WIDE-SENSE STATIONARY POLICY OPTIMIZATION WITH BELLMAN RESIDUAL ON VIDEO GAMES

Chen Gong †, ‡, ∗, *Qiang He*†, ‡, ∗, *Yunpeng Bai*†, ‡, *Xinwen Hou*†, ✉, *Guoliang Fan* †, *Yu Liu*†

†Institute of Automation, Chinese Academy of Sciences;
‡University of Chinese Academy of Sciences, School of Artificial Intelligence;
{gongchen2020, heqiang2019, baiyunpeng2020, xinwen.hou, guoliang.fan, yu.liu}@ia.ac.cn;

ABSTRACT

Deep Reinforcement Learning (DRL) has an increasing application in video games. However, it usually suffers from unstable training, low sampling efficiency, etc. Under the assumption that Bellman residual follows a stationary random process when the training process is convergent, we propose the **W**ide-sense **S**tationary **P**olicy **O**ptimization (WSPO) framework, which leverages the Wasserstein distance from the Bellman Residual Distribution (BRD) between two adjacent time steps, to stabilize the training stage and improve the sampling efficiency. We minimize the Wasserstein distance with Quantile Regression, where the specific form of BRD is not needed. Finally, we combine WSPO with Advantage Actor-Critic (A2C) algorithm and Deep Deterministic Policy Gradient (DDPG) algorithm. We evaluate WSPO on Atari 2600 video games and continuous control tasks, illustrating that WSPO compares or outperforms the state-of-the-art algorithms we tested.

Index Terms— Video Game; Reinforcement Learning; Quantile Regression; Bellman residual; Wasserstein Distance

1. INTRODUCTION

DRL has been widely applied in various fields, and shows superior performance on video games, such as Atari 2600 games [1], Go [2], StarCraft II [3] and so on. DRL trains a state-to-action mapping – Agent, which outputs an action or a probability distribution of candidate action based on the current state, in order to maximize the accumulative rewards. Besides, combined with the neural networks' strong feature-extraction ability, DRL has an outstanding solution for video games – a sequential decision problem [4]. However, DRL still suffers from unstable training, low sampling efficiency, slow convergence, and so on. For example, even in the later stage of training, total rewards can fluctuate significantly, which is hardly appeared in supervised learning. We usually consume millions of sample frames for training [5], revealing the low sample efficiency. These problems cause enormous computing resource consumption and time cost. Especially, in some tasks with complex video game environments such as Dota, Starcraft, it is hardly possible to learn a robust agent due to the unbearable computational costs and difficult convergence caused by unstable training.

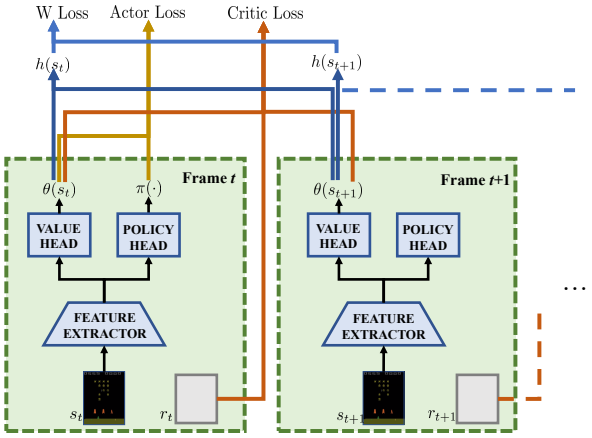

Fig. 1. The architecture of Wide-sense Stationary Policy Optimization (WSPO)

A variety of algorithms have been proposed to suppress the unstable training or incorrect updates direction. For example, experience replay, the key of the powerful off-policy algorithms [2, 6, 7], where the samples can be batched [8, 9] or randomly sampled [6] from different time steps. However, experience replay consumes much memory and can only be applied to off-policy algorithms, which can learn from the data generated by another policy. Proximal Policy Optimization (PPO) [10] and Trust Region Policy Optimization (TRPO) [11] constrain the update amplitude of policy between two adjacent time steps to stabilize the training stage. In complex video games, it is necessary to reduce the instability of the algorithms from the Bellman residual view.

*THESE TWO AUTHORS CONTRIBUTED EQUALLY.

THIS PROJECT WAS SUPPORTED BY THE NATIONAL NATURAL SCIENCE FOUNDATION OF CHINA (61379099), FOUNDATION STRENGTHENING KEY PROJECT 021-00, 034-00, AND BASIC SCIENTIFIC RESEARCH PROGRAM OF CHINA B022.

978-1-6654-3864-3/21/$31.00 ©2021 IEEE

Our contributions are as follows. Firstly, we find that the Bellman residual has large fluctuations, even if the training is convergent. In theory, the Bellman residual follows a stationary random process when the training process is convergent. As the uncertainty of video game environments, the Bellman residual is impossible to follow the ideal solution. Based on the above analysis, the BRD between two adjacent time steps should follow a stationary random process. Secondly, we propose the WSPO framework, which optimizes the Wasserstein distance of BRD between two adjacent time steps. The architecture of WSPO is presented in Figure 1. Compared with Kullback-Leibler (KL) divergence, the Wasserstein distance does not suffer from disjoint-support issues [12]. Besides, Quantile Regression (QR) is utilized to approximate the Wasserstein distance, where we do not need to know the specific form of BRD, which is intractable in general [13]. Finally, We combine WSPO with the commonly used RL algorithms, A2C, and DDPG. We perform experiments on Atari games and PyBullet continuous control tasks respectively, which shows that WSPO compares or outperforms the state-of-the-art algorithms we tested. We also verify that WSPO does enjoy the stable value function learning process and the policy learning process.

2. RELATED WORK

Recently, the applications of DRL in video games have been increased gradually [1–3, 14]. The Atari 2600 video game has become one of the most popular environments for evaluating RL algorithms. Value function optimization is an essential part of DRL. Deep Q-Networks (DQN) [2] approximates the action-value function by deep neural networks. Dueling DQN [15] estimates the state value function and the advantage function respectively instead of approximating the Q function directly. Bellemare et al. break the thought of conventional RL and propose Distributional RL [16], which directly optimizes the distribution of value function. Dabney et al. use Wasserstein distance to measure the loss function and approximates by quantile regression [13], which we get insight from. What's more, the DQN [2], Double DQN [17], Prioritized Experience Replay, Dueling Networks [15], Noisy DQN [18] are also the common used algorithms on video game. David et al. integrate them and propose Rainbow [5], which realizes amazing performance in Atari games.

If the action space is very large or continuous, the value-based methods, such as Q-learning, cannot cope with this situation. Therefore, Sutton et al. introduce the policy gradient method with function approximation tools such as neural networks [19]. Policy Gradient methods are extended to a wide area. Silver et al. propose a Deterministic Policy Gradient method (DPG) [20] to optimize the expected reward which uses a deterministic policy. Inspired by the DQN, Lillicrap et al. combine DPG algorithm with deep neural networks and propose the Deep Deterministic Policy Gradient (DDPG) algorithm [7]. A well-known issue in the DPG algorithm is the lack of exploration capability. Lillicrap et al. claim that adding noise drawn from Ornstein-Uhlenbeck process [21] to actions can help DDPG explore better. TD3 (Twin Delayed Deep Deterministic Policy Gradient) [22] algorithm is an improved version of the DDPG algorithm, also deterministic. Although the TD3 algorithm achieves a better performance than the DDPG algorithm, the TD3 algorithm still estimates the value function inaccurately. Because of the existence of the mine operator, the estimation of value function tends to underestimated, thus harming the performance [23].

3. PRELIMINARIES

We consider the standard RL paradigm as an agent interacting with an environment in discrete time steps. We formalize the standard reinforcement learning as a Markov Decision Process (MDP), which is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \rho_0, \gamma)$, i.e., a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a reward function: $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a transition probability $p(s_{t+1}|s_t, a_t)$ with a certain state and action pair (s_t, a_t), an initial state distribution ρ_0, and a discount factor: $\gamma \in [0, 1)$.

The policy $\pi(a|s)$ is a mapping from state to an action or an action distribution. Every episode starts with an initial state $s_0 \sim \rho_0$. In each time step t, the agent generates an action based on current state, i.e., $a_t \sim \pi(\cdot|s_t)$. Then, the agent receives a reward $r(s_t, a_t)$ from environment and a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$ until a termination state is encountered. The return is defined as $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$. The optimization goal is to find a policy π which maximizes the $\mathbb{E}_{\tau\sim\pi}[R_0|s_0]$, where τ is a trajectory. State value function is defined as $V^\pi(s_t) = \mathbb{E}_{\tau\sim\pi}[R_t|s_t]$, a prediction for the further reward, which represents the expected return induced by policy π in state s. Similarly, the action-state value function is defined as $Q^\pi(s_t, a_t) = \mathbb{E}_{\tau\sim\pi}[R_t|s_t, a_t]$. Besides, the advantage function $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$ denotes whether the action a_t is better or worse than the average action which the policy π takes in the state s_t. In DRL, the policy function and value function are usually represented by neural networks: $V^\pi(s; \theta) \approx V^\pi(s), \pi(a|s; \theta) \approx \pi(a|s)$, parameterized by θ.

4. METHOD

This section is organized as follows. Firstly we introduce the motivation and give WSPO's framework. Secondly, we explain how to optimize WSPO with QR. Finally, we extend WSPO to off-policy algorithms.

4.1. Wide-sense Stationary Policy Optimization

In deep actor critic algorithms, the state value function $V^\pi(s)$, a prediction for the further reward, represents the expected return for following policy π in state s. The Bellman equation defines the tie to the adjacent time step value function:

$V^\pi(s_t) = \mathbb{E}_{\tau\sim\pi}[r_{t+1} + \gamma V^\pi(s_{t+1})]$. Besides, Bellman residual is a powerful tool to calculate value function:

$$\mathcal{B}(s_t;\theta) = r_{t+1} + \gamma V^\pi(s_{t+1};\theta) - V^\pi(s_t;\theta). \quad (1)$$

Note that the state s_{t+1} follows a conditional distribution $p(s_{t+1}|s_t, a_t)$, so we can regard the Bellman residual in s_t as a distribution $U(s_t;\theta)$, called Bellman Residual Distribution (BRD). Let $\mathcal{U}$ be the space of BRD with finite moments:

$$\mathcal{U} = \{U : \mathcal{X} \to \mathscr{P}(\mathbb{R}) \mid \mathbb{E}[|U(x)|^p] < \infty, \forall(x), p \geq 0\} \quad (2)$$

Theoretically, when the training process is convergent, the Bellman residual follows a stationary random process. Because of the uncertainty of the environment, the Bellman residual is impossible to meet the ideal situation. However, the BRD between two adjacent time steps should be similar. Therefore, we consider the framework that optimizes the Wasserstein distance between two adjacent time steps to stabilize the training stage, called Wide-sense Stationary Policy Optimization. The p-Wasserstein distance of two distributions U, V is defined as

$$W_p(U,V) = \left(\int_0^1 \left|F_U^{-1}(x) - F_V^{-1}(x)\right|^p dx\right)^{1/p}, \quad (3)$$

where for a random variable U, the inverse Cumulative Distribution Function (CDF) F_U^{-1} of U is defined by $F_U^{-1}(x) := \inf\{u \in \mathbb{R} : x \leq F_U(u)\}$, and $F_U(u)$ is the CDF of U. Compared with the KL divergence, the Wasserstein distance is a true probability measure, thus is suitable for domains where an underlying similarity in outcome is more important than exactly likelihoods [13, 16]. Let $p = 1$. The 1-Wasserstein distance of the BRD between two adjacent time is defined by

$$W_1(U(s_t), U(s_{t+1})) = \int_0^1 \left|F^{-1}_{U(s_t)}(x) - F^{-1}_{U(s_{t+1})}(x)\right| dx. \quad (4)$$

Assuming that the loss term of vanilla RL algorithms is $\mathcal{L}(\theta)$. We utilize the Eq (4) as a regularization of vanilla RL algorithms. The loss term of WSPO is defined by

$$Loss_{\text{WSPO}} = \mathcal{L}(\theta) + \lambda W_1(U(s_t;\theta), U(s_{t+1};\theta)). \quad (5)$$

4.2. Optimized with Quantile Regression

Obviously, the $\nabla_\theta \mathcal{L}(\theta)$ is tractable, but it is difficult to calculate the $\nabla_\theta W_1(U(s_t;\theta), U(s_{t+1};\theta))$. We approximately minimize the Wasserstein distance by QR method referencing distributional RL view. We recommend readers refer to [13] for details.

We denote the CDF with quantile distribution by $\{\tau_1, \cdots, \tau_N\}$, where $\tau_i = \frac{i}{N}$ for $i = 1, \ldots, N$. Let $\theta : \mathcal{X} \to \mathbb{R}^N$ be some parameterized models, where N is the number of quantile. A quantile distribution Z_θ maps a state s to a uniform probability distribution supported on $\{\theta_1(s), \cdots, \theta_N(s)\}$, and $\theta_i(s) = F_Z^{-1}(\tau_i)$, i.e.,

$$Z_\theta(s) = \frac{1}{N}\sum_{i=1}^N \delta_{\theta_i(s)}, \quad (6)$$

where δ_x is a Dirac at $x \in \mathbb{R}$. In this paper, we approximate the quantile distribution of value function – Z_θ by a neural network, instead of using the expectation form of the value function in standard RL. Then the Bellman operator is defined as

$$\mathcal{T} Z_\theta(s_t) = r_t + \gamma Z_\theta(s_{t+1}), \quad (7)$$

where $s_{t+1} \sim p(\cdot|s_t, a_t)$, $a_t \sim \pi(\cdot|s_t)$. The quantile distribution is denoted as $M_\theta(s_t)$ which maps a state s to a uniform probability distribution supported on $\{h_i(s)\}_{i=1}^N$, and $h_i(s) = F^{-1}_{U(s)}(\tau_i)$. The $M_\theta(s_t)$ is calculated by

$$M_\theta(s_t) = r_t + \gamma Z_\theta(s_{t+1}) - Z_\theta(s_t). \quad (8)$$

Then the Wasserstein distance in Eq (4) can be denoted by *quantile Huber loss* [13],

$$W_1(U(s_{t+1}), U(s_t)) = \frac{1}{N'}\sum_{i=1}^N \sum_{j=1}^{N'} \rho^\kappa_{\tau_i}(h_j(s_{t+1}) - h_i(s_t)), \quad (9)$$

in which $\rho^\kappa_\tau(x) = |\tau - \delta_{\{x<0\}}|\mathcal{L}_\kappa(x)$, with

$$\mathcal{L}_\kappa(x) = \begin{cases} \frac{1}{2}x^2, & \text{if } |x| \leq \kappa \\ \kappa\left(|x| - \frac{1}{2}\kappa\right), & \text{otherwise} \end{cases} \quad (10)$$

where κ is a hype-parameter. We can leverage the Wasserstein distance by optimizing Eq (9). Especially, this loss gives unbiased sample gradients, so that we can find the minimizing $\{\theta_i\}_{i=1}^N$ by stochastic gradient descent. We summarize the WSPO algorithm on Algorithm 1.

4.3. Extend WSPO to off-policy setting

We extend the Bellman residual constraint to the off-policy setting. Under the on-policy settings, we can naturally get the distribution of Bellman residual at different time steps. However, different from the on-policy setting, off-policy cannot naturally obtain the BRD of the last time step. Note that the Bellman distribution at the current time step can be calculated. The Bellman residual distribution at the last optimization time step is tractable under the off-policy setting. So we extend the concept of Bellman residual constraint to an off-policy setting. We argue that the distribution of Bellman residual should be a wide-sense stationary process for adjacent time steps under the off-policy continuous control tasks. The benefits of such promotion are apparent. We can stabilize the learning process of the value function, i.e., the same transition in close proximity can produce similar output.

Algorithm 1: Wide-sense Stationary Policy Optimization (WSPO)

Requires: θ: initialize network parameters; λ: regularization parameter; ϵ: learning rate; N: number of quantile; s_0: initial state; T: the counter; κ: hyperparameter

repeat

  Sample a batch of data $\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{n}$ from enviroment

  for $k = 0, 1, \cdots, n-2$ **do**

    # Calculate the quantile distribution of BRD

    for $m = 0, 1, \cdots, N-2$ **do**

      $h_m(s_k) = r_k + \gamma\theta_m(s_{k+1}) - \theta_m(s_k)$

      $h_m(s_{k+1}) = r_{k+1} + \gamma\theta_m(s_{k+2}) - \theta_m(s_{k+1})$

    end

    # Calculate the Wasserstein distance

    $W_1(U_\theta(s_k), U_\theta(s_{k+1})) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\rho_{\tau_i}^{\kappa}(\theta_j^M(s_{k+1}) - \theta_i^M(s_k))$

    $\Delta\theta \leftarrow \frac{1}{n-1}\left[\nabla_\theta\mathcal{L}(\theta) + \lambda\nabla_\theta W_1(U_\theta(s_k), U_\theta(s_{k+1}))\right]$

    $\theta \leftarrow \theta + \epsilon\Delta\theta$

  end

  $T \leftarrow T + 1$

until $T > T_{max}$;

5. EXPERIMENTS

To evaluate our WSPO, we combine on-policy algorithm A2C [24] and off-policy algorithm DDPG [7] with WSPO on video game Atari 2600 and PyBullet continuous control tasks, comparing with well-known baseline algorithms. We analyze the training stability, sampling efficiency, and performance to show the validity of WSPO. We also investigate the impact of the regularization coefficient λ in our framework. Note that all the environments we chose are the most commonly used for evaluating RL algorithms.

5.1. Evaluation on Atari 2600

We combine WSPO with A2C algorithm and conduct experiments on 15 video game Atari 2600 (NoFrameskip-v4). In WSPO, the number of quantiles N is 32, and the regularization coefficient λ takes 0.001. If computation resources are enough, readers can try to take more quantiles. For all the experiments, the total number of sampling is 14 million. There are five agents carried out simultaneously for one seed, and each agent uses eight steps return for gradient estimation. Besides, all curves are averaged over three random seeds. To ensure the fairness of comparison, except for the new parameters introduced by WSPO, all the other hyper-parameters are the same as follow link[1], and the hyper-parameters of quantile regression are the same as [13].

On Atari games, we test three algorithms, **vanilla A2C** [24], **WPSO combining A2C**, **SVA2C** [25]. The code of SVA2C follows the link[2]. The experimental results of 15 Atari games presented in Figure 2 show that WSPO significantly improves stability and sample efficiency on most Atari games. Note that WSPO only adds a small amount of computational complexity than vanilla RL algorithms. At the same time, the SVA2C consumes many redundant computing resources. What's more, WSPO optimizes the Wasserstein distance, which is a true probability distance, and considers both the probability and the distance between various outcome events, unlike the KL divergence in the SVA2C method. In Figure 5, we compare the Bellman residual performance. The Bellman residual of WSPO is smoother, which has a more stable training process. Combining the above experimental results, we can conclude that WSPO stabilizes the RL algorithm training process by stabilizing the learning stage of the state value function.

5.2. Evaluation on PyBullet continuous control tasks

Following the aforementioned idea, we build an auxiliary critic network to record the critic's parameters in the last update. Similar to discrete settings, we also use quantile networks for the critic to calculate the Bellman residual directly. Note that since we have introduced a distributional critic, we can use the quantile Huber loss to update the critic. Similarly, we can recover the Q value from the critic to perform a conventional Bellman update. In experiments, we find no apparent difference in the performance of these two update styles.

Our implementation is based on the DDPG algorithm. When the agent is updated, mini-batch samples are collected from the replay buffer and then calculate the target. Then, we can calculate the critic loss function and BRD. Finally, the constraints are added together to form the loss function of the critic. For the policy network, we use the DPG update method. We generate new actions from the actor and then maximize the Q value. We update the policy through the gradient chain rule. To evaluate our algorithm, we test the WSPO algorithm on the PyBullet suite of OpenAI gym continuous control tasks. Considering the recent concerns about the algorithm's reproducibility, we implement the WSPO algorithm without any engineering skills. We test the WSPO algorithm on four continuous control tasks. The results show WSPO compares or outperforms other algorithms we tested. We compare the commonly used baseline algorithm PPO, DDPG, and TRPO, which are viewed as a strong baseline on continuous control tasks. Note that although we have not fine-tuned the hyperparameters of the algorithm, our algorithm is relatively good. We show the experimental results in Figure 3.

[1] https://github.com/openai/baselines

[2] https://github.com/2019ChenGong/SVA2C

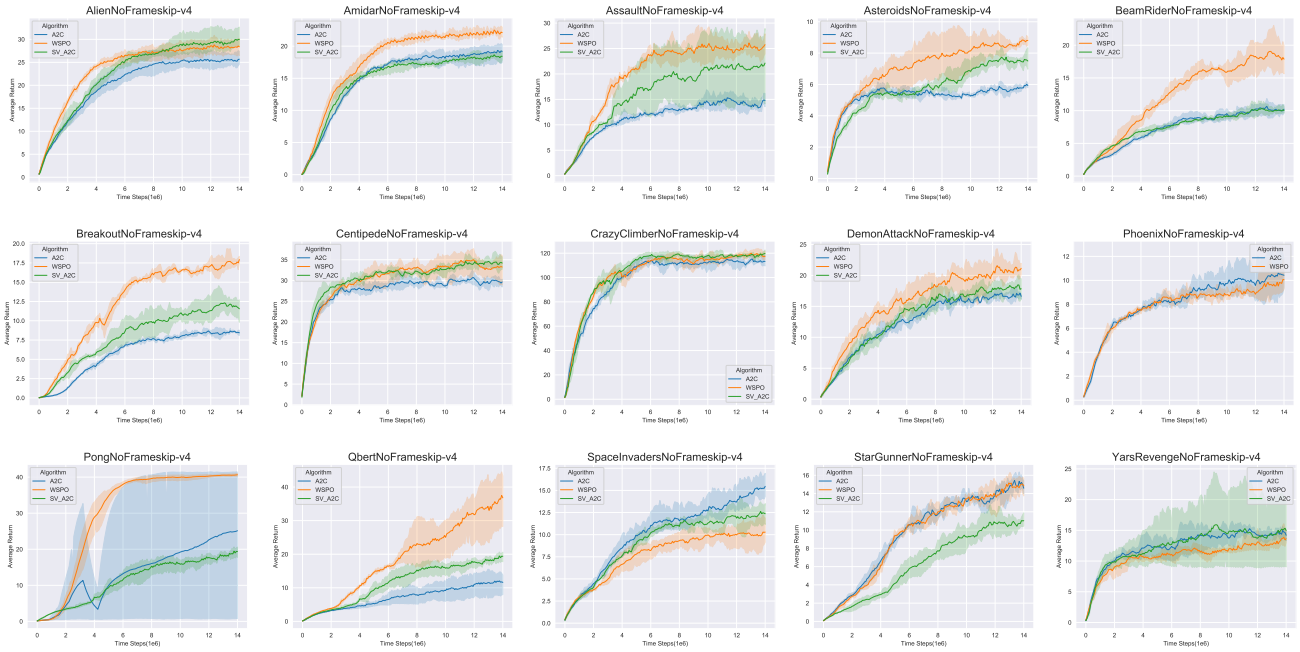

Fig. 2. Performance curves with vanilla A2C, SV-A2C and WSPO for 15 Atari games. The exponential moving average to each curve with 3 steps. The solid line is the mean and the shadow part denotes the standard deviation.

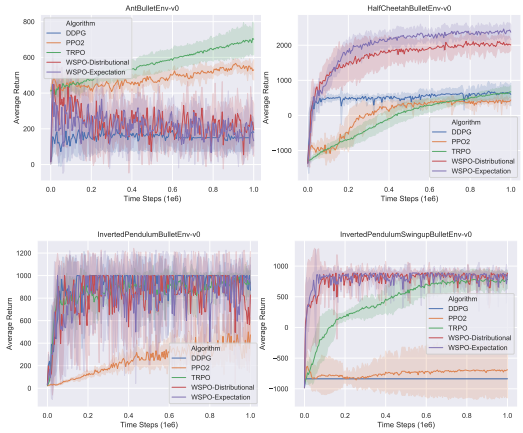

Fig. 3. Performance curves for the PyBullet suite of OpenAI gym continuous control tasks. The shaded region represents a standard deviation of the average evaluation over five seeds.

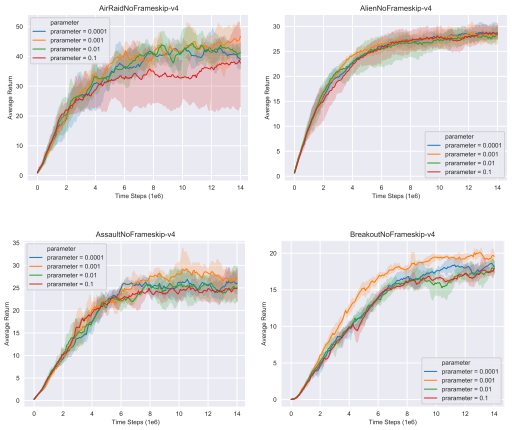

Fig. 4. Performance curves with WSPO for different Regularization Coefficient λ, 0.1, 0.01, 0.001, 0.001, in Atari games.

5.3. Results of Sensitivity to Regularization Coefficient λ

In this section, we investigate the influence of the regularization coefficient λ value for our framework, which is the unique new parameter in WSPO. We perform experiments with $\lambda = \{0.1, 0.01, 0.001, 0.0001\}$ on four Atari games respectively. Results show in Figure 4 that λ setting as 0.001 obtain better performance. In our experiments, λ is set as 0.001 in all games. However, better results may be obtained if the readers try different hyper-parameters in certain environments.

6. CONCLUSION

In this work, we investigate the problem of unstable training and low sampling efficiency of RL algorithms on video games, proposing Wide-sense Stationary Policy Optimization (WSPO) framework. In WSPO, we leverage the BRD's Wasserstein distance from two adjacent time steps, thus stabilizing the train-

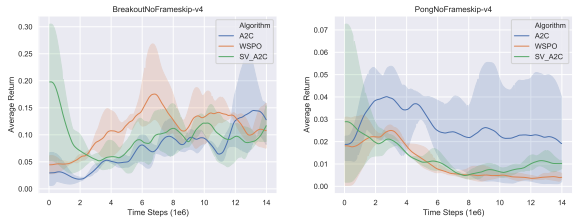

Fig. 5. Bellman residual curves with vanilla A2C, SV-A2C and WSPO for two Atari games.

ing stage and improving the sample efficiency. Besides, the Wasserstein distance is optimized with quantile regression. We combine WPSO with A2C and DDPG algorithms and conduct experiments on Atari 2600 games and PyBullet continuous control tasks. Compared with baseline algorithms, WSPO presents a significant improvement in most environments. As WSPO is independent of the algorithms' specific expression, it can be applied to all model-free RL algorithms related to the value function optimization for solving more complex video games in the future.

7. REFERENCES

[1] David Silver, Aja Huang, Chris J. Maddison, et al., "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484, 2016, Publisher: Nature Publishing Group.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] Oriol Vinyals, Igor Babuschkin, et al., "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, pp. 350–354, 11 2019.

[4] Jian Zhao, Wengang Zhou, Tianyu Zhao, Yun Zhou, et al., "State representation learning for effective deep reinforcement learning," 07 2020, pp. 1–6.

[5] Matteo Hessel, Joseph Modayil, et al., "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] Hado Van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[7] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, et al., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[8] Lasse Espeholt, Hubert Soyer, Remi Munos, et al., "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561*, 2018.

[9] Ofir Nachum, Shixiang Shane Gu, et al., "Data-efficient hierarchical reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 3303–3313.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, et al., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] John Schulman, Sergey Levine, Pieter Abbeel, et al., "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[12] Arjovsky Martin, Chintala Soumith, and Bottou Léon, "Wasserstein generative adversarial networks," 2017, vol. 70, pp. 214–223.

[13] Will Dabney, Mark Rowland, Marc G. Bellemare, et al., "Distributional reinforcement learning with quantile regression," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] Bin Wu, "Hierarchical macro strategy model for moba game ai," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 1206–1213.

[15] Ziyu Wang, Tom Schaul, Matteo Hessel, et al., "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[16] Marc G. Bellemare, Will Dabney, and Rémi Munos, "A Distributional Perspective on Reinforcement Learning," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, Sydney, NSW, Australia, 2017, ICML'17, pp. 449–458, JMLR.org.

[17] Hado van Hasselt, "Double Q-learning," in *Advances in Neural Information Processing Systems 23*, 2010.

[18] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, et al., "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[19] Richard S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[20] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller, "Deterministic policy gradient algorithms," 2014.

[21] George E Uhlenbeck and Leonard S Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, pp. 823, 1930.

[22] Scott Fujimoto, Herke Van Hoof, and David Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[23] Qiang He and Xinwen Hou, "Popo: Pessimistic offline policy optimization," *arXiv preprint arXiv:2012.13682*, 2020.

[24] Volodymyr Mnih, Adria Puigdomenech Badia, et al., "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[25] Chen Gong, Yunpeng Bai, Xinwen Hou, and Xiaohui Ji, "Stable training of bellman error in reinforcement learning," in *International Conference on Neural Information Processing*. Springer, 2020, pp. 439–448.