

Bootstrap Advantage Estimation for Policy Optimization in Reinforcement Learning

Md Masudur Rahman, Yexiang Xue

Department of Computer Science

Purdue University, West Lafayette, Indiana, USA

{rahman64, yexiang}@purdue.edu

Abstract—This paper proposes an advantage estimation approach based on data augmentation for policy optimization. Unlike using data augmentation on the input to learn value and policy function as existing methods use, our method uses data augmentation to compute a bootstrap advantage estimation. This Bootstrap Advantage Estimation (BAE) is then used for learning and updating the gradient of policy and value function. To demonstrate the effectiveness of our approach, we conducted experiments on several environments. These environments are from three benchmarks: Procgen, Deepmind Control, and Pybullet, which include both image and vector-based observations; discrete and continuous action spaces. We observe that our method reduces the policy and the value loss better than the Generalized advantage estimation (GAE) method and eventually improves cumulative return. Furthermore, our method performs better than two recently proposed data augmentation techniques (RAD and DRAC). Overall, our method performs better empirically than baselines in sample efficiency and generalization, where the agent is tested in unseen environments.

Index Terms—Deep Reinforcement Learning, Advantage Estimation, Generalization in Reinforcement Learning

I. INTRODUCTION

The policy gradient method directly involves learning policy function, which enjoys performance improvement in function approximation settings [1]. The policy gradient theorem gives a rather simple formulation of the gradient estimation, which gives an unbiased estimation [2]. However, it requires the return estimation of the entire trajectory, leading to very high variance. A commonly used technique to reduce variance is to use a baseline, which can help reduce variance without introducing bias. Several effective methods originated from this concept [3], [4]. An effective way is to use the value function as a baseline to indicate whether the action taken by the current policy is better than the average action taken in that state, which can be formulated as an advantage estimation. However, based on a single trajectory, the estimate can be local and have high variance. Thus we can add a truncated scenario where the value function can potentially give a global estimate. Combining these two is the Generalized advantage estimation [5] which shows strong empirical results [6].

However, due to procedural style content generation, the value estimation can be erroneous and give different advantage estimates even when the observation context changes. At the same time, the semantic meaning remains the same. As the procedural scenario can exist in a real-world scenario and might cause agent to perform sub-optimally [7], [8], thus this

advantage estimation can be problematic in those scenarios, resulting in poor sample efficiency. This issue might exist partly due to difficulty in reducing policy estimation (policy loss) and value function estimation (value loss).

This paper proposes Bootstrap Advantage Estimation (BAE), which calculates the advantage estimation by computing advantage estimation on the original and its transformed observations. We assume the transformation to be a semantic invariant; the reward semantic remains the same, but the contextual information can be changed. For example, if the background of a game is not relevant, then changing the background color from red to green can be a semantic invariant transformation. The ultimate goal is to train an agent to be robust against any such background change, which performs well in the blue background in this example. The transformed observation can be of any form; we experimented with data augmentation-based observation transformation (e.g., random crop, amplitude scaling). The intuition is that taking advantage of estimate over augmented data forces the advantage estimate to consider the error over many variations of the observations. We demonstrate our BAE on the policy gradient method (i.e., PPO [3]) and show a comparison over GAE-based estimation. We observed that our method BAE achieved better sample efficiency and zero-shot generalization in Procgen environments (starpilot and miner) with image-based observation.

In recent times, data augmentation demonstrated an effective choice in improving sample efficiency in high-dimensional observation space and improving generalization [9]–[11]. Though this process sometimes generates empirical success [10], such methods might lead to detrimental performance [11] as we observed in our experiments. To mitigate this issue, the DRAC [11] method suggests regularizing the policy network and value network by augmented observation and not using augmented data for policy training. In contrast, we propose a novel way to leverage data augmentation. Our method augmented observations for advantage estimation, one of the core components of many policy optimization algorithms (e.g., PPO). We conducted extensive experiments on six environments consisting of image and vector-based observation; and discrete and continuous action spaces. Our method falls in the general model-free on-policy category, and we experimented with Proximal Policy Optimization (PPO) [3] in this paper. In particular, our experiments on Procgen Starpilot and Miner environments demonstrate that our method can be beneficial in

the zero-shot generalization setup compared to baseline GAE [5], and two data augmentation techniques: RAD [10], and DRAC [11].

We further evaluated our method on several robotic locomotion tasks with the high-dimensional observation from Deepmind Control Suite [12]: Quadruped Run, and Cartpole - Three Poles; and PyBullet [13]: Minitaur and HalfCheetah.

In experiments, we observe that our method BAE performs better than baseline agents, including base PPO, RAD, and DRAC. Our method achieves a much lower loss for policy and value function estimation. Eventually, it performs better in sample efficiency and zero-shot generalization than baseline agents, including data augmentation. We observe that the baseline data augmentation methods (RAD and DRAC) sometimes worsen the base model performance. In contrast, our BAE method improves the performance in most tested environments and performs consistently with the base algorithm in other cases. These results show that our method BAE is more robust in performance compared to baseline data augmentation methods.

The source code of our method is available at <https://github.com/masud99r/bae>.

II. PRELIMINARIES AND PROBLEM SETTINGS

Reinforcement Learning We assume the task/environment is a Markov Decision Process (MDP) which is denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where \mathcal{S} is state space, \mathcal{A} is that action space, \mathcal{P} is the transition probability between states resulted by an action. In this setup, at every timestep t , the agent take an action $a_t \in \mathcal{A}$ in a state $s_t \in \mathcal{S}$ and the environment transition to next state $s_{t+1} \in \mathcal{S}$ determined by the transition probability $P(s_{t+1}|s_t, a_t)$. In a reinforcement learning framework, the goal of the agent is to learn a policy $\pi \in \Pi$ by maximizing the expected return in an MDP, where Π is a set of all possible policies and π is a single policy which is a mapping from state to action. The policy which achieves the highest expected return is optimal $\pi^* \in \Pi$.

Policy Gradient Proximal Policy Optimization (PPO) [3], a type of policy gradient method which achieved tremendous success and is popularly used in many setups because of its effective learning and simple implementation. However, the choice of implementation details might impact the performance of such algorithms in significant ways [6], [14], [15]. These implementation details consist of training individual components of the algorithms, such as learning value function, policy function, and advantage estimation. The following is the objective of the PPO [3].

$$\mathcal{L}_\pi = -\mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t \right] \quad (1)$$

, where $\pi_\theta(a_t|s_t)$ is the probability of choosing action a_t give state s_t at timestep t using the current policy parameterized by θ . On the other hand the $\pi_{\theta_{old}}(a_t|s_t)$ refer to the probabilities using an old policy parameterized by previous parameter values θ_{old} . The advantage A_t is an estimation which is the advantage of taking action a_t at s_t . A popular and effective

choice of estimating advantage using a value function is as follows (equation 2):

$$A_t = -V(s_t) + r_t + \gamma r_{t+1} + \dots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T). \quad (2)$$

Here $V(s)$ is a value function that gives the average future return under the underlying policy. The first term $V(s_t)$ is the value prediction at timestep t , and the rest of the terms except the last term in the equation is the discounted Monte Carlo Estimation which can be computed for a given episode from t to $T-1$ ($T > t$). The last term $V(s_T)$ is the value prediction at state s_T . Thus overall, this A_t represents how much the current action a_t is doing compared to the current value prediction. The more the advantage of action, the more the policy should weigh that action. This is done by multiplying $\pi_\theta(a_t|s_t)$ with A_t . The $\pi_{\theta_{old}}(a_t|s_t)$ in Equation 2 introduced due to importance sampling which allows estimating the advantage from the old policy samples. For details discussion, we refer the reader to [3], [4].

Value Function Estimation An effective value function estimation [3] is to regress value prediction with an advantage-based return estimation. Here the $V_{error} = |V_{prediction} - V_{Return}|$, where $V_{prediction}$ is the predicted value and the V_{Return} is value computed from the rewards $R = \sum_t r_t$ of sampled trajectories and advantage A . Thus, $V_{Return} = A + R$. Note that in this way, both the policy (in equation 1 and value function is dependent on the advantage estimation. Thus, an accurate advantage estimation should give us lower policy and value losses and thus a better performing policy.

Generalization in RL Now we turn attention to the scenarios where different episode varies by confounding features in the observation. Note that these confounders (also called context) impact the reward in the environment; however, they might misguide the agent to think otherwise. Due to nature, the agent might overfit the confounding features and fails to generalize to slightly modified test environments [7], [8]. For a details overview of generalization in reinforcement learning, we refer the reader to the servery papers [16].

Data augmentation in various forms has been leveraged [10], [11]. The general idea is to transform the observation so that the observation's semantic meaning remains the same but the contextual information changes. However, the context information is readily not available, and thus we need to impose various assumptions that certain transformations on the observation $s' = f(s)$ keep the reward semantic. For image-based observation, various image manipulation can be used, such as cropping, rotation, and color-jitter.

Advantage Estimation An essential component in policy training is to estimate the advantage. Generalized Advantage Estimation (GAE) [5] is a useful way to compute advantage, which combines the value function estimate and the Monte Carlo method. However, the data augmentation-based regularization approaches [10], [11] do not handle the case of advantage estimation. The advantage is estimated using a single trajectory; thus, the computed advantage has a high variance due to the systematic noise in the advantage esti-

mation. Given two similar states (observation), the advantage estimation should be the same. For example, an observation with the same semantic but a different background (red and blue) should have the same advantage if the background is not essential and thus confounded.

III. BOOSTRAP ADVANTAGE ESTIMATION (BAE)

We proposed to bootstrap the advantage estimation using observation transformation to mitigate the abovementioned issue. Formally, we generate m additional estimation with m transformation. For each such transformation i , we compute an estimate as in equation 3.

$$A_t^{(k,i)} = -V(f(s_{t+1}, v_i)) + r_t + \gamma r_{t+1} + \dots + \gamma^{T-t} V(f(s_{t+k}, v_i)), \quad (3)$$

where v_0 refer to no augmentation. Furthermore, finally, we take the average of all estimates as in equation 4 to estimate the final advantage estimation for k -step return.

$$A_t^{(k,b)} = \frac{1}{m+1} (A_t^{(k,0)} + A_t^{(k,1)} + A_t^{(k,2)} + \dots + A_t^{(k,m)}) \quad (4)$$

Finally, we can achieve bootstrap advantage estimation of a trajectory of length T by combining several k -step returns using exponential-weighted average as in 5 following the GAE method [5].

$$A_t^{BAE(\gamma,\lambda)} = (1-\lambda)(A_t^{(1,b)} + \lambda A_t^{(2,b)} + \dots + \lambda^{T-1} A_t^{(T,b)}) \quad (5)$$

This $A_t^{BAE(\gamma,\lambda)}$ is used to compute the advantage at state s_t of timestep t in an episode. Note that, our BAE differs from the GAE [5] in computing the k -step return as in equation 4. Algorithm 1 shows the details step of using our BAE with the PPO-based policy optimization method. In this paper, we

Algorithm 1 BAE for Policy Optimization

- 1: Get transformation function $f(s, v)$ with augmentation type v
 - 2: Get PPO for policy optimization RL agent
 - 3: **for** each iteration **do**
 - 4: **for** each environment step **do**
 - 5: $a_t \sim \pi_\theta(a_t | s_t)$
 - 6: $s_{t+1} \sim P(s_{t+1} | s_t, a_t)$
 - 7: $r_t \sim R(s_t, a_t)$
 - 8: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
 - 9: **end for**
 - 10: Transform all $s \in \mathcal{B}$ to get \mathcal{B}' using v augmentation with function $f(s, v)$.
 - 11: Compute Bootstrap Advantage Estimate (BAE) from data \mathcal{B} and \mathcal{B}' using equation 5.
 - 12: Perform PPO updates with BAE to optimize for L_π as in equation 1
 - 13: **end for**
-

leverage PPO [3] as the base RL algorithm, which uses generalized advantage estimation (GAE) as the default estimator. In contrast, our method BAE-PPO uses bootstrap advantage

estimation (BAE) instead of GAE. The data augmentation baselines RAD and DRAC use base PPO with GAE advantage estimation. In the experiments we use $m = 1$ in equation 4. This means we use one data augmentation approach and combine it with original advantage estimation as in equation 5.

Note that we do not apply any observation transformation in other parts of the agent objective, and thus equation 2 remains theoretically and practically sound. Furthermore, we empirically show how our Bootstrap Advantage Estimation leads to a smaller value, policy loss, and performance boost. Finally, we also compared the baseline RAD [10] and DRAC [11] and show that our method performs better in many setups.

IV. EXPERIMENTS

A. Setup

Environments We experimented with image-based observations with discrete action space and vector-based observations with continuous action space.

Procgen We use Procgen [17]: Starpilot and Miner (Figure 1) which use image-based observation and procedural generation to produce challenging game logic that changes episode by episode. This benchmark allows for evaluating both sample efficiency and generalization capacity of RL agents. Each environment has around 100K levels. A subset of levels can be used to train the agent, and then the full distribution, that is, 100K levels, can be used to test the agent’s generalization capacity. For our experiment, we use the standard evaluation protocol from [17]; 200 levels of each environment are used for training in the difficulty level *easy*. All the environments have discrete action space of dimension 16. Intuitively, during training, the agent has access to a limited number of environment variability (e.g., 200 levels). The trained agent is tested on all the available variabilities, which consist of unseen scenarios. Thus, to master the game, the agent must focus on essential aspects of the state and ignore irrelevant information such as background color.

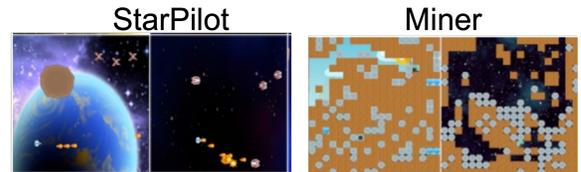


Fig. 1. **Procgen**: Some snapshots of Starpilot and Miner. The environments are generated procedurally, which results in different observations (e.g., background) in each episode.

Deepmind Control We use two environments from *dm_control* [12]: Quadruped run, and Cartpole with three poles (Figure 2). The Quadruped Run has high-dimensional vector observation, and the task is to run as far as possible. On the other hand, the Cartpole variation consists of three procedurally generated poles. The complexity of these environments is suitable for evaluating the data augmentation-based approaches.

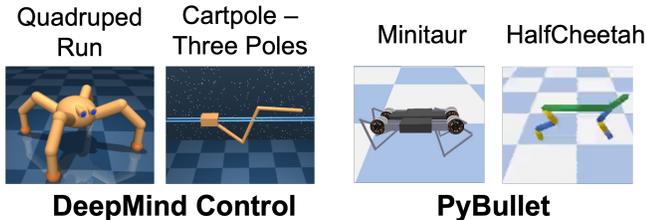


Fig. 2. [Left] *Deepmind Control*: Some snapshots of Deepmind Control tasks. [Right] *Pybullet*: Some snapshots of Pybullet Minitaur quadruped and HalfCheetah environments. These environments contain vector-based state space, and the action space is continuous.

Pybullet We use Pybullet [13]: Minitaur quadruped and HalfCheetah environments with vector-based observation. Each observation consists of raw sensory inputs. The Minitaur quadruped is a 4-legged robot, and the task is to travel as long as possible on flat ground. Furthermore, the HalfCheetah is a two-legged robot that can control its movement in 2D, and the task is to travel as much distance as possible. The action spaces are continuous in these environments. Snippets of these environments are in Figure 2.

Baselines All agents usage on-policy PPO [3] as the base policy. We compare our method with Generalized Advantage Estimation (GAE) [5] which is referred to as **GAE-PPO** in our experiments. GAE is shown to perform better compared to other advantage estimation techniques [6]. Moreover, we compare with the data augmentation-based approach uses data augmentation to transform the observation and then uses the transformed observation to train the base policy. In particular, we compare our method with existing baselines RAD [10] and DRAC [11]. RAD, referred to as **RAD-PPO**, proposes various data augmentation techniques to improve learning from pixel-based observation. DRAC, referred to as **DRAC-PPO** leverages the data augmentation to regularize the policy and value learning, showing improved performance in policy learning. In our method, we replaced the GAE estimation with our proposed Bootstrap Advantage Estimation (BAE), which is referred to as **BAE-PPO**.

We build all the agents, including our BAE-PPO and baseline, using the implementation available in [15]. In a PPO-based scenario, many factors have been identified as key in implementing algorithms that impact the performance [6], [14]. Thus, we use the same implementation logic for all the baselines and our method for a fair comparison.

Data augmentation We evaluate *Cutout Color* data augmentation for image-based observation, which performs best in our setup compared to another popularly used Random Crop. Thus, we report Cutout Color data augmentation results for RAD, DRAC, and our BAE. We use the implementation available in RAD [10] for data augmentation. For the vector-based observation robotic task, we use a random amplitude scale proposed in RAD [10]. This method multiply the observation with a number generated randomly between a range α to β . We used best performing range $\alpha = 0.8$ to $\beta = 1.4$ for our

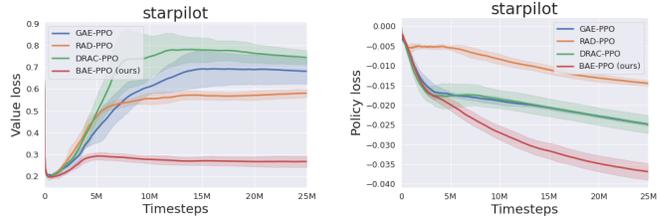


Fig. 3. *Starpilot Env*. Training time **policy** and **value** loss [lower is better]. Our method achieves lowest value and policy losses than the base algorithm (GAE-PPO) and data augmentation baselines (RAD and DRAC).

BAE method, and a range $\alpha = 0.6$ to $\beta = 1.2$ for RAD, and DRAC (suggested in RAD [10]).

Implementation and Hyper-parameters For the Procgen Starpilot and Miner, we report mean and standard deviation across 3 seeds run following the setup of Procgen paper [17]. We used an Nvidia A100 GPU to run agents with the IMPALA CNN model [18] on the image observation-based Procgen environments. We use neural networks to represent policy and value functions for vector-based observations. For Deepmind control environments, we report results with 10 random seed runs, and for Pybullet environments, we report results over 5 seeds. For all experiments, we keep the common hyperparameters the same for a fair comparison. The implementation and hyperparameters are based on [15], [19]. For all results, we report the mean (showed in solid line) and standard deviation (showed in shaded areas) across runs.

B. Results

The PPO-based agent’s objective consists of value loss and policy loss. The objective is to reduce them and potentially improve the expected return. We show that our method BAE reduces the losses and thus learns a better value function and policy than the baselines. We then show how our method performs in the expected return.

Procgen Results Figure 3 shows value and policy loss during policy training on Procgen *Starpilot* environments. As the training progresses, the loss of our method BAE reduces drastically compared to the baselines. These results show the sign of the effectiveness of our method in reducing the agent’s losses. Note that the advantage estimation is used to train both the value function and the policy; thus, better estimation of advantages should generally give better value and policy. In this sense, our method shows empirical evidence that it can help better advantage estimation.

We observe that in the Starpilot environment, the success in policy and value loss translates to the final return. In Figure 4 we see that our method (BAE-PPO) shows improved sample efficiency (Train Return) and generalization capacity compared to the baseline GAE-PPO, RAD-PPO, and DRAC-PPO.

Note that the RAD-PPO performance worsens the performance of the base GAE-PPO algorithms. This result is consistent with the findings in [11], and it shows that naive data

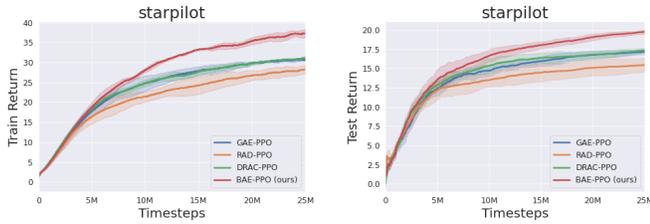


Fig. 4. *Starpilot Env.* **Sample efficiency** performance measured in train time return. We see our method BAE-PPO achieves higher returns where DRAC does not improve the base agent’s (GAE-PPO) performance. RAD slightly worsens the performance of the base agent. **[Right] Generalization** performance measured in test time return. We see a similar trend and observe that our method performs the best.

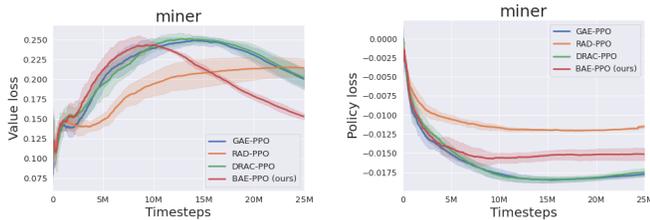


Fig. 5. *Miner Env.* Training time **policy** and **value** loss [lower is better]. The value loss of BAE eventually become lower. Losses of RAD increases compared to base PPO.

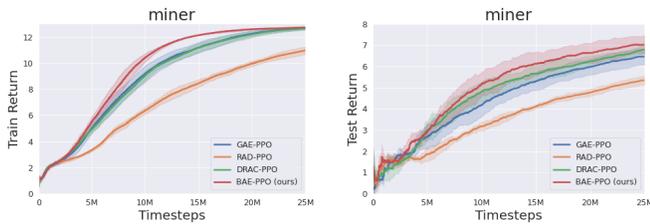


Fig. 6. *Miner Env.* **[Left] Sample efficiency** performance measured in train time return and **[Right] Generalization** performance measured in test time return. Overall, we see our method BAE-PPO shows consistence improvement over baselines.

augmentation can be detrimental to performance. Furthermore, we observe a similar trend for the value and policy loss results.

We observe a similar performance trend in the Procgen **Miner** environment. In Figure 5, we see that in our method, BAE shows a smaller value loss eventually despite being higher at the beginning compared to the baselines. GAE and DRAC show slightly lower values in the policy loss plot than BAE. However, both show smaller policy losses in general.

In performance measure (Figure 6), we see that our method shows better performance throughout the training than the baseline in both sample efficiency and generalization. The performance difference is consistent across timestep. Similar to Starpilot, RAD also performs worse compared to base GAE-PPO.

We further evaluate our method on vector-based state space

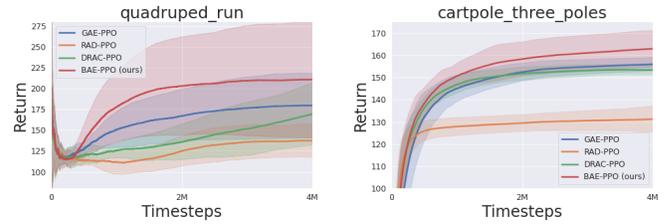


Fig. 7. **[Left]** Performance on Quadruped run environments. Our method BAE-PPO shows higher mean returns compared to all other agents. Data augmentation baseline RAD and DRAC worsen the performance of the base agent (GAE-PPO). **[Right]** Our method BAE consistently achieves a higher mean where DRAC fails to improve upon the base agent, and RAD worsens the base performance.

and continuous robotic tasks: Deepmind control and Pybullet. Note that the setup of these benchmarks are different from Procgen’s train-test setup, and here we evaluate how our agents can perform in high-dimensional states and procedurally generated task. Thus we can only report returns during training.

Deepmind Control Results Figure 7 shows performance comparison on Quadruped run and Cartpole - Three Poles environments. We observe that our method BAE achieves the best performance in both environments. On the other hand, the baseline RAD severely worsens the base agent’s (GAE-PPO) performance in both environments. Another baseline, DRAC, worsens the base agents’ performance in Quadruped Run and fails to improve performance in the Cartpole Three Poles environment. These results show that properly using the augmentation in policy learning can lead to strong performance. In this case, all data augmentation agents (RAD, DRAC, and BAE) use the same *random amplitude modulation* augmentation. However, using this augmentation in advantage computation, our method BAE shows a substantial performance boost. On the other hand, other baselines, RAD and DRAC, worsen the performance (Quadruped Run).

Pybullet Results In Figure 8, for the HalfCheetah environment, we see that our method BAE performs better than base agent GAE-PPO and other data augmentation baselines RAD and DRAC. On the other hand, in Minitaur, the data augmentation baseline DRAC and RAD worsen the base agent’s (GAE-PPO) performance where BAE can maintain the base performance.

Overall, these results show the robustness of our method in performance compared to baseline data augmentation methods. Therefore, the proposed augmented observation is expected not to worsen the base performance. However, in our experiments, we observe that the RAD and DRAC barely match the base agent’s (GAE-PPO) results and sometimes worsen the performance. These variabilities in performance hinder the widespread adaptation of these methods. In contrast, our method BAE shows a consistent performance across various tasks without reducing the base agent’s performance.

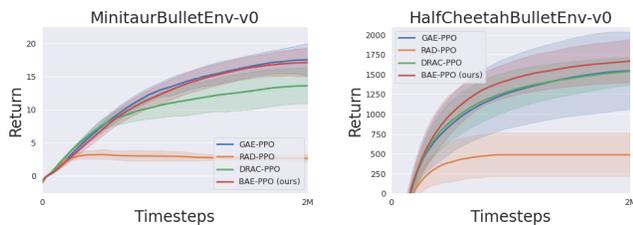


Fig. 8. Performance on Minitaur [Left] and HalfCheetah [Right] environments. Our method shows better or similar performance compared to the base agent (GAE-PPO), where the data augmentation baselines sometimes worsen the base performance.

V. RELATED WORK

Advantage Estimation. The baseline has been leveraged to reduce variance in policy gradient update [2], [20], [21]. Furthermore, effective modification of such methods is the use of advantage estimation. This method is commonly used in policy optimization and enjoys strong empirical success [6], especially the Generalized Advantage Estimation (GAE) [5] method. In contrast to GAE, our method BAE leverages data augmentation and incorporates advantage computation across various semantically similar states. Empirically we observe that our method of computing advantage can be beneficial over GAE, especially in high-dimensional and procedural generated environments.

Data augmentation. Data augmentation has been demonstrated to be an effective and efficient approaches to improve performance [9]–[11]. Other methods proposed to improve generalization which includes regularization [22], and style-transfer [23]. Depending on how the augmented observation is used, the method can be different; for example, RAD [10] and DRAC [11]. In contrast to these methods, our method incorporates data augmentation into advantage estimation, which shows better empirical performance compared to these methods (RAD and DRAC).

VI. CONCLUSION

In this paper, we propose a data augmentation-based advantage estimation method for policy optimization. Our Bootstrap advantage estimation (BAE) method replaces the GAE method in policy gradient-based algorithms. We demonstrated the effectiveness of our method on PPO algorithms. Furthermore, we evaluated our methods on both image-based observation space with discrete action space and vector-based observation with continuous action space (Procgen, Deepmind Control, and Pybullet). Our BAE method showed better performance in various environment setups than GAE. Furthermore, our method performs better than two existing data augmentation techniques (RAD and DRAC).

REFERENCES

[1] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in neural information processing systems*, vol. 12, 1999.

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

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[4] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[5] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[6] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, “What matters for on-policy deep actor-critic methods? a large-scale study,” in *International Conference on Learning Representations*, 2021.

[7] X. Song, Y. Jiang, S. Tu, Y. Du, and B. Neyshabur, “Observational overfitting in reinforcement learning,” in *International Conference on Learning Representations*, 2020.

[8] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, “A study on overfitting in deep reinforcement learning,” *arXiv preprint arXiv:1804.06893*, 2018.

[9] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, “Quantifying generalization in reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 1282–1289.

[10] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, “Reinforcement learning with augmented data,” in *Advances in neural information processing systems*, 2020.

[11] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, “Automatic data augmentation for generalization in deep reinforcement learning,” *arXiv preprint arXiv:2006.12862*, 2020.

[12] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, “dm_control: Software and tasks for continuous control,” *Software Impacts*, vol. 6, p. 100022, 2020.

[13] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2021.

[14] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, “Implementation matters in deep rl: A case study on ppo and trpo,” in *International Conference on Learning Representations*, 2020.

[15] S. Huang, R. F. J. Dossa, A. Raffin, A. Kanervisto, and W. Wang, “The 37 implementation details of proximal policy optimization,” in *ICLR Blog Track*, 2022. [Online]. Available: <https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/>

[16] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, “A survey of generalisation in deep reinforcement learning,” *arXiv preprint arXiv:2111.09794*, 2021.

[17] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, “Leveraging procedural generation to benchmark reinforcement learning,” in *International conference on machine learning*. PMLR, 2020, pp. 2048–2056.

[18] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, “Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures,” *arXiv preprint arXiv:1802.01561*, 2018.

[19] S. Huang, R. F. J. Dossa, C. Ye, and J. Braga, “Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms,” 2021.

[20] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[21] C. Wu, A. Rajeswaran, Y. Duan, V. Kumar, A. M. Bayen, S. Kakade, I. Mordatch, and P. Abbeel, “Variance reduction for policy gradient with action-dependent factorized baselines,” in *International Conference on Learning Representations*, 2018.

[22] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, and K. Hofmann, “Generalization in reinforcement learning with selective noise injection and information bottleneck,” in *Advances in neural information processing systems*, 2019, pp. 13 978–13 990.

[23] M. M. Rahman and Y. Xue, “Bootstrap state representation using style transfer for better generalization in deep reinforcement learning,” in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2022)*, 2022.