Swap Softmax Twin Delayed Deep Deterministic Policy Gradient

1st Chaohu Liu
Department of Automation
University of Science and Technology of China
Hefei, China
liuchaohu@mail.ustc.edu.cn

2nd Yunbo Zhao*
Department of Automation
University of Science and Technology of China
Institute of Artificial Intelligence
Hefei Comprehensive National Science Center
Hefei, China
ybzhao@ustc.edu.cn

2023 6th International Symposium on Autonomous Systems (ISAS) | 979-8-3503-1615-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISAS59543.2023.10164333

***Abstract*—Reinforcement learning algorithms have attained noteworthy accomplishments in the field of continuous control. One of the classic algorithms in continuous control, the DDPG algorithm, is widely used and has been shown to be susceptible to overestimation. Following this, the TD3 algorithm was introduced, which integrated the notion of double DQN. TD3 takes into account the minimum value between a pair of critics to restrict overestimation. Nevertheless, TD3 may lead to an underestimation bias. To mitigate the impact of errors, we present a novel approach by integrating Swap Softmax with TD3, which can counterbalance the extreme values. We assess the efficacy of our proposed technique on continuous control tasks that are simulated by MuJoCo and provided by OpenAI Gym. Our experimental findings demonstrate a significant enhancement in the performance and robustness.**

***Index Terms*—reinforcement learning, underestimation, continuous control, policy gradient, softmax**

I. Introduction

Over the past few years, there have been significant advancements in several domains through the amalgamation of deep learning and reinforcement learning methodologies [1]–[3]. A pivotal breakthrough has been the advent of Deep Q-learning (DQN), which can attain and even surpass the proficiency level of human experts in numerous Atari video games by utilizing raw images as direct inputs [4]. The DQN algorithm employs a deep neural network to approximate the action value function [5]. However, it is worth mentioning that DQN is predominantly utilized for cases where the action space is low-dimensional and discrete.

Within the realm of continuous control, Deep Deterministic Policy Gradients (DDPG) is an off-policy and model-free algorithm that relies on deterministic policy gradients for actor-critic architectures. It utilizes deep neural network approximations to acquire policies within high-dimensional, continuous action spaces [6]. The DDPG algorithm is versatile enough to be employed across a diverse range of tasks, encompassing pendulum swinging, dexterous maneuvers, leg movements, and car driving. Nevertheless, the DDPG algorithm continues to encounter two major issues:

- The overestimation of the Q value is primarily caused by the properties of Temporal-Difference (TD) learning and the inaccurate fitting of the neural network. This error accumulates over the process of learning, eventually resulting in either the learning of an inadequate policy or even the absence of convergence.
- An excessive amount of variance in the estimation process may cause amplified perturbation to policy gradients [7], leading to the unpredictability of an agent's behavior and hindering the effectiveness of the training procedure.

To tackle the challenges stemming from the DDPG algorithm, the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm was developed and has demonstrated superior performance on related tasks [8]. It adopts the concept of double Q-learning from Double DQN and leverages the use of two separate critic networks [9]. When computing the Q value estimation, the two networks are asked to predict separately and then the smaller value is selected. Although the employment of this update rule may induce an underestimation bias, it is highly desirable as compared to an overestimation bias, in which the value of an action would be propagated explicitly through the policy update. From an intuitive perspective, the excessively conservative nature of the TD3 algorithm can facilitate the agent in achieving satisfactory performance and stable convergence. Nevertheless, it is also likely to constrain the exploration capabilities thereof.

To obtain more precise target estimations, Softmax Deep Double Deterministic Policy Gradients (SD3) can be implemented. This algorithm is based on a single-dual estimator and effectively mitigates the biases of both overestimation and underestimation [10]. Nevertheless, SD3 is substantially more intricate and computationally demanding compared to other algorithms.

This paper draws significant inspiration from both TD3 and SD3 algorithms, and proposes a novel Swap Softmax operation to enhance the performance of our proposed algorithm. The structure of this paper is as follows: Section I illustrates the

This work was supported by the National Key Research and Development Program of China (No. 2018AAA01008001, No.2018YFE01068001), the National Natural Science Foundation of China (62033012, 61725304), the Science and Technology Major Project of Anhui Province (912198698036).

*Corresponding author.

practical applications of reinforcement learning and highlights the limitations of existing approaches. Section II expounds on the fundamental framework of reinforcement learning and conducts a more in-depth analysis of the problem. Section III outlines our reinforcement learning algorithm for resolving continuous control problems. Section IV showcases the performance of our algorithm across various MuJoCo simulation environments [11]. Finally, Section V concludes this paper by providing a comprehensive summary of its contents.

II. Background

Reinforcement Learning (RL) is conventionally conceptualized as a Markov Decision Process (MDP). The fundamental components of the Reinforcement Learning (RL) framework comprises a five-tuple (S, A, P, R, γ), wherein S denotes the set of states, A represents the set of actions, P refers to the transition probability, R denotes the reward function and γ represents the discount factor [12], [13].

The MDP framework allows the agent to perceive the existing state of the environment and take a corresponding action, which subsequently causes the environment to transition to the next state. The environment then provides a reward based on the reward function associated with the transition. The essence of Reinforcement Learning is to enable the agent to master a policy $\pi(s;\phi)$ where ϕ is a parameter for the agent to learn what to do in a certain state, which aims to obtain the maximum expected long-term reward $J(\pi(\cdot;\phi)) = E\left[\sum_{k=0}^{\infty}\gamma^k R\left(s_k, a_k\right) \mid \pi(\cdot;\phi)\right]$. This paper focuses on deterministic policy.

The Actor-Critic architecture is a significantly influential RL paradigm that incorporates two core elements: an Actor, responsible for generating the policy, and a Critic, accountable for evaluating the policy's effectiveness. In particular, the Critic evaluates the quality of the policy by utilizing a value function $Q(s, a;\theta)$ that expresses the expected maximum reward attainable by the agent in state s upon selecting action a.

In reinforcement learning algorithms based on the Actor-Critic architecture, the Actor plays the primary role in determining the action to undertake while the Critic provides a corresponding score to assess the efficacy of the Actor's decision-making ability [12]. By constantly adapting its policy to the feedback provided by the Critic, the Actor endeavors to optimize its decision-making ability. It is rational to assume that a superior Critic would enable the identification of an improved Actor. Therefore, Actor-Critic algorithms are expected to enhance the performance of both the Actor and the Critic continuously. As both the Actor and Critic are typically implemented using neural networks, developing an appropriate loss function is essential to updating the network via gradient descent in practical applications. For instance, in the DDPG algorithm, target networks are utilized to expedite convergence, and the loss function is mathematically expressed as:

$$y = r + \gamma Q'(s', \pi'(s')) \tag{1}$$

$$Loss = \sum (y - Q(,a))^2/N \tag{2}$$

where $Q'(s, a|\theta')$ and $\pi(s|\phi')$ are the corresponding target networks. Although this algorithm is highly effective, the presence of overestimated value still causes the performance of it to deteriorate. To alleviate the problem of overestimation, the TD3 approach, which draws inspiration from the double Q-learning technique, adopts a strategy that entails taking the minimum value of a pair of Critic networks to regulate overoptimistic estimates. The formula can be expressed as follows:

$$y = r + \gamma min_{j=1,2} Q'_j(s', \pi'(s')) \tag{3}$$

$$Loss = \sum (y - Q_1(s,a))^2/N \tag{4}$$

The aforementioned update rule exhibits a clear predisposition towards the occurrence of underestimation bias, which, in turn, remains a more desirable outcome when compared to the potential for overestimation bias.

Accurately estimating values is crucial for actor-critic based algorithms to perform better, thus our aim is to develop an algorithm that can neutralize the two estimated values based on TD3.

III. The Proposed Approach

In this section, we incorporate an Attention Mechanism (AM) into the TD3 algorithm to achieve significant performance improvement with minimal additional computation. AM is a critical constituent integrated into deep learning that enables the autonomous acquisition and measurement of the input data's impact on output efficacy [14]. As there are multiple target values available to choose from for the TD3 algorithm, it is intuitive to consider them together. In this scenario, the Attention Mechanism is a perfect fit to address this type of problems.

A. Preliminaries

Some commonly-used methods in the practical application of Reinforcement Learning are initially presented.

1) Double Network.: DPG algorithm, a derivative of DQN, is an effective methodology for solving the continuous control predicament. However, DQN is prone to the overestimation issue, which is the estimated value function is larger than the true value function. DQN represents an off-policy methodology, which entails the usage of a predicted action with the highest value to update the target value function rather than the real action of the subsequent occurrence in every learning cycle. Unfortunately, this approach tends to produce overestimations of the Q value.

The function approximation approach expresses the formula that governs the value function update process as follows:

$$y = \ r + \gamma \max_{a'} Q\left(s', a';\theta\right) - Q(s,a;\theta) \tag{5}$$

$$\theta_{t+1} = \theta_t + \alpha y \nabla Q(s,a;\theta) \tag{6}$$

The value function precision is naturally diminished by this approach due to its underlying reliance on the Bellman

equation for value estimation, in which the value is iteratively updated through subsequent states. Consequently, the accuracy loss issue is further exacerbated. The utilization of an imprecise estimate at each update of the policy will give rise to the buildup of errors. These amounting mistakes can result in an overestimation of a bad state, which in the end brings about a policy that cannot be improved to the ideal and hinders the algorithm from arriving at a convergence.

Double DQN [15] carries out both selection and evaluation of actions through the use of distinct value functions. Specifically, the selection of actions is given by $a^* = argmax_x Q(s', a; \theta)$. Subsequently, the target is constructed as $t = r + \gamma Q(s', argmax_a Q(s', a; \theta); \theta')$.

Double DQN leverages the notion of Double Q-Learning to mitigate the overestimation issue in Q-Value that its predecessor, DQN, encounters. Specifically, Double DQN calculates the selected action and its estimated value on both the prediction network and the target network, respectively. Analogously, DDPG has also been susceptible to this same problem of overestimation.

The TD3 algorithm is equipped with a pair of networks that are responsible for calculating distinct Q values. To tackle the issue of persistent overestimation, TD3 adopts a strategy of designating the smaller of the two as the target.

Note: Here, we employ two Critic networks, each of which has a matching target network, thus the TD3 algorithm necessitates a total of six networks.

2) Delayed Policy Updates.: Within the dual network architecture, asynchronous updating is adopted for the target network following the updating of the current network for d iterations. This can effectively mitigate the accumulation of errors and subsequent variance [16]. Analogously, delaying updates to the policy network is also viable given the gradual parameter updates in actor-critic methods. Such a strategy can help curtail superfluous updates and accumulated errors over multiple updates [17]. Furthermore, when reducing update frequency, it is recommended to employ soft updates as well:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{7}$$

which is also known as momentum update.

3) Target Policy Smoothing Regularization.: To prevent the occurrence of too many errors, we introduce a delay to the update process. Furthermore, we consider if there is any way to decrease the magnitude of errors themselves. In this case, we first need to identify the origin of the errors [18], [19].

The source of the mistake lies in the bias induced by the estimation of the value function. A usual method for eliminating the bias in the estimation in machine learning is to regularize the parameter updates. Following this line of thinking, we can apply this approach to reinforcement learning as well.

A natural concept in reinforcement learning is that similar actions should possess a comparable value. To ameliorate errors, one viable approach is to apply value smoothing to a localized region around the targeted action within the action space [19]. One plausible strategy entails adding a certain level of noise to the Q-value of the target action ϵ.

$$y = r + \gamma Q_{\theta'}\left(s', \pi_{\phi'}\left(s'\right) + \epsilon\right) \epsilon^{\sim} \operatorname{clip}(N(0, \sigma), -c, c) \tag{8}$$

where c is the parameter governing the magnitude of error fluctuations. The noise present here can be perceived as a form of regularization, which renders the value function update process more consistent.

B. Swap Softmax

The Softmax function, also referred to as the normalized exponential function, is a mathematical concept originating from probability theory and its affiliated domains. It serves as a generalized version of the logistic function [20]. This method has the capacity to normalize a given dataset to a set of decimal numbers ranging from 0 to 1, with a resulting sum of 1. The formula for this function is usually given by the following equation:

$$\sigma(\mathbf{z})j = \frac{e^{zj}}{\sum k = 1^K e^{zk}} \quad \text{for}; j = 1, \cdot, K; . \tag{9}$$

Softmax is capable of computing the relative weights of a sequence of values according to their respective magnitudes, as demonstrated in Fig. 1. It should be noted that the magnitude

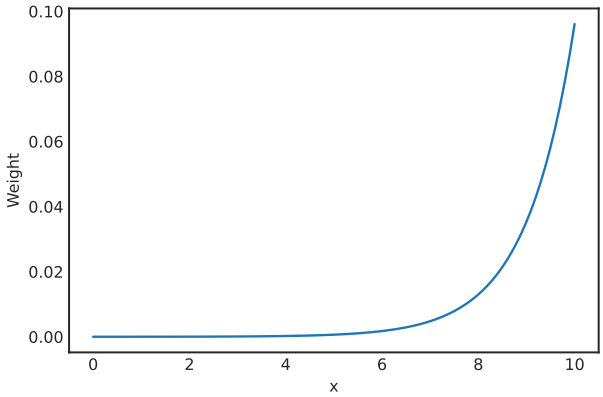

Fig. 1. A series of values from 0 to 10 and their Softmax mapping.

of the values has an impact on their weight distribution, meaning that if a set of numbers is multiplied by a certain factor, the weight of the larger values will also become greater.

The Softmax-based attention mechanism is widely adopted for various Artificial Intelligence tasks. In particular, for Reinforcement Learning, employing Softmax can also be very beneficial. As introduced in Section III-A, two sets of target networks are used to calculate the target values, and a conservative approach is taken by choosing the lower of these two values. However, such an approach would limit the exploration capability of the agent. Consequently, the Softmax function is employed to merge two target networks and estimate their corresponding target values. More specifically, the Softmax function is utilized to compute the weights of the two Q values, and subsequently these weights are utilized to calculate the

weighted sum of the two values as per the ensuing expression, thereby producing the target Q values:

$$\begin{aligned} w =& softmax(Q'_1, Q'_2) \\ Q' =& w[1]Q'_1 + w[2]Q'_2 \end{aligned} \tag{10}$$

Nevertheless, it is preferable to underestimate than to overestimate, so we have added a swap operation to ensure that smaller values are given more weight:

$$Q' = w[2]Q'_1 + w[1]Q'_2 \tag{11}$$

We refer to this operation as Swap Softmax, and it does not necessitate additional extensive computational effort.

C. Implementation

The crux of our algorithm is the incorporation of the attention mechanism, for which we use the Swap Method outlined above.

Algorithm 1 Swap Softmax TD3

for $t = 1$ to T **do**
  Take an action a entails incorporating exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$ base on π
  Obtain a reward r from the environment and undergo a transition to a new state s'
  Save transition tuple (s, a, r, s', d) in $\mathcal{B}$
  Randomly select a batch of transitions $\{(s, a, r, s', d)\}$ from $\mathcal{B}$
  Generate a noise $\epsilon \sim \mathcal{N}(0, \bar{\sigma})$
  $\hat{a}' \leftarrow \pi(s'; \phi^-) + \text{clip}(\epsilon, -c, c)$
  $\delta =$ Swap Softmax$[Q'_1(s', \hat{a}'), Q'_2(s', \hat{a}')]$
  $y \leftarrow r + \delta$
  Update the parameter θ_i of critic networks by using Bellman loss: $\sum_s (Q_i(s, a) - y)^2 / N$
  if t % d == 0 **then**
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)\big|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

Through numerous experiments, we have observed a common trend: it is beneficial to reduce the estimate at the initial stages of training. Therefore, in certain circumstances, we can incorporate a parameter to reduce the initial estimates and then gradually restore them to their original values later.

IV. Experiments

A. Experimental setup

In this section, our algorithm is employed to tackle one of the most difficult scenarios in the reinforcement learning continuous control problem. MuJoCo [11] is an accessible open-source physics engine intended for the execution of simulation experiments in control-oriented fields by researchers.

To ensure fairness of the game, every intelligence was allowed to take 1,000,000 actions and the same update frequency was used. Moreover, the actor and critic networks are implemented as multilayer perceptrons, which are proficient in processing multi-dimensional data. The neural network was uniformly activated with the Relu activation function, which is widely adopted nowadays. Since the agent was not given any prior knowledge, it was essentially blind in the early stages of training, thus learning in this period was not an effective strategy. To overcome this, random strategies were employed to explore the environment at the beginning of the training, and only after the intelligence had acquired some understanding of the external environment, was it able to learn more effectively and avoid local optima. Furthermore, a 25,000-step warm-up process was uniformly applied. The algorithm has a number of hyperparameters, which are described in detail in Table I.

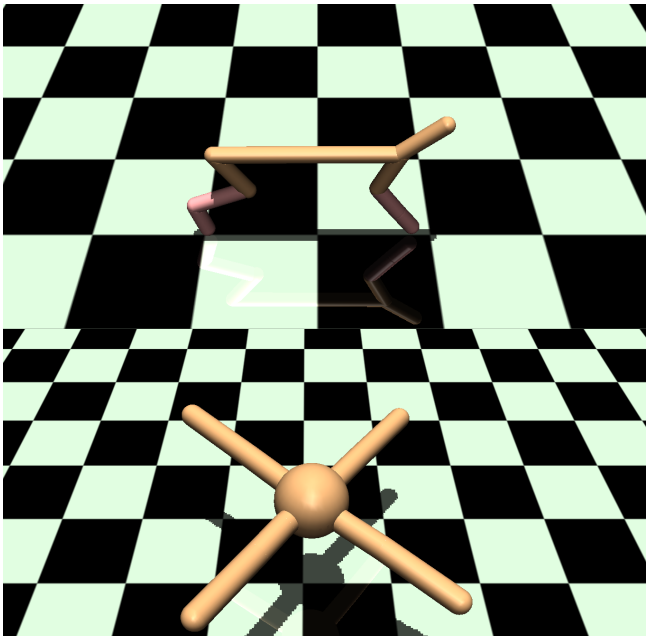

Fig. 2. MuJoCo simulation environments: the top is HalfCheetah with 7-dimensional action space and 17-dimensional state space and the bottom is Ant with 8-dimensional action space and 111-dimensional state space

Reinforcement learning carries a great deal of uncertainty, so a single experiment hardly provides an accurate indication of the algorithm's robustness. As a result, we conduct multiple experiments and take the average value in each situation. Furthermore, in order to guarantee the reproducibility of experimental results, we utilized identical random seeds at all instances in which random numbers may be introduced.

Different parameters of deep learning can have drastically distinct performance, thus it is of utmost importance to guarantee the conformity of these parameters.

B. Experimental results

We mainly employ the Ant and HalfCheetah of MuJoCo to assess the proposed algorithm.

The results of our experiments conducted within the Ant environment can be observed in Fig. 3. Our proposed algorithm demonstrated a significantly faster convergence speed and higher average reward than TD3. The highest average

TABLE I
HYPERPARAMETERS OF OUR PROPOSED METHOD

Shared hyperparameter	Value
Start Timesteps	25e3
Max Timesteps	1e6
Policy Freq	2
Expl Noise	0.1
Batch Size	256
Discount	0.99
Tau	0.005
Policy Noise	0.2
Noise Clip	0.5

reward of our proposed algorithm was approximately 5315, while that of TD3 was only around 4372, representing an improvement of around 21%. This implies that our agent was able to better perform the target task.

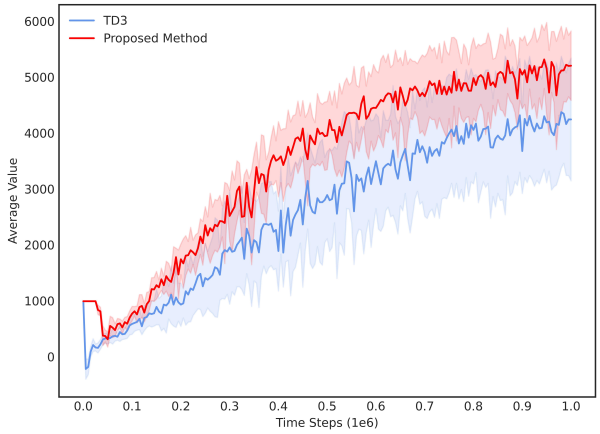

Fig. 3. Average value for Ant during learning process

Furthermore, the proposed algorithm displays substantially superior robustness performance, as determined by the standard deviation of the training process, as illustrated in Fig. 4. The largest standard deviation exhibited by TD3 amounted to roughly 1418, whereas that of the proposed algorithm was approximately-only 985. The experimental results demonstrate that the proposed algorithm consistently exhibited lower standard deviation values during the entire learning process, indicating that the agent successfully accomplished the task with superior efficiency.

The outcomes of our experimental investigations carried out within the HalfCheetah environment are presented in Fig. 5. We employed the starting shrinkage method mentioned earlier in HalfCheetah. This attention mechanism enlarges the value estimate earlier than TD3, and then gradually reduces it and returns to the original level, thus enhancing the performance of the model. The highest average reward of our proposed algorithm was 10827, significantly outperforming TD3 at 9636, representing a remarkable 1000-point improvement.

We also assess the stability of our approach using the same methodology. The remarkable advantage of the attention mechanism is that it can focus on the more critical components

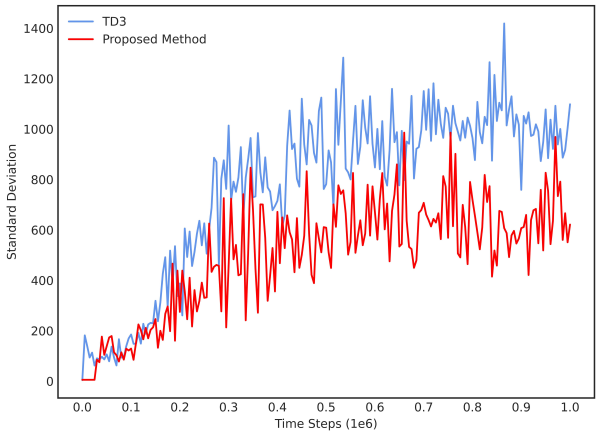

Fig. 4. Standard deviation for Ant during learning process

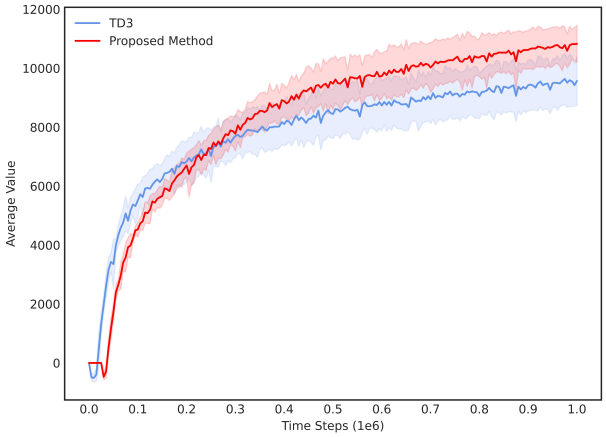

Fig. 5. Average value for HalfCheetah during learning process

and diminish the effect of irrelevant data, hence reducing the fluctuation of the data. Hence, it is conceivable to considerably reduce the instability of the algorithm with respect to the values, namely by decreasing the standard deviation

V. CONCLUSION

This paper presents an introduction of a Softmax-based attention mechanism to reinforcement learning algorithms, which effectively tackles the prevalent problems of over- and under-estimation in the continuous control field. The proposed algorithm demonstrates notable performance enhancement without necessitating any supplementary computational resources. Specifically, we use the Swap Softmax method to combine two target values, thus making the final target estimation more accurate. To appraise the efficacy of our proposed algorithm, we conducted experiments within the prevalent Gym MuJoCo simulation environment. The results indicate that our proposed algorithm achieved a higher average value with guaranteed consistent parameters. Furthermore, the introduced attention mechanism directs focus to salient data components, thereby mitigating the influence of data distribution on the model and reinforcing the robustness of the

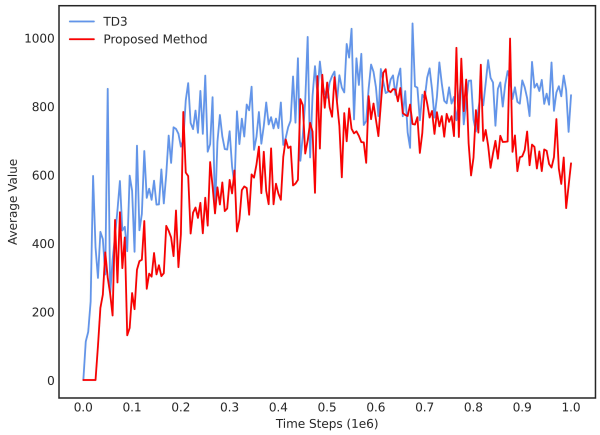

Fig. 6. Standard deviation for HalfCheetah during learning process

proposed algorithm's performance. Additionally, as the robustness of reinforcement learning algorithms has historically been weak, it is essential to ascertain the stability of the algorithm while guaranteeing performance improvement.

The attention mechanism facilitates the most efficient utilization of data; however, in this paper, we resort to a dual-network configuration. If computational consumption is not a concern, deploying additional networks might produce even better outcomes. We shall leave this thought for further investigation.

REFERENCES

[1] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual* (M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds.), pp. 15084–15097, 2021.

[2] M. M. Afsar, T. Crump, and B. H. Far, "Reinforcement learning based recommender systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 145:1–145:38, 2023.

[3] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Reinforcement learning with prototypical representations," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 11920–11931, PMLR, 2021.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. 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," *Nat.*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[7] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Networks*, vol. 9, no. 5, pp. 1054–1054, 1998.

[8] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591, PMLR, 2018.

[9] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA* (D. Schuurmans and M. P. Wellman, eds.), pp. 2094–2100, AAAI Press, 2016.

[10] L. Pan, Q. Cai, and L. Huang, "Softmax deep double deterministic policy gradients," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[11] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[12] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, vol. 32 of *JMLR Workshop and Conference Proceedings*, pp. 387–395, JMLR.org, 2014.

[13] S. Mannor and J. N. Tsitsiklis, "Mean-variance optimization in markov decision processes," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011* (L. Getoor and T. Scheffer, eds.), pp. 177–184, Omnipress, 2011.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, 2017.

[15] H. van Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 2613–2621, Curran Associates, Inc., 2010.

[16] V. R. Konda and J. N. Tsitsiklis, "Onactor-critic algorithms," *SIAM J. Control. Optim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[17] R. Fox, A. Pakman, and N. Tishby, "Taming the noise in reinforcement learning via soft updates," in *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, June 25-29, 2016, New York City, NY, USA* (A. Ihler and D. Janzing, eds.), AUAI Press, 2016.

[18] H. van Seijen, H. van Hasselt, S. Whiteson, and M. A. Wiering, "A theoretical and empirical analysis of expected sarsa," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2009, Nashville, TN, USA, March 31 - April 1, 2009*, pp. 177–184, IEEE, 2009.

[19] O. Nachum, M. Norouzi, G. Tucker, and D. Schuurmans, "Smoothed action value functions for learning gaussian policies," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (J. G. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 3689–3697, PMLR, 2018.

[20] I. Kouretas and V. Paliouras, "Simplified hardware implementation of the softmax activation function," in *8th International Conference on Modern Circuits and Systems Technologies, MOCAST 2019, Thessaloniki, Greece, May 13-15, 2019*, pp. 1–4, IEEE, 2019.