introduces new problems, such as when the ratio is extremely big and the clipped ratio will shoot to infinity or negative infinity, and thus contrast with its original aim. Additionally, the inner workings of PPORB’s hyperparameter $\alpha$, the slope of the rollback function, is still abstruse and poorly understood because PPORB only provides the empirical values for specific experiments instead of a tuning method or function.

Inspired by the insights above, we propose a novel PPO clipping method, named Proximal Policy Optimization Smoothed algorithm (PPOS) which combines the strengths of both PPO and PPORB. We apply a functional clipping mechanism to prevent large policy updates while keeping the convergence of the clipped ratio. We compare PPOS with PPO and PPORB in high-dimensional control environments. We also analyze the hyperparameter here introduced in relation to the dimensions of five problems and provide a useful guideline for readers to orient their hyperparametric choices according to the dimension of their problems.

II. PRELIMINARIES

This section discusses the background information needed to understand our policy search approach. We first describe the reinforcement learning (RL) framework, which is based on the Markov decision process (MDP). Then, we give an overview of policy search methods. We continue by introducing trust-region methods and actor-critic methods. Finally, we delve into the Proximal Policy Optimization (PPO) algorithm, which is currently one of the state-of-the-art algorithms in policy search.

A RL procedure consists of an agent and an environment, where the agent can interact with the environment and gain rewards during this interaction. We consider RL as a process running by a Markov decision process (MDP). An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, p, \rho_0, r, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^d_s$ is a finite set of states of the agent and $\mathcal{A} \subseteq \mathbb{R}^d_a$ is a finite set of actions which can be conducted by the agent. $p (s_{t+1} | s_t, a_t)$ denotes the probability of transfer from state $s_t$ to $s_{t+1}$ given the action conducted by the agent at time step $t$. $p (s_{t+1} | s_t, a_t)$ is usually unknown in high-dimensional control tasks, but we can sample from $p (s_{t+1} | s_t, a_t)$ in simulation or from a physical system in the real world. $\rho_0$ denotes the initial state distribution of the agent, and $r(s_t, a_t)$ denotes the valued reward in state $s_t$ when the agent executes action $a_t$. The last $\gamma \in (0, 1)$ is the discount factor. A policy in RL is defined as the map (or function) from the state space to the action space. The goal of RL is to find a policy, which was first introduced in the relative entropy policy search (REPS) algorithm [17]. Many variants of TRO methods were introduced after the REPS algorithm (e.g. [1], [2], [3]). Most of these TRO methods use a bound of the KL-divergence of the policy update which prevents the policy update from being unstable while keeping the update suitable. Trust region policy optimization (TRPO) [18] adopts this bound to optimize the policy. The surrogate objective of TRPO is maximized while being subjected to a constraint on the KL-divergence between the old and current policy.

\[
\max \mathbb{E}_{\pi_{\text{old}}, \pi_{\text{old}}(a | s)} \left[ \frac{\pi(a | s)}{\pi_{\text{old}}(a | s)} A_{\text{old}}(s, a) \right] \tag{4}
\]

subject to

\[
\mathbb{E}_{\pi} [\text{KL} (\pi \parallel \pi_{\text{old}}(\cdot | s), \pi_{\text{old}}(\cdot | s)) ] \leq \delta \tag{5}
\]

where $\delta$ is a constant hyper-parameter and “$\parallel$” means any action $a$. In general, when $\delta$ is small enough, the update is

\[
Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots} \left[ \sum_{t=0}^T y^t r(s_t, a_t) \right] \tag{2}
\]

in which $\rho_\pi(s)$ is the state distribution under $\pi$, i.e., the probability of visiting state $s$ under $\pi$. $Q^\pi(s_t, a_t)$ is called state-action value function which evaluate the state-action pair. $s_{t+1} \sim p (s_{t+1} | s_t, a_t)$. $a_t \sim \pi (a_t | s_t)$ and $T$ is the terminal time step. We could also compute the value function $V^\pi (s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots} \sum_{t=0}^T y^t r(s_t, a_t)$ which is the evaluation of the state $s_t$, and advantage function $A^\pi (s_t, a_t) = Q^\pi (s_t, a_t) - \hat{V}^\pi (s_t)$ which indicates how reasonable the chosen action $a_t$ is given the state $s_t$.

In policy search, the goal is to find a policy $\pi$ that maximized Eq.1 In recent work, the policy usually adopts the neural network structure (e.g. [14], [15] and [24]) which simulates the network of neuron of human-being to some extent. A policy can be improved by update the policy by the following surrogate performance objective which is called policy gradients methods [22],

\[
L_{PG}^{\pi_{\text{old}}} (\pi) = \mathbb{E}_{s, a} \left[ r_{s,a}(\pi) A_{\text{old}}(s, a) \right] + \eta (\pi_{\text{old}}) \tag{3}
\]
valid since the difference between the new policy and the old policy is not substantial.

**B. PROXIMAL POLICY OPTIMIZATION**

TRPO has been theoretically instantiated as capable of ensuring monotonic performance improvements of the policy. Nonetheless, due to the KL-divergence constraint TRPO is computationally inefficient and difficult to scale up for high-dimensional problems. In order to simplify the implementation and computational requirements, the PPO algorithm [20] was introduced. PPO restricts the policy update by a clipping function \( F \) on the ratio \( r_{s,a}(\pi) \) directly,

\[
L_{\text{CLIP}}(\pi) = \mathbb{E} \left[ \min(\frac{r_{s,a}(\pi)}{\epsilon}, \epsilon) A^{\pi_{\text{old}}}(s, a) \right] \tag{6}
\]

In PPO, \( F \) is defined as,

\[
F_{\text{PPO}}(r_{s,a}(\pi), \epsilon, \alpha) = \begin{cases} 
1 - \epsilon & r_{s,a}(\pi) \leq 1 - \epsilon \\
1 + \epsilon & r_{s,a}(\pi) \geq 1 + \epsilon \\
r_{s,a}(\pi) & \text{otherwise}
\end{cases} \tag{7}
\]

where \( 1 - \epsilon \) and \( 1 + \epsilon \) are called the lower and upper clipping range, respectively, and \( \epsilon \in (0, 1) \) is a hyper parameter. The minimization operation in Eq. (6) is to keep the gradient when the \( r_{s,a} \) is out of the clipping range. The illustration of the clipping mechanism as the red dash line can be seem in Fig.1.

Both TRPO and PPO adopt the actor-critic method [10], in which the Value function \( Q(\pi) \) is estimated by a parameterized function approximator \( Q(s, a; \omega) \), where the vector \( \omega \) is the parameter of the approximator e.g. a linear model or usually a neural network. This estimator is called the critic which can have a lower variance than traditional Monte Carlo estimators [27]. The critic’s estimate is used to optimize the policy \( \pi(a \mid s; \theta) \) which is also called the actor parameterized by \( \theta \). The actor is usually represented by a neural network. After that, the policy parameters can be updated by the gradient of surrogate objective Eq. (6) as,

\[
\theta_{t+1} = \theta_t + \beta \nabla L_{\text{CLIP}}(\theta_t) \tag{8}
\]

where \( \beta \) is the step size and \( \nabla L_{\text{CLIP}}(\theta_t) \) denote the gradient of \( L_{\text{CLIP}} \) at \( \theta_t \).

**C. PPO WITH ROLLBACK**

PPO has made progress in different control tasks, especially with high-dimensional observation and action space, but it also has raised concerns of researchers about whether this clipping mechanism can truly restrict the policy ([8], [25]). In reality, PPO is incapable of binding the likelihood ratio within the clipping range because its clipping mechanism still pushes the ratio out of its range, rendering the policy out of the bounds.

To address this problem, PPORB [26] proposed a rollback operation on the clipping function attempting to remove the incentive of moving far away, which is defined as

\[
F_{\text{PPORB}}(r_{s,a}(\pi), \epsilon, \alpha) = \begin{cases} 
-\epsilon r_{s,a}(\pi) + (1 + \alpha)(1 - \epsilon) & r_{s,a}(\pi) \leq 1 - \epsilon \\
-\epsilon r_{s,a}(\pi) + (1 + \alpha)(1 + \epsilon) & r_{s,a}(\pi) \geq 1 + \epsilon \\
r_{s,a}(\pi) & \text{otherwise}
\end{cases} \tag{9}
\]

where \( \alpha \) is a hyperparameter to decide the force of the rollback operation. When the ratio \( r_{s,a} \) is out of the clipping range, it will make negative effects on the output instead of keeping the value as a constant. PPORB has made vast progress from the original PPO algorithm, but at the same time, it leaves space for improvement. Firstly, the rollback operation cannot converge when the ratio is extremely large: In high dimensional tasks, it will make the policy search sway around the optimal policy. Secondly, the rollback operation introduces an extra hyperparameter which can be only tuned by experience of the same problem. However, the hyperparametric possibilities result in significantly different results, with the values for this parameter ranging from 0.02 to 0.3.

**III. METHOD**

In this section, we first describe our reinforcement learning method, and then compare it with other methods of proximal policy optimization algorithms. Finally, we discuss the stability and strength of our method in this paper.
As discussed in the section Preliminaries II-C, while the rollback operation can, to some extent, restrict the likelihood ratio, it also leaves unsolved problems: the convergence property and the choice of hyperparameters. We deploy our method on the structure of PPO with a general form for sample \((s, a)\),

\[
L_{s,a}(\pi) = \min \left( r_{s,a}(\pi) A^{\text{old}}(s, a), \mathcal{F}(r_{s,a}(\pi), \cdot) A^{\text{old}}(s, a) \right)
\]

where \(\mathcal{F}\) is a clipping function which attempts to restrict the policy update and ‘.’ means any hyperparameters of \(\mathcal{F}\).

In PPO, \(\mathcal{F}\) is a ratio-based clipping function \(\mathcal{F}^{\text{PPO}}\) as Eq. (7), which will be flat whenever out of the clipping range. In PPORB, \(\mathcal{F}^{\text{PPORB}}\) has the rollback operation as Eq. (9) instead of keeping it flat. We modify this function to promote the performance and stability of restricting the likelihood ratio by a functional clipping method, which is defined as

\[
\mathcal{F}^{\text{PPOS}}(r_{s,a}(\pi), \epsilon, \alpha) =
\begin{cases}
-\alpha \tanh(r_{s,a}(\pi) - 1) + 1 + \epsilon + \alpha \tanh(\epsilon) & r_{s,a}(\pi) \leq 1 - \epsilon \\
-\alpha \tanh(r_{s,a}(\pi) - 1) + 1 - \epsilon - \alpha \tanh(\epsilon) & r_{s,a}(\pi) \geq 1 + \epsilon \\
 r_{s,a}(\pi) & \text{otherwise}
\end{cases}
\]

where \(\alpha\) is a hyperparameter to decide the scale of the functional clipping. The decreasing slope will be more steep when \(\alpha\) is larger.

Combining the original PPO algorithm with the novel function clipping method, we obtain the Proximal Policy Optimization Smoothed algorithm as shown in Fig. 1 which plots \(L^{\text{CLIP}}(\pi)\) as functions of the likelihood ratio \(r_{s,a}(\pi)\). We utilize the gradient change of hyperbolic tangent activation function to obtain the smoothed effect. As the figure depicted, when \(r_{s,a}(\pi)\) is out of the clipping range, the slope of \(L^{\text{PPOS}}(\pi)\) starts with a large absolute value and generally converges to zero. This smoothing mechanism guarantees the convergence of the clipping function. In contrast to PPORB, PPOS will never have an extremely large clipping output which is unacceptable in policy search. When \(\alpha < 0\), the function is more natural and smoother in shape. However, the actual performance will not be better than \(\alpha > 0\) which can be viewed in section IV-C. At the same time, the hyperbolic tangent activation function could be replaced by other activation functions to explore more.

IV. EXPERIMENTS

In this section, we present experiments to assess the performance of our proposed algorithm in terms of performance and stability during policy searches. We compare our method with the aforementioned state-of-the-art PPO variants in different environments. We also take different hyperparameteric choices into consideration and their corresponding results are shown, along with a quick guideline with tuning suggestions to quickly deploy our algorithm on different applications.

A. EXPERIMENTAL SETUP

We test the performance of our algorithm with the following benchmark: (a) PPO: the original PPO algorithm with the author recommended clipping range hyperparameter \(\epsilon = 0.2\), as seen in [20]. (b) PPORB: PPO with the rollback function. The rollback coefficient \(\alpha\) is set to be 0.3 for all tasks (except for the Humanoid-v2 task) as suggested by the authors in [26].

(c) PPOS: Our proposed method with an \(\alpha\) coefficient set according to the Table 2.

The algorithms are evaluated on continuous control benchmark tasks implemented on OpenAI Gym [4], simulated by MuJoCo [23] in a Tianshou platform.\(^1\) The chosen MuJoCo environment is the version-2 (v2) for all experiments. The actor and critic networks are neural networks with one hidden layer and 128 neurons per layer, and the adopted activation function is ReLU function [6]. Each algorithm was run with ten random seeds and 250,000 timesteps, which are obtained by 2500 steps per epoch and 100 epochs (except for the Humanoid-v2 task with 1 million timesteps, due to its higher dimensionality) using ADAM optimizer [11]. The details can be viewed on the Table 3 in the Appendix.

B. COMPARISON WITH OTHER METHODS

We conducted experiments in five Mujoco environments with continuous control tasks of both high and low dimensions, as shown in Fig. 2. In Fig. 3 we plot episode rewards during the training process on the tasks. Our PPOS method with the functional clipping mechanism significantly outperforms both PPO and PPORB on hard tasks characterized by high observation and action dimension (e.g., Humanoid-v2 with \(|\mathcal{O}| = 376\) and \(|\mathcal{A}| = 17\) and Ant-v2 with \(|\mathcal{O}| = 111\) and \(|\mathcal{A}| = 8\), both in terms of learning speed and final

\(^1\)https://github.com/thu-ml/tianshou
FIGURE 3. Episode rewards of the policy during the training process averaged over ten random seeds on MuJoCo tasks, comparing our methods with PPO and PPORB. The solid lines are mean values of ten trials while the shaded area depicts the 60% confidence interval. |C| represents the dimension of observation.

TABLE 1. Continuous control environments.

| Task             | PPO            | PPORB          | PPOS            |
|------------------|----------------|----------------|-----------------|
| Humanoid-v2      | 473.3 ± 61.18  | 499.4 ± 59.15  | 535.6 ± 59.67   |
| Ant-v2           | 2481 ± 380.6   | 2582 ± 404.9   | 2964 ± 189.3    |
| HalfCheetah-v2   | 2625 ± 164.5   | 2341 ± 399.7   | 2684 ± 191.1    |
| Swimmer-v2       | 53.68 ± 5.126  | 57.89 ± 2.733  | 57.50 ± 1.744   |
| Reacher-v2       | -4.306 ± 0.053 | -4.294 ± 0.061 | -4.290 ± 0.053  |

Average top episode rewards ± standard deviation over ten random seeds of three methods. Bold denotes the best result with the highest reward among three methods.

FIGURE 4. Entropy of the policy during the training process, averaged over ten random seeds on MuJoCo tasks, in comparison with PPO and PPORB. The solid lines are mean values of ten trials while the shaded area depicts the 60% confidence interval. |C| represents the dimension of observation.

rewards. For the HalfCheetah-v2 with medium dimensionality (|C| = 17 and |A| = 6) our method achieves a better final reward in the later stage. The advantage given by the clipping mechanism is less salient in lower dimensionality, and our PPOS is comparable to both PPO and PPORB in such tasks (e.g., Swimmer-v2 with |C| = 8 and |A| = 2 and Reacher-v2 with |C| = 11 and |A| = 2). The stricter clipping mechanism from both PPORB and PPOS have a tendency to slow down these algorithms on their initial stage (especially in low dimensions, while making it faster.
FIGURE 5. Episode rewards of the policy during the training process, averaged over ten random seeds on the Humanoid and the Ant tasks, in comparison with different $\alpha$ values of PPOS. The solid lines are mean values of ten trials while the shaded area depicts the 60% confidence interval. $|O|$ represents the dimension of observation.

in higher dimensionalities), with smaller policy updates, but eventually it enables these algorithms to asymptote at higher rewards.

The numerical analysis with mean and standard deviation is presented in Table 1. PPOS reaches a better performance in 80% of the adopted Mujoco tasks, and, as seen on Table 1, it was virtually tied with PPORB on the low dimensional Swimmer-v2 task (and presented a lower standard deviation). On average, PPOS reaches 8% higher rewards than PPO and 7% higher than PPORB, while the standard deviation of PPOS is 3% lower than both PPO and PPORB. A possible explanation for this is that the functional clipping mechanism successfully keeps the policy update within a reasonable range, while PPO and PPORB made the update either too large or too small. Besides, the nature of PPORB’s clipping can generate very negative updates, which can be detrimental to the policy search.

We also compared three methods on entropy during all training procedures as shown in Fig. 4. In general, PPOS and PPORB have lower entropy than PPO in most of the tasks which means PPOS and PPORB can make the trained policy more deterministic. In Humanoid-v2, PPOS can even have lower entropy than PPORB which means the policy is more practical for implements on real robots.

C. CHOICE OF THE HYPERPARAMETER

Correct hyperparametric choices are crucial for policy search algorithms. For example, in PPORB the coefficient of the rollback operation $\alpha$ could vary from 0.3 to 0.05 for different tasks. However, the suggested value was only provided by empirical methods for specific tasks in most of the former algorithms e.g. [18], [20], and [26]. With this in mind, we evaluate our choices of hyperparameters on varied tasks and provide a selection criterion of the hyperparameter value for different problem dimensionalities.

We conduct the experiment on Humanoid-v2 and Ant-v2 tasks, which have a higher dimensionality in their observation space, since in low-dimensional tasks the performance difference between PPO, PPORB and PPOS can be very subtle. As shown in Fig. 5, in the Humanoid-v2 task the coefficient $\alpha = 0.05$ achieves the best performance among different values, while $\alpha = 0.2$ is the best choice in the Ant-v2 task. As discussed in section III, the negative $\alpha$, which enables a smoother clipping function, is incapable of improving the performance of the Ant-v2. A possible solution for this issue is decreasing the clipping range towards a lower $\epsilon$ value. For the other three tasks with relatively low dimensions, the default $\alpha = 0.3$ can be regarded as the benchmark, which is presented in Table 2.

Once we empirically found the optimal parameters for all five environments, we use an exponential regression function $\alpha = 0.3333e^{-0.0048*|O|}$ to fit $\alpha$ value regarding the dimension of observation as shown in Fig. 6. In general, the proposed $\alpha$ is lower when the dimension of observation is higher. Our explanation for this is that a lower $\alpha$ value is effective in preventing unnecessarily large policy updates which are more likely to take place in high-dimensions. For low-dimensional tasks, default $\alpha = 0.3$ can balance the performance and robustness. The exponential regression function has two key

| Environment          | Value |
|----------------------|-------|
| Humanoid-v2          | 0.05  |
| Ant-v2               | 0.2   |
| HalfCheetah-v2       | 0.3   |
| Swimmer-v2           | 0.3   |
| Reacher-v2           | 0.3   |

FIGURE 6. The proposed $\alpha$ values in relation to the dimension of observation, derived from the exponential regression of five MuJoCo tasks we conducted. Generally speaking, higher dimensions of observation compel us to choose a lower $\alpha$ value to achieve a better performance.
advantages compared with linear and polynomial regressions: Firstly, it converges to 0 as $\alpha$ grows with an elegant shape which meets the intuition that higher dimensions will result in lower $\alpha$ values. Secondly, the function is more efficient to express the same data than higher-order polynomial expressions.

### V. CONCLUSION

Although PPO group algorithms have achieved state-of-the-art performance on policy searches on high-dimensional control tasks, they also present flaws when restricting the step of each policy update. The original clipping method of PPO cannot prevent large policy updates, while the improved PPORB cannot guarantee the convergence of this very same update. Additionally, as hyperparameters were proposed according to specific tasks, users are left without guidance to retune those parameters when redeploying the algorithm to their needs. Based on these observations, in this paper we proposed the PPOS algorithm with a functional clipping mechanism combined to the original PPO. Our work significantly improves the ability to restrict policy updates $\epsilon$ function. The default $\epsilon = 0.2$ is widely used in different variants of PPO, so there is considerable space for exploration of the clipping range. Another possible improvement in the future is the trust-region method, which is another mechanism to restrict the policy updates [18]. Finally, the implementations of deep reinforcement learning algorithms suffer from the repetitive tuning of hyperparameters [7]. Although we proposed a tuning method for one hyperparameter, there are still many hyperparameters, as seen in Table 3, which may need to be explored in those frameworks.

### APPENDIX

See Table 3.

### REFERENCES

[1] A. Abdolmaleki, R. Lioutikov, J. R. Peters, N. Lau, L. P. Reis, and G. Neumann, “Model-based relative entropy stochastic search,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 3537–3545.

[2] R. Akrou, G. Neumann, H. Abdulsamad, and A. Abdolmaleki, “Model-free trajectory optimization for reinforcement learning,” in Proc. Int. Conf. Mach. Learn., Jun. 2016, pp. 2961–2970.

[3] R. Akrou, A. Abdolmaleki, H. Abdulsamad, J. Peters, and G. Neumann, “Model-free trajectory-based policy optimization with monotonic improvement,” J. Mach. Learn. Res., vol. 19, no. 1, pp. 565–589, 2018.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “OpenAI gym,” 2016, arXiv:1606.01540. [Online]. Available: http://arxiv.org/abs/1606.01540

[5] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, “Benchmarking deep reinforcement learning for continuous control,” in Proc. Int. Conf. Mach. Learn., Jun. 2016, pp. 1329–1338.

[6] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in Proc. 14th Int. Conf. Artif. Intell. Statist., Jun. 2011, pp. 315–323.

[7] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep reinforcement learning that matters,” 2017, arXiv:1709.06560. [Online]. Available: http://arxiv.org/abs/1709.06560

[8] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, “A closer look at deep policy gradients,” 2018, arXiv:1811.02553. [Online]. Available: http://arxiv.org/abs/1811.02553

[9] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jastrzaski, “VizDoom: A doom-based AI research platform for visual reinforcement learning,” in Proc. IEEE Conf. Comput. Intell. Games (CIG), Sep. 2016, pp. 1–8.

[10] H. Kimura and S. Kobayashi, “An analysis of actor-critic algorithms using eligibility traces: Reinforcement learning with imperfect value functions,” J. Jpn. Soc. Artif. Intell., vol. 17, no. 1, pp. 267–275, 2000.

[11] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 1334–1373, 2015.

[13] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, “Using deep reinforcement learning to learn high-level policies on the ATRIAS biped,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2019, pp. 263–269.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013, arXiv:1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602

| Hyperparameter       | Value            |
|----------------------|------------------|
| $\epsilon$           | 0.2              |
| $\alpha$ in PPORB     | 0.02 (Humanoid), 0.3 (Other) |
| buffer size           | 20000            |
| learning rate         | 0.0003           |
| $\gamma$              | 0.99             |
| epoch                 | 100              |
| step per epoch        | 2500             |
| collect per step      | 10               |
| repeat per collect    | 2                |
| batch size            | 128              |
| hidden layer number   | 1                |
| hidden neurons per layer | 128            |
| training number       | 8                |
| test number           | 100              |
| $\lambda$ (GAE)       | 0.95             |
| policy                | Gaussian         |
16] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” *Neural Netw.*, vol. 21, no. 4, pp. 682–697, May 2008.

17] J. Peters, K. Mülling, and Y. Altun, “Relative entropy policy search,” in *Proc. AAAI*, vol. 10, Jul. 2010 pp. 1607–1612.

18] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 1889–1897.

19] J. Peters, K. Mülling, and Y. Altun, “Relative entropy policy search,” in *Proc. AAAI*, vol. 10, Jul. 2010 pp. 1607–1612.

20] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 1889–1897.

21] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “High-dimensional continuous control using generalized advantage estimation,” 2015, arXiv:1506.02438. [Online]. Available: http://arxiv.org/abs/1506.02438

22] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

23] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

24] E. Todorov, T. Erez, and Y. Tassa. “MuJoCo: A physics engine for model-based control,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5026–5033.

25] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” 2015, arXiv:1509.06461. [Online]. Available: http://arxiv.org/abs/1509.06461

26] Y. Wang, H. He, X. Tan, and Y. Gan, “Trust region-guided proximal policy optimization,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 626–636.

27] Y. Wang, H. He, C. Wen, and X. Tan, “Truly proximal policy optimization,” 2019, arXiv:1903.07940. [Online]. Available: http://arxiv.org/abs/1903.07940

28] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning.” *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.

WANGSHU ZHU received the bachelor’s degree in computer science from ShanghaiTech University, in 2019, where he is currently pursuing the master’s degree in computer science with the School of Information and Science Technology. He joined the Living Machines Laboratory, ShanghaiTech University, in 2018. His current research interests include model-based fault tolerance for manipulators and reinforcement learning method for robotics. He also focuses on the combination of model-based learning methods and robots. He was awarded by the Merit Student of ShanghaiTech University, in 2020.

ANDRE ROSENDO (Member, IEEE) received the bachelor’s degree in mechanical engineering from Bahia Federal University, in 2008, the master’s degree in computer science from Hokkaido University, in 2011, and the Ph.D. degree in computer science from Osaka University, in 2014. From 2014 to 2015, he was a Postdoctoral Researcher with Osaka University. From 2015 to 2017, he was a Research Associate with the Engineering Department, University of Cambridge. He was appointed as an Assistant Professor of robotics and artificial intelligence with ShanghaiTech University, China, in 2017, where he is currently the Director of the Living Machines Laboratory. His current research interest includes the capacity of robots to recover and learn new behaviors after morphological/control failures. He also focuses on model-based fault tolerance for manipulators, and Bayesian learning methods for legged robots, manipulators, and autonomous robots. He won the National Natural Sciences Foundation of China 2019 Grant and the Oriental Talent 2019 Award.