Controlling Agents by Constrained Policy Updates

Mónika Farsang
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
monika.farsang@edu.bme.hu

Luca Szegletes
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
luca.szegletes@aut.bme.hu

***Abstract*—Learning the optimal behavior is the ultimate goal in reinforcement learning. This can be achieved by many different approaches, the most successful of them are policy gradient methods. However, they can suffer from undesirably large updates of policies, leading to poor performance. In recent years there has been a clear trend toward designing more reliable algorithms. This paper addresses to examine different restriction strategies applied to the widely used Proximal Policy Optimization (PPO-Clip) technique. We also question whether the analyzed methods are able to adapt not only to low-dimensional tasks but also to complex, high-dimensional problems in control and robotic domains. The analysis of the learned behavior shows that these methods can lead to better performance compared to the original PPO-Clip algorithm, moreover, they are also able to achieve complex behavior and policies in high-dimensional environments.**

***Index Terms*—Control, Constrained policy, Proximal Policy Optimization, Reinforcement learning**

I. Introduction

Deep reinforcement learning (DRL) has demonstrated that it can be successfully applied in a number of challenging simulated and real-world problems, encompassing games [1], [2], [3], continuous control [4], [5] and robotics [6], [7].

Policy gradient methods [8], [9] update policy parameters along an estimated ascent direction of the expected return. These approaches are well suited in domains of continuous control, since they scale to high-dimensional action spaces without difficulty and have a more stable performance compared to value-based methods, especially with function approximation [10]. However, getting the desired results via these techniques is challenging. Depending on the step-size parameter, the progress can be extremely slow if it is too small or there can be unintended drops in performance if it is too large. Another problem is their poor sample efficiency and even learning simple tasks can take a long time.

In recent years, there has been a trend in the direction of improving policy gradient approaches to be more robust and data efficient. Starting from Trust Region Policy Optimization (TRPO) [11], through the Actor-Critic with Experience Replay (ACER) [12] algorithm and ending with PPO [13], which has two variants: PPO-Penalty and PPO-Clip.

It has been shown in [13] that PPO-Clip performs better than the vanilla policy gradient with adaptive step-size, the TRPO algorithm, the Cross-Entropy Method (CEM) [14], the Advantage Actor Critic (A2C) [15] and its variant with trust region [12] on continuous control tasks. In the case of Atari games, PPO-Clip outperforms A2C and gets similar results to ACER but has much simpler implementation.

In this paper we aim to further refine the robust PPO-Clip algorithm with changing policy update restrictions over the learning phase. For this, we modify its surrogate objective function. As an extension to our earlier work [16], we analyze new clipping strategies in addition to the previous ones and investigate their importance in more simulated environments to better understand their impact on the behavior of the agent.

The rest of this paper is organized as follows. Section II presents the research background of policy gradient algorithms. The setup and the proposed methods are discussed in Section III. In Section IV, the evaluation metrics and the results in different simulation environments are presented. Section V gives the conclusion.

II. Background

The loss function of policy gradient methods (1) takes the empirical average of the parametrized policy π_θ multiplied by the estimate of the advantage function $\hat{A}_t$ over a finite batch of samples, where the latter gives a relative benefit of selecting a certain action in a given state. For instance, negative advantage values lead to negative gradients, resulting in a decrease in the probability of selecting the current action. A potential problem with this approach is that it might lead to undesirable behavior with large policy updates.

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right] \tag{1}$$

TRPO was proposed as a solution to ensure monotonic improvement during the policy updates. Instead of applying a fixed penalty coefficient, it operates with the Kullback–Leibler divergence (i.e., relative entropy). Under this constraint, it takes the largest step possible towards improving the policy (2). This approach provides a more stable and reliable policy gradient method. However, due to its second-order optimization technique, scaling is difficult and it has high computational complexity.

$$\begin{aligned} \max_\theta \quad & L_{\pi_{\theta_{old}}}(\pi_\theta) \\ \text{s.t.} \quad & \overline{D}_{KL}^{\rho_{\theta_{old}}}(\theta_{old}, \theta) \le \delta, \text{ where} \\ \overline{D}_{KL}^{\rho}(\theta_1, \theta_2) &= \mathbb{E}_{s\sim\rho}\left[D_{KL}\left(\pi_{\theta_1}(\cdot|s) \;||\; \pi_{\theta_2}(\cdot|s) \right) \right]. \end{aligned} \tag{2}$$

To overcome the aforementioned limitations of TRPO, Schulman *et al.* proposed two variants of PPO [13], called PPO-Penalty and PPO-Clip.

Similar to TRPO, the PPO-Penalty method approximately computes updates constrained by the KL-divergence. The difference between them is that the former uses it as a hard constraint, the latter penalizes it in the objective and scales the penalty coefficient throughout the training.

PPO-Clip simplifies the algorithm by completely removing the KL-divergence from the objective function and has no constraints (3).

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \Big[\min\Big(r_t(\theta)\hat{A}_t, \\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big], \qquad (3)$$

$$\text{where } r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}.$$

Instead of them, it operates with a special clipping term in the objective function which removes updates that are far from the old policy. This is solved by clipping the probability ratio $r_t(\theta)$, which measures the changed probability of the chosen action a_t in state s_t under the new and the old policy, respectively. The novel surrogate objective function uses a hyperparameter ϵ with the clipping term, which keeps r_t inside the interval $[1-\epsilon, 1+\epsilon]$. Taking the minimum value of the clipped and unclipped objectives gives a lower bound on the latter.

III. Methods

To further refine the idea of PPO-Clip algorithm, we propose alternative clipping range approaches. Our motivation is to continuously influence the behavior of the agent with changing policy-update restrictions during training. For this, instead of using the original objective function (3), we modify it to have a time-dependent ϵ_t parameter (4).

$$L^{CLIP_t}(\theta) = \hat{\mathbb{E}}_t \Big[\min\Big(r_t(\theta)\hat{A}_t, \\ \text{clip}(r_t(\theta), 1-\epsilon_t, 1+\epsilon_t)\hat{A}_t\Big)\Big] \qquad (4)$$

Figure 1 shows two cases of the L^{CLIP_t} as a function of the probability ratio, which are distinguished by the sign of the advantage values. This kind of illustration comes from the original paper [13], but we use a slight modification of it with including the time t aspect. First, Fig. 1(a) presents the action with an estimated positive impact on the outcome. In this case, clipping occurs when the action becomes more likely under the current policy than under the old one. Since this is only a local approximation and comes from a sample of the policy, using large updates may not be accurate, thus applying the clipping mechanism prevents the objective function from growing. The other case is depicted in Fig. 1(b), when the advantage function is negative, i.e., where the action has an estimated negative effect on the outcome. Here the clipping occurs in the other region where the action is unlikely under the current policy. This clipping region similarly prevents too large updates in this direction, without which the update would make the action much less likely. Overall, it can be seen how the ϵ value controls the allowed policy updates. As the extra part, we also change these clipping ranges throughout the training according to some strategy.

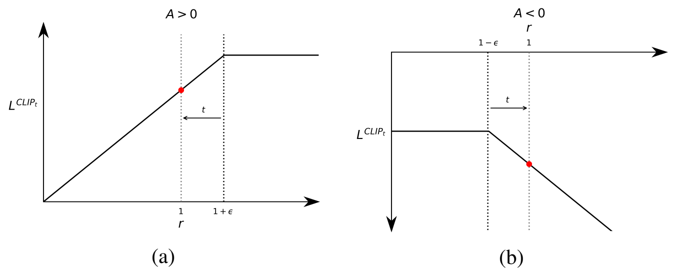


Fig. 1: Clipping mechanism with time parameter in the case of positive (a) and negative (b) advantages.

Our examined methods (excluding the moving average strategy) support: (1) higher exploration at the beginning when the agent has limited information of its environment and (2) stronger restrictions at the end of the training when exploiting the knowledge of the agent obtained so far should be favored.

Equation (5) shows the clipping with a constant ϵ_0 during the training. This technique was used in the original paper. It serves as a baseline when comparing the other methods.

$$\epsilon_t^{const} = \epsilon_0 \qquad (5)$$

Linear and exponential decrease can be possible approaches of varying the clipping range from the initial value ϵ_0. Clipping parameter values go to zero at the end of the training in the case of linear decrement (6), while a slower decay can be determined with the exponential method if the value of α is relatively close to 1 (7). These strategies have been proposed and analyzed in one of our previous works [16].

$$\epsilon_t^{lin} = \frac{T-t}{T}\epsilon_0 \qquad (6)$$

$$\epsilon_t^{exp} = \alpha^{100\frac{t}{T}}\epsilon_0 \qquad (7)$$

We extended these methods with two new alternative techniques. First, we present a Z-shaped function (8), which starts slowly decaying from an initial value ϵ_0 followed by a rapidly decreasing phase in the middle and finally converges to a final value of ϵ_T.

$$\epsilon_t^{Z-shaped} = \epsilon_0 + \frac{\epsilon_T}{1 + e^{\frac{10(t-\frac{T}{2})}{\frac{T}{2}}}} \qquad (8)$$

Our other proposed approach adapts to the latest clipping values with a moving average method (9). In this case, a window of size n stores the last n clipping values, whose average is the current clipping range. However, this mean value might be too small in some situations, so a lower bound ϵ_{lb} stops it from falling below that certain point. At the same time,

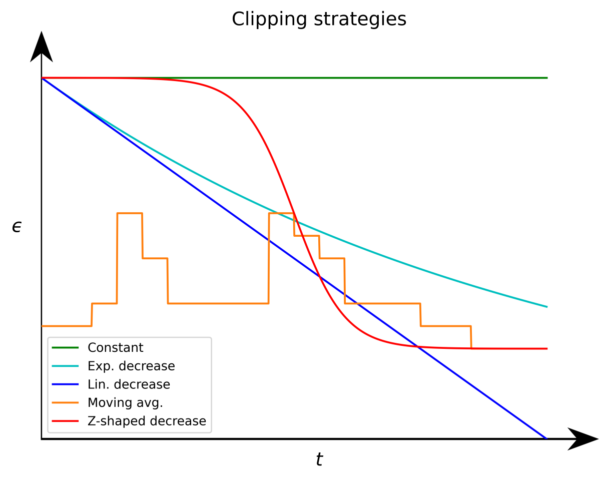


Fig. 2: Analyzed clipping strategies: constant value, exponential and linear decrease, moving average approach and Z-shape decrease.

we also give an upper bound ϵ_{ub} to prevent large changes in order to preserve the idea of not stepping too far away from the previous policy during an update.

$$\epsilon_t^{mov.avg.} = \min\left(\max\left(\frac{\sum_{i=1}^{n}\epsilon_{t-i}}{n}, \epsilon_{lb}\right), \epsilon_{ub}\right) \tag{9}$$

Overall, the strategies analyzed in this paper consist of the baseline constant approach, the linear, the exponential and the Z-shaped clipping range reduction throughout the training, as well as the moving average approach. Fig. 2 depicts examples of these methods.

Algorithm 1 shows the resulting PPO algorithm extended with the different clipping range strategies. The unmodified data collection and a part of the policy optimization are taken from [13].

Algorithm 1: PPO-Clip algorithm with clipping strategies

```
Choosing one of the following clipping range
  strategies:
    (a) Not decremented, using constant value $\epsilon$ (5)
    (b) Decreasing linearly (6)
    (c) Decreasing exponentially (7)
    (d) According to a Z-shaped curve (8)
    (e) Using moving average with upper and lower
      bounds (9)

for iteration = 1, 2, ... do
    Data collection
    for actor = 1, 2, ..., N do
        Run policy $\pi_{\theta_{old}}$ in environment for $T$
          timesteps
        Compute advantage estimates $\hat{A}_1 \dots \hat{A}_T$
    end

    Policy optimization
    Update $\epsilon_t$ according to the chosen clipping range
      method
    Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and
      minibatch size $M \leq NT$
    $\theta_{old} \leftarrow \theta_{new}$
end
```

IV. Experiments

During our experiments, in order to measure how efficient the learning is, the evaluation metrics proposed by Schulman *et al.* are used to compare the clipping methods:

- **Metric I.** *Average reward per episode over the entire training period to measure how well the approach does during learning.*
- **Metric II.** *Average reward of the last 100 episodes of training to measure the final performance.*

We benchmark the different clipping strategies in nine control and robotic environments from OpenAI Gym [17] and PyBullet [18] task suites. Our state and action spaces are based on the original simulations, with minor modifications: input features are normalized, image observations are downsampled and converted to grayscale. In the PyBullet environments, time features are concatenated to the current observations.

The mean rewards computed for 5 seeds are shown in each figure. The lighter shaded area in the learning curve figures depicts the standard deviation of the results, while the darker shaded region corresponds to the standard error of the mean.

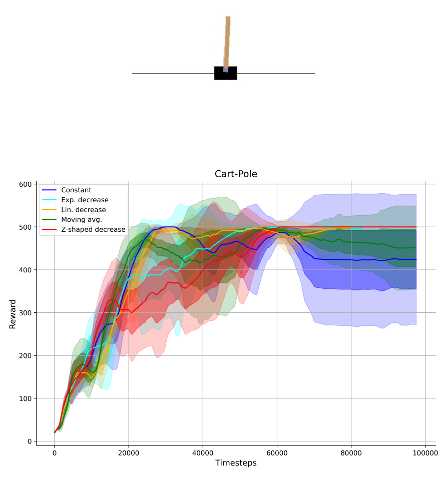
A. Classical control tasks

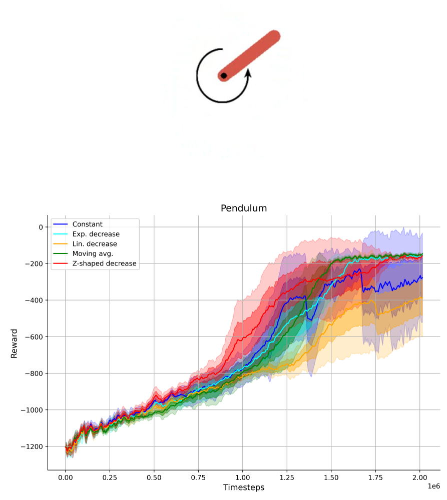
OpenAI Gym offers numerous control theory problems, which include (1) stabilizing a pole on a cart, (2) keeping a pendulum in an upright position and (3) swinging up the end-effector of a two-link robot to a specified height. These are shown in the first row of Fig. 3. Solving them can be used as a demonstration that a DRL algorithm is successful in problems that can be addressed by traditional control techniques as well.

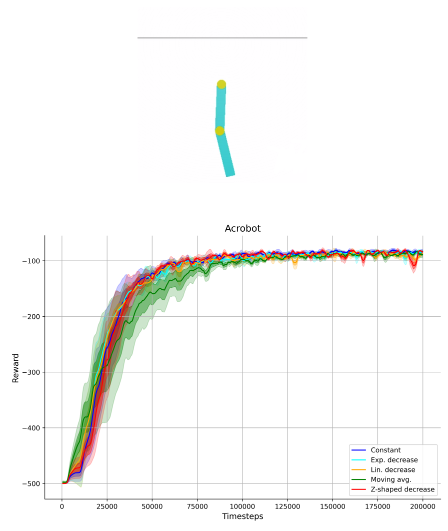
The analyzed clipping strategies can play a significant role in terms of average reward in the Cart-Pole and Pendulum environments. Many of them worked effectively, to highlight the best ones, the linear, exponential and Z-shaped decaying approaches showed the best performance over the whole training. Meanwhile, the Z-shaped clipping decline and the moving average method achieved outstanding results in the last 100 episodes. In the case of the Acrobot, the marginal changes in the performance suggest that the different clipping mechanisms do not affect the behavior in this environment.

B. Continuous control tasks

Continuous control problems in the Box2D simulator are also part of OpenAI Gym, from which the selected ones are displayed in Fig. 4. These problems consist of (1) landing a

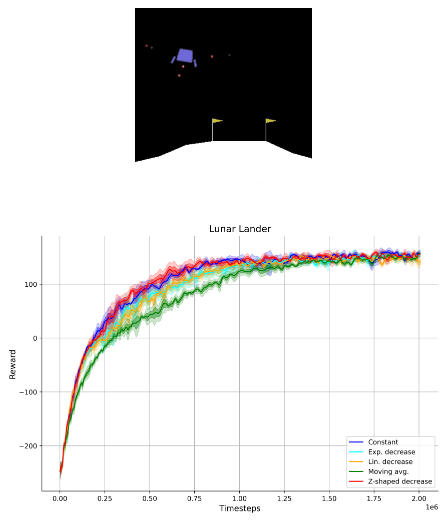


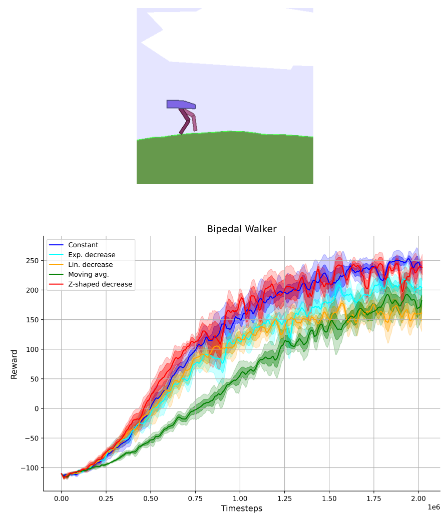


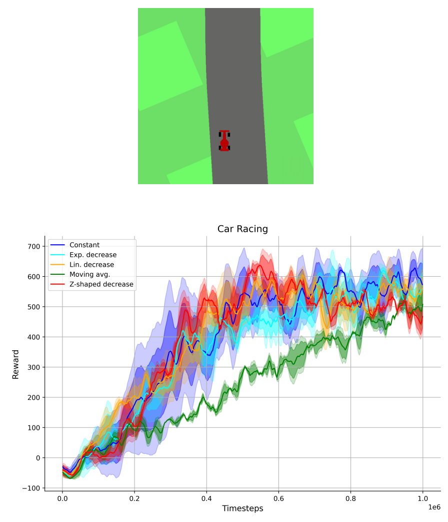


(a) (b) (c)

Fig. 3: Comparison of clipping strategies on OpenAI Gym classical control simulation environments. (a) Training CartPole-v1 for 100 thousand timesteps. (b) Pendulum-v0 for 2 million timesteps. (c) Acrobot-v1 for 200 thousand timesteps.







(a) (b) (c)

Fig. 4: Learning curves on continuous control environments from OpenAI Gym. (a) LunarLanderContinuous-v2 for 2 million timesteps. (b) BipedalWalker-v3 for 2 million timesteps. (c) CarRacing-v0 for 1 million timesteps.

rocket on a landing pad as quickly and efficiently as possible, (2) learning the walking motion of a 4-joint robot on slightly uneven terrain and (3) racing a car around a track from raw pixels.

In the case of these Box2D simulation environments, the proposed clipping strategies could not significantly exceed the constant clipping baseline. However, we also find it important to illustrate the limitations of these approaches by presenting these environments, where their results are moderate.

C. Robotic control tasks

Robotic environments in PyBullet are more detailed than the aforementioned environments and are good indicators of whether the clipping strategies are capable of succeeding in higher dimensional state and action spaces. The selected environments include different configurations of 2D robots with to goal of (1) hopping forward, (2) walking and (3) running.

The analyzed clipping range approaches have a significant impact on PyBullet environments, especially on the Hopper and Walker robots. To highlight the best strategies, Z-shaped and exponentially declining methods achieved the highest average results in these three environments. These results are especially important because these environments are considered to be the most complex ones as they require the learning of high-dimensional robot-movements.

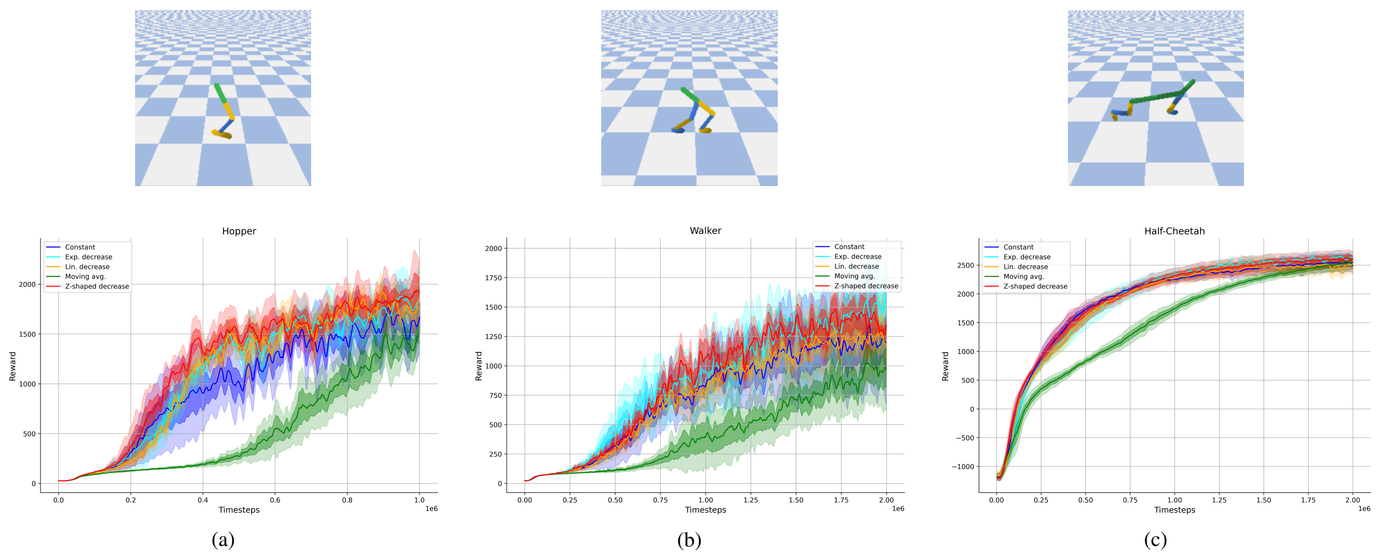


Fig. 5: Experiment results from simulated robotic locomotion, using PyBullet. (a) HopperBulletEnv-v0 for 1 million timesteps. (b) Walker2DBulletEnv-v0 for 2 million timesteps. (c) HalfCheetahBulletEnv-v0 for 2 million timesteps.

TABLE I:
OVERALL PERFORMANCE OF THE CLIPPING STRATEGIES

	Metric I	Metric II
Constant	0	4
Linear decrease	2	0
Exponential decrease	2	2
Z-shaped decrease	5	2
Moving average	0	1

D. *Comparison*

Table I shows the number of times when the strategy created the most successful agent based on the given metric in the nine different environments. Our proposed clipping range strategies, which are designed to further refine this state-of-the-art method of PPO-Clip, are able to achieve better results than the original constant approach in all environments based on Metric I, which measures the overall performance. Regarding Metric II with the final rewards, 5 out of 9 performed with higher scores. Especially the exponential and Z-shaped declining strategies can be considered promising.

The detailed results of the average and the standard deviation values for the each environment is presented in Table II and Table III of Metric I and Metric II, respectively.

These results suggest that the strategies can be successfully applied on low-dimensional tasks - as in the case of the Cart-Pole or Pendulum, and on high-dimensional environments as well - like in the locomotive motion of the Hopper and Walker robots.

V. CONCLUSION

In this paper, we have introduced new clipping strategies which are possible refinements to the robust PPO-Clip algorithm. The key idea underlying our proposed methods is to continuously influence the behavior of the agent with changing policy-update restrictions throughout the training. We evaluated these approaches on several control and robotic benchmarks and demonstrated that they can adapt better in many cases than the constant baseline.

However, there are some environments where these strategies could not result in better performance. Taking this also into account, we conclude that even though they are not general solutions but are likely to be promising alternatives to the constant clipping approach and they are able to deal with problems that have complex, high-dimensional characteristics.

REFERENCES

[1] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[2] D. Silver *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. [Online]. Available: https://doi.org/10.1038/nature16961

[3] S. Risi and M. Preuss, “From chess and atari to starcraft and beyond: How game AI is driving the world of AI,” *Künstliche Intell.*, vol. 34, no. 1, pp. 7–17, 2020. [Online]. Available: https://doi.org/10.1007/s13218-020-00647-w

[4] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” 2019.

[5] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, “Deepmimic: Example-guided deep reinforcement learning of physics-based character skills,” *ACM Trans. Graph.*, vol. 37, no. 4, jul 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201311

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013. [Online]. Available: https://doi.org/10.1177/0278364913495721

[7] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[8] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[9] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 2000. [Online]. Available: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf

[10] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 2000. [Online]. Available: https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[11] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, vol. 37, 2015, pp. 1889–1897.

[12] Z. Wang *et al.*, "Sample efficient actor-critic with experience replay," 2017.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[14] I. Szita and A. Lörincz, "Learning tetris using the noisy cross-entropy method," *Neural Computation*, vol. 18, no. 12, pp. 2936–2941, 2006.

[15] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, p. 1928–1937.

[16] M. Farsang and L. Szegletes, "Decaying clipping range in proximal policy optimization," in *15th IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI 2021, Timisoara, Romania, May 19-21, 2021*. IEEE, 2021, pp. 521–526. [Online]. Available: https://doi.org/10.1109/SACI51354.2021.9465602

[17] G. Brockman *et al.*, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning," 2017. [Online]. Available: http://pybullet.org

TABLE II:
METRIC I EVALUATION OF CLIPPING STRATEGIES

Environment	Constant	Linear decrease	Exponential decrease	Z-shaped decrease	Moving average
CartPole-v1	399.765±112.633	**421.945±131.627**	414.259±124.211	398.662±120.767	407.131±116.380
Pendulum-v0	−690.829±314.949	−787.112±247.400	−676.595±352.304	**−623.782±350.296**	−676.285±373.450
Acrobot-v1	−141.117±111.108	−141.960±104.012	**−140.325±103.925**	−143.725±113.115	−150.722±101.809
LunarLanderContinuous-v2	105.186±77.946	95.357±79.155	96.142±79.096	**106.692±78.407**	80.838±84.499
BipedalWalker-v3	111.768±122.980	75.473±94.602	85.374±106.999	**117.401±119.138**	43.425±100.628
CarRacing-v0	384.525±207.664	**389.937±191.891**	362.941±195.433	383.473±209.828	241.474±164.703
HopperBulletEnv-v0	977.438±537.832	1 103.823±645.376	1 100.564±620.442	**1 199.577±632.202**	523.357±465.855
Walker2DBulletEnv-v0	728.424±413.510	733.732±426.243	**853.175±476.530**	849.202±491.114	420.007±305.338
HalfCheetahBulletEnv-v0	1 889.533±874.252	1 851.713±891.705	1 901.982±930.460	**1 927.747±875.512**	1 472.236±950.840

TABLE III:
METRIC II EVALUATION OF CLIPPING STRATEGIES

Environment	Constant	Linear decrease	Exponential decrease	Z-shaped decrease	Moving average
CartPole-v1	423.359±1.036	499.914±0.383	499.482±0.888	**500.000±4.019**	458.027±5.372
Pendulum-v0	−309.373±32.111	−428.424±26.175	−170.990±8.109	−199.032±35.718	**−159.330±6.468**
Acrobot-v1	**−84.783±1.746**	−89.356±3.311	−86.075±2.917	−87.561±6.850	−88.208±2.383
LunarLanderContinuous-v2	**152.633±5.151**	146.928±3.340	150.674±2.953	152.484±2.973	147.870±4.131
BipedalWalker-v3	**238.901±9.248**	160.512±7.534	202.059±10.620	230.381±12.223	169.297±10.833
CarRacing-v0	**567.447±31.709**	533.025±33.237	548.630±31.593	498.842±24.701	462.883±30.622
HopperBulletEnv-v0	1 582.347±78.729	1 776.766±61.027	1 724.929±88.722	**1 818.443±59.110**	1 296.595±131.718
Walker2DBulletEnv-v0	1 194.325±44.653	1 227.991±33.365	**1 415.977±79.473**	1 371.510±60.074	882.579±79.159
HalfCheetahBulletEnv-v0	2 520.181±18.998	2 480.074±21.191	**2 600.089±28.761**	2 586.257±21.353	2 462.170±57.156