Robust Adversarial Reinforcement Learning

Khalil Virji **Julie Alhosh** **Ben Lo**

Abstract

This paper aims to reproduce the results cited in (1), which introduces the concept of Robust Adversarial Reinforcement Learning. Specifically, the paper notes that differences between training and testing scenarios can be modeled as external forces or disturbances, and seeks to improve the performance of a given policy by training an agent to operate within environments controlled by a destabilizing adversarial agent. To reproduce these findings, we evaluate the reported method using both an on-policy and off-policy gradient optimizer across three complex environments. Our study agrees with the conclusions found- notably, that adversarial reinforcement learning leads to better performing policies, is robust to changes between training and testing scenarios, and consistently outperforms baseline reinforcement learning in different adversarial contexts. The work behind this paper was split equally between all authors.

1 Introduction

A significant concern when training agents to operate in real-world environments is the sparsity of available training data. This results in difficulty generalizing to novel situations during test time, and acts as a bottleneck for training real-world physical agents, particularly in robotics. In recreating the results found in (1), we increase the reliability of the cited method to resolve this issue, as well as motivate further study into the benefits and applications of adversarial reinforcement learning. We show that not only are our findings consistent with the original paper, but that the findings also generalize to other policy-gradient optimizers as well, adding further support to the claims made in (1)

2 Background

Robust Adversarial Reinforcement Learning (RARL). In RARL, two agents are trained in the same environment using the same type of policy gradient optimizer. One of the agents is dubbed the "protagonist", and acts in order to maximize environment returns, while the other is dubbed the "adversary" and seeks to minimize returns in the same environment. Both the protagonist and adversary train in the same environment at the same time, where state transitions consist of sampling actions from both agents before computing the corresponding next state. The action space of the adversary consists of applying destabilizing forces to the protagonist, while the protagonist attempts to learn the environment as per normal. This results in a protagonist that has been exposed to destabilizing forces during training, which translates to more robust performances during test time. For a full explanation of RARL, see (1)

In order to replicate the results found within the original paper, the Gymnasium Mujoco environment suite (2) was used to benchmark this approach, specifically the Half-cheetah, Hopper, and Inverted Pendulum environments. Likewise, in order to test the efficacy of the RARL method, two alternate policy gradient optimizers were used, TD3 (3) and PPO (4), which were not implemented in the original paper.

Twin Delayed Deep-Deterministic Policy Gradient (TD3). Expanding on traditional actor critic methods, TD3 utilizes two separate critic networks when estimating the Q-values of environment states, and acts on the smaller of the two values in order to reduce value over-estimation. In addition, the algorithm updates the actor network less often than the critic network, allowing for the generation of more stable Q-values. Finally, TD3 implements noise regularization via target policy smoothing, which adds random noise to the actions of the actor (clipped by a small constant) in order to reduce variance between target values. This results in an off-policy actor-critic network that produces more stable and reliable target estimates. For a full explanation of TD3, see (3)

Proximal Policy Optimization (PPO). PPO, which is also considered an actor-critic method, involves generating a policy from a batch of transitions and comparing it with the actor's current policy. PPO computes the advantage of this new policy

over the current policy, as well as the importance sampling ratio (ratio of action-probabilities for both policies). These values are combined and then clipped, becoming the loss function the agent seeks to minimize. This results in an on-policy algorithm which prevents the actor from making policy updates which deviate too much from the original policy, while still allowing the actor to improve the policy evaluated by the critic. For a full explanation of PPO, see (4)

3 Methodology

To ensure consistency with the original paper, we use the same framework as in (1) which contains the Half-cheetah, Hopper, and Inverted Pendulum environments modified to allow for the application of adversarial forces[1]. For the Inverted Pendulum environment, the adversary can apply a 2D force to the center of the pendulum. For the Half-cheetah environment, the adversary can apply 2D forces to the torso, the front foot, and the back foot. For the Hopper environment, the adversary can apply a 2D force to the foot (see Figure 5).

To implement TD3, we follow a tutorial[2] as well the original implementation[3]. We made changes where needed to allow for training under the presence of an adversary. To implement PPO, a similar tutorial was followed [4], using the original implementation [5]. We again made modifications to allow for adversarial environments and training.

The RARL schema we implement from scratch is shown in Algorithm 1. After initializing the protagonist and adversarial policies, we train the protagonist agent for N_{pro} iterations while keeping the adversarial agent fixed. Next, we train the adversarial agent for N_{adv} iterations while keeping the protagonist fixed. The adversarial agent receives the negative reward of the protagonist, thus the better the returns of the protagonist, the worse the returns of the adversary, and vice versa. Each agent is hence rewarded for undermining the actions of the other. This process repeats for a total of N_{train} iterations.

The learning curves averaged over 5 independent experiments using the above RARL schema are presented in Figures 1 and 2. Note the baseline agents we compare to are agents trained using the same learning algorithm with the same hyperparameters, but without the presence of an adversary. We see the learning curves are similar for both the RARL schema agents and the baseline agents in most environments. However the RARL schema agents often have a higher performance ceiling and in the Half-cheetah environment using TD3, learn a significantly better policy. The full implementation and code used to run these experiments, including the trained models can be found here.

Algorithm 1 RARL Schema

Initialize θ_{pro} and θ_{adv}
for $i = 1, 2, \ldots, N_{train}$ **do**
    for $j = 1, 2, \ldots, N_{pro}$ **do**
        Run policies θ_{pro} and θ_{adv} and store data in replay buffer
        $\theta_{pro} \leftarrow$ train(replay buffer, θ_{pro})
    end for
    for $j = 1, 2, \ldots, N_{adv}$ **do**
        Run policies θ_{pro} and θ_{adv} and store data in replay buffer
        $\theta_{adv} \leftarrow$ train(replay buffer, θ_{adv})
    end for
    Evaluate θ_{pro} for 10 episodes without adversarial disturbances
end for
Return θ_{pro}, θ_{adv}

4 Experimental Results

We use three evaluations to compare agents trained using the RARL schema with their corresponding baseline. The first evaluates the learned policies without the presence of adversarial disturbances. The second evaluates the learned policies with changing mass values at test time. The third evaluates the learned policies under manually defined adversarial disturbances. The results consistently show that agents trained using the RARL schema yield stronger performance and are more robust to differences between training and testing environments.

[1]Available at: https://github.com/lerrel/gym-adv

[2]Available at: https://towardsdatascience.com/td3-learning-to-run-with-ai-40dfc512f93

[3]Available at https://github.com/sfujim/TD3

[4]Available at: https://medium.com/analytics-vidhya/coding-ppo-from-scratch-with-pytorch-part-1-4-613dfc1b14c8

[5]Available at: https://github.com/ericyangyu/PPO-for-Beginners

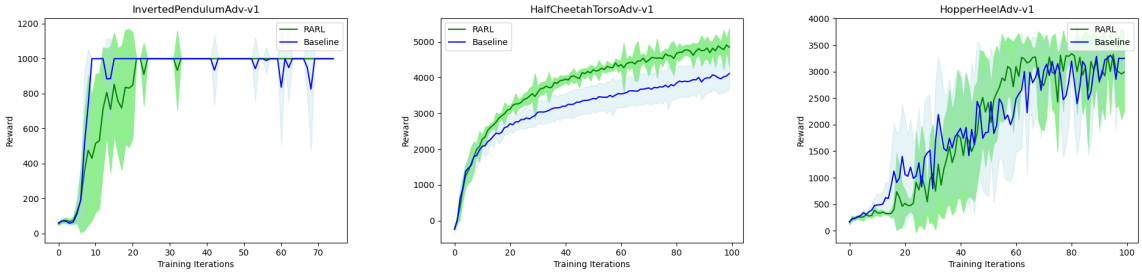

Figure 1: Learning curves for TD3-RARL policies versus the baseline (TD3) when evaluated without adversarial disturbances after each training iteration.

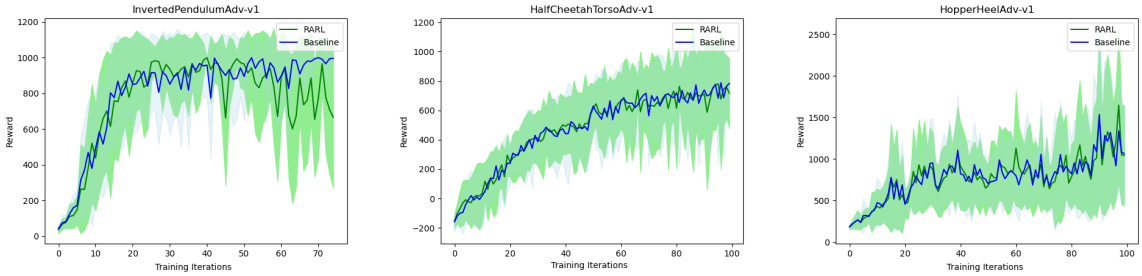

Figure 2: Learning curves for PPO-RARL policies versus the baseline (PPO) when evaluated without adversarial disturbances after each training iteration.

Evaluating without adversarial disturbances. For each environment and learning algorithm, we evaluate the agent trained using the RARL schema and its corresponding baseline for 100 episodes without adversarial disturbances. We report the mean and standard deviation of the obtained rewards in Table 1. For the TD3 algorithm, the policies learned by the RARL schema yield higher mean rewards with lower variance than the baseline policies. For PPO, a similar trend is seen, with RARL policies reporting higher mean rewards, however with significantly higher variance. This is believed to be the result of insufficient hyperparameter tuning. Nevertheless, these results indicate that training under the presence of an adversary helps the agent learn a stronger policy.

Evaluating with changing mass values. Modeling errors between simulation and real-world environments can have a large and detrimental impact on the performance of a policy. Thus it is vital that learned policies are robust to such errors. For the second evaluation, we manually inject modeling errors at test time to asses how robust the policies learned by the RARL schema are. We do this by changing the pendulum mass for the Inverted Pendulum environment and the torso mass for the Half-cheetah and Hopper environments. We evaluate for 100 episodes and report the mean and standard deviation of the obtained rewards in Figures 3 and 4. For the TD3 algorithm, the policies learned by the RARL schema are much more robust to changing mass values at test time, especially for the inverted pendulum environment. The PPO algorithm likewise exhibits similar robustness, significantly outperforming the baseline in both the Inverted Pendulum and Hopper environments. We note, however, a comparable performance with the baseline in the Half-cheetah environment, as well as significantly higher standard-deviations when compared to TD3. Considering the strong performance of the RARL schema policy in the PPO Inverted Pendulum and Hopper environments, these errata are likely to be the result of an insufficient hyperparameter tuning, rather than a reflection of the robustness of the RARL schema or PPO. Even with these irregularities considered, these results still show that the RARL schema helps agents learn policies that are much more robust to modeling errors between testing and training environments.

Evaluating with adversarial disturbances. For the third evaluation, we explore how agents trained using the RARL schema perform under the presence of adversarial disturbances at test time. To do this, we define adversarial forces regulated by a given maximum strength to try to decrease the performance of the agent. For the Inverted Pendulum environment, we apply a random 2D force in any direction to the pendulum at each time step. For the Half-cheetah environment, we

	InvertedPendulum	**HalfCheetah**	**Hopper**
TD3-Baseline	1000.00 $\pm$0.00	4563.53 $\pm$233.75	3262.41 $\pm$9.46
TD3-RARL	1000.00 $\pm$0.00	5121.98 $\pm$153.83	3436.27 $\pm$4.59
PPO-Baseline	977.08 $\pm$100.48	668.07 $\pm$110.71	704.73 $\pm$260.15
PPO-RARL	1000.00 $\pm$0.00	685.59 $\pm$437.20	1878.63 $\pm$969.02

Table 1: Evaluation of learned policies averaged over 100 episodes without adversarial disturbances

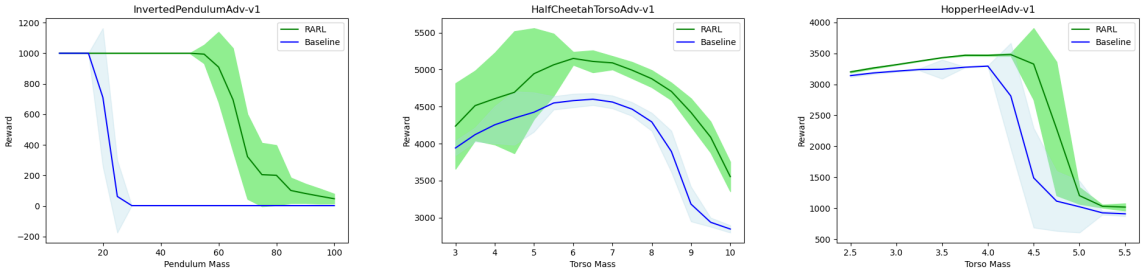

Figure 3: Evaluation of TD3-RARL policies versus the baseline (TD3) with changing mass values.

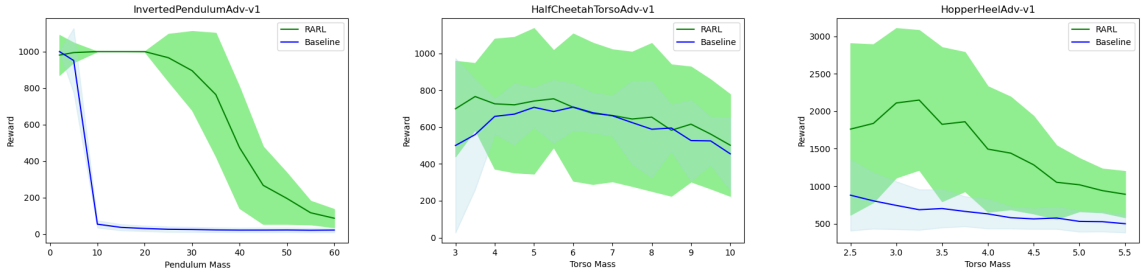

Figure 4: Evaluation of PPO-RARL policies versus the baseline (PPO) with changing mass values.

apply a random 2D force to the feet and torso at each time step but only in directions with a backward (left) component. Similarly, for the Hopper environment, we apply a random 2D force to the feet but again only in directions with a backward (left) component. We do not apply forward (right) forces for these two environments as this would reward the agent for an unintended rightward motion. The forces described above are illustrated in Figure 5 for reference.

We evaluate the policies learned using the RARL schema and their corresponding baselines for 100 episodes under the presence of adversarial disturbances of varying maximum strengths. We record and report the mean and standard deviation of the obtained rewards in Figures 6 and 7. As we expect, as the maximum strength of the adversarial force increases, the performance of the agents ultimately decreases. However, for the TD3 learning algorithm, the policies learned using the RARL schema still yield higher mean rewards in all cases. While the differences between PPO policies are less pronounced, we still observe either equal or higher mean rewards across all environments, with the most noticeable improvements found in the Inverted Pendulum and Hopper environments. These results further indicate the strength of policies trained using the RARL schema in challenging test environments.

5 Conclusion and Future Work

The purpose of this paper was to recreate the findings found in (1), which found that Adversarial Reinforcement learning 1) improves the performance of the policy learnt by the agent during training, 2) is robust to differences training and testing scenarios, and 3) consistently outperforms the baseline. As a result, we have created 3 experiments to test these claims, including evaluating policies trained using the RARL schema with and without adversarial disturbances, as well as with different test conditions (changing mass values). We have found that both TD3 and PPO algorithms, when trained using the RARL schema, are more robust to changes in environment conditions and either outperform or perform on par with their baseline counterparts. The TD3 algorithm in particular significantly outperforms the baseline across the board and learns a significant better policy in the Half-cheetah environment (Figure 1). In cases where the RARL policy performs on par with the baseline, we attribute the worse-than-expected performance to inadequate hyper-parameter tuning, which is specific to each environment.

As such, future studies should first include an in-depth hyper-parameter tuning of both algorithms across all three environments in order to ensure stable performance. Once this is accomplished, another potential area for study would be to modify the RARL schema in order to optimize performance. In particular, modifying the sequence in which the protagonist and adversaries are trained (in series, in parallel, protagonist-first, adversary-first, multiple iterations, etc.) This would allow us to further understand the relationship between the protagonist and adversary during training, and how to leverage this relationship for better performance.

References

[1] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*, pp. 2817–2826, PMLR, 2017.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[3] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

6 Appendix

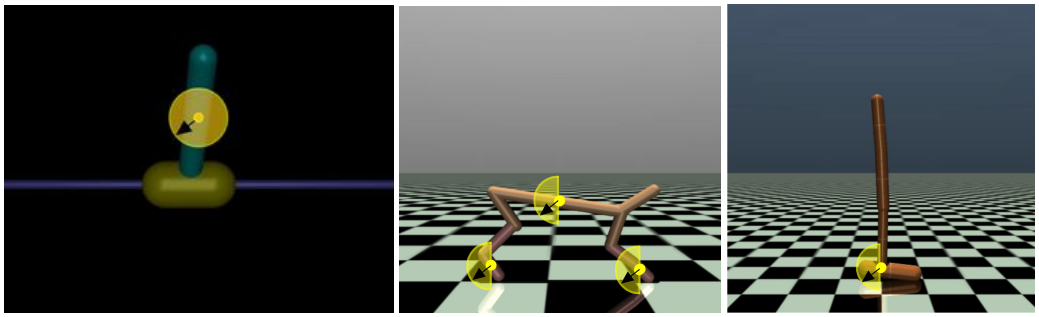

Figure 5: Manually defined adversarial disturbances applied to environments at test time.

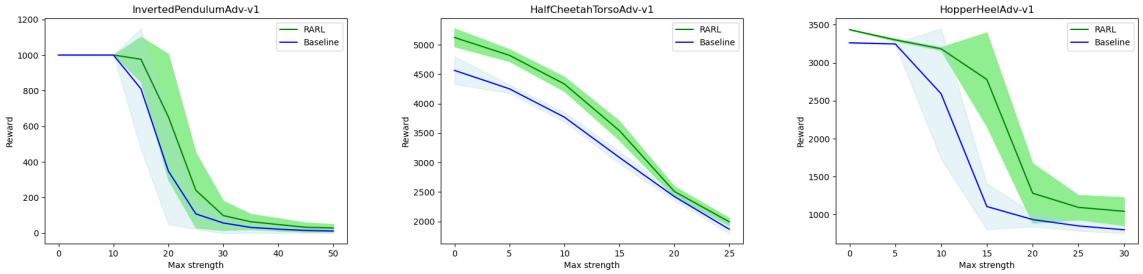

Figure 6: Evaluation of TD3-RARL policies versus the baseline (TD3) with manually defined adversarial disturbances.

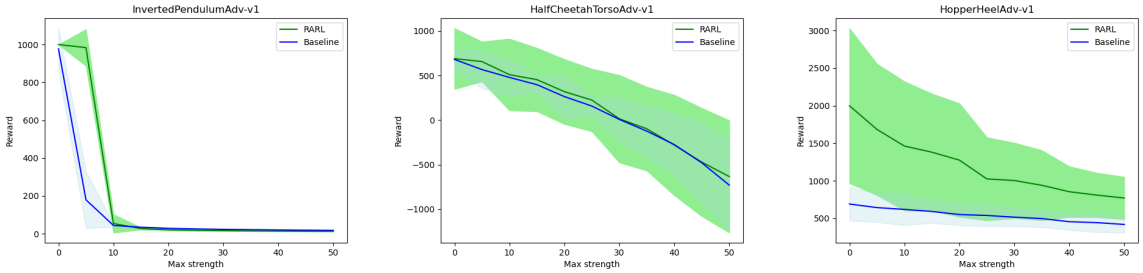

Figure 7: Evaluation of PPO-RARL policies versus the baseline (PPO) with manually defined adversarial disturbances.