Robust Adversarial Reinforcement Learning Analysis

Zisu Dong[1], Yujia Luo[2], Xingyou Song[3], and Xin Wang[4]

Abstract— **The goal of this project is to re-implement and further analyze the RARL algorithm found in ”Robust Adversarial Reinforcement Learning” [PDSG17], and to discuss this approach against alternative approaches. The baseline reimplementation was performed on OpenAI Gym environments, and further implementation was made for real-world data sets, including the pedestrian detection dataset collected by Berkeley DeepDrive. The policy algorithm was also changed from the original work for further testing.**

I. Problem Definition and Motivation

A. Background

Algorithmic game theory has been a recently popular approach to reinforcement learning algorithms. With the popular successes of AlphaGo Zero [SHM$^+$16] and Open AI's DOTA2-bot [AI17] based on self-playing reinforcement learning, there is much inspiration from the game theory literature.

For this project, the specific goal is to model adversarial disturbances in a system by an agent. In training policy gradient methods, there is a reliance of large amounts of data in real world policies. A large amount of data is needed for the policy to train on many different scenarios, which is usually unavailable. Furthermore, the inherent noisiness of real world data can also affect both training and testing. For instance, training on simulated models in OpenAI Gym is much simpler than the many, possibly infinite variables in real world applications, which explains the discrepancy between simulated and actual performance in robotic tasks.

In ”Robust Adversarial Reinforcement Learning” [PDSG17], the authors solve these problems by framing the problem in terms of a zero sum game, in which a learner trains his policy against an adversary (which represents noise from data or malicious forces in Gym).

The goal of the game is for both players reach a max-min solution (after defining a reward function), in which the learner's policy is robust against adversarial forces, but also more-well trained even under data scarcity.

B. Problem Statement

In RARL, we are given an Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{P}, r, \gamma, s_0)$ where $\mathcal{S}$ is the set of continuous states, $\mathcal{A}_1, \mathcal{A}_2$ are sets of continuous actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A}_1 \times A_2 \times \mathcal{S} \to \mathbb{R}$ is the transition probability, $r : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{R}$ is a reward of both players. Suppose player 1 plays strategy μ and player 2 (adversary) plays strategy ν. Then the reward function for player 1 is $r_{\mu,\nu} = \mathbb{E}_{a^1 \sim \mu(.|s), a^2 \sim \nu(.|s)}[r(s, a^1, a^2)]$ with the reward of player 2 being the negative, $-r_{\mu,\nu}$.

Essentially, in the adversarial game, at every timestep t, both players observe the state s_t and take actions $a_t^1 \sim \mu(s_t)$, and $a_t^2 \sim \nu(s_t)$. Then the state transitions are generated with $s_{t+1} = \mathcal{P}(s_t, a_t^1, a_t^2)$, with the reward as $r_t = r(s, a_t^1, a_t^2)$, with player 1 receiving the positive value of this reward, and player 2 receiving the negative value of this reward. Thus we may define the total reward function as $R^1 = \mathbb{E}_{s_0 \sim \rho, a^1 \sim \mu(s), a^2 \sim \nu(s)}[\sum_{t=0}^{T-1} r^1(s, a^1, a^2)]$. Since μ, ν are the only inputs allowed, then we may define this as a game in which actions are policies: $R^1 = R^1(\mu, \nu)$, from which the objective is to calculate the equilibrium $\min_\nu \max_\mu R^1(\mu, \nu) = \max_\mu \min_\nu R^1(\mu, \nu)$.

II. Related Work

In this section, we review related work and compare their implementations. In the constraint of time, we did not implement the alternative methods, but include them as alternative approaches for improvement and exposition, as many of these methods are currently the state of the art.

We will first focus on the general methods from algorithmic game theory, representing the policies as actions in a game. First, we provide notation: A **normal-form** game is a tuple $(\mathbf{\Pi}, U, n)$ where $\mathbf{\Pi} = (\Pi_1, ..., \Pi_n)$ is a set of actions (or policies for the RL case), with $U : \mathbf{\Pi} \to \mathbb{R}^n$ a utility function of the actions inputted. **Extensive Form** Games are sequential games which require more turns. In zero-sum games, there are generally the following strategies used to compute Nash-Equilibria. These are: Double-Oracle, Fictitious play, Regret Minimization. Fictitious play has been popularized in the RL community with works such as [HLS15]. Since II-.1 is a generalization of fictitious play and [HLS15] is already well known to the RL community, we direct the reader to those. We will review the other theory-based approaches and compare these to the RARL framework.

One note is that most games must be formulated in the zero-sum framework. For general classes of games without any basic structure, the complexity of finding Nash Equilibrium is PPAD-Complete [DGP09] it has been proven [Rub15] that there also does not exist even an approximation algorithm for computing Nash equilibria in polynomial time. This makes models, such as our regularization and cross-entropy loss framework in VI-A, VI-B, not provide

*

[1] j_dong1021@berkeley.edu

[2] yujialuo@berkeley.edu

[3] xsong@berkeley.edu

[4] xinw@berkeley.edu

any theoretical guarantees. It is interesting to see if later provide a format other than zero-sum games.

To the best of the authors' knowledge, almost all approximation algorithms for general games use linear programming or numerical solutions, which is ill-suited for reinforcement learning. This becomes a very strong theoretical limitation to handling games in which players need to e.g. coordinate with each other towards a common goal, which explains the lack of literature on such topics currently.

Another note is that most sufficiently sophisticated equilibria algorithms, either: 1. Require discrete action spaces, or 2. Are suited for continuous spaces. For the first case, this implies that to fit for RL-based algorithms, there is a weighted mixture of "pure" RL strategies, which makes the game stochastic (i.e. only playing one action-policy at a time). Furthermore, these RL strategies must approximate the continuous policy space, which can sometimes be difficult. For algorithms of the second form, to fit into the RL framework, this means using the gradient estimator $\hat{\mathbb{E}}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right]$. Nonetheless, we explore both types of approaches from the literature.

1) Double-Oracle Approaches: The idea of adversarial learning and pursuing zero-sum min-max strategies for reinforcement learning has been well known even early in 2003 [MGB03]. In their work and subsequent works which use the *double oracle* algorithm, inspired by approaches from adding more constraints in linear programming, the basic premise of the algorithm works as follows as below:
In the finite-action space case, at time t, a subgame G^t consists of the payoffs only generated by action space $\mathbf{\Pi}^t \subset \mathbf{\Pi}$. For this subgame, a Nash-Equilibrium σ^t is determined for G^t. On time $t+1$, each player plays a best response (pure strategy) π_i^{t+1} using the actions from the *entire* space $\mathbf{\Pi}_i$ assuming other players' strategies are fixed from equilibrium, i.e. $BR(\sigma_{-i}^t)$ where BR is the best response function. Then the new game G^{t+1} is given by the updated action space $\mathbf{\Pi}_i^{t+1} = \Pi_i^t \cup \{\pi_i^{t+1}\}$. The algorithm terminates if the best responses do not improve the result.

We may then create an approximate Double-Oracle algorithm for continuous action spaces (such as policies in RL) by terminating the algorithm when the best response oracles do not improve the result for at least player by at least ϵ in value.

This approach in the best case scenario does not need to enumerate the entire search space, but for games such as Rock-Paper-Scissors, the algorithm does indeed enumerate all actions, which is a strong criticism of its efficiency. Furthermore, conceptually speaking, for the case of reinforcement learning especially for continuous policy spaces, it is not known how to find a best response policy *purely* by utility values - i.e. a question of how to find a best-response policy against adversary policies $\mu_1, ..., \mu_k)$ under a weighted distribution $(p_1, ..., p_k)$, without merely calculating all possible reward values for every policy.

In subsequent works, this approach has been recently re-modified [LZG$^+$17] into a "Policy-Space Response Oracle", for large scale reinforcement learning when the action space are policies, which solves this problem of best-response against weighted averages of policies. In this algorithm (again in the finite case), there exists an initial policy set $\mathbf{\Pi}$, from which the entire utility function also exists. Each player i has a probability distribution σ_i over his policy space $\mathbf{\Pi}_i$, initialized to be uniform. At time t, for each player i, for many episodes, other players sample their policies from their own distributions σ_{-i}, and ultimately player i continually trains a policy π_i' based on the performance of his policy in the previous episode combined with the currently sampled σ_{-i}, i.e. (π_i', π_{-i}). This is repeated until convergence to a particular policy π_i for player i, which is then added to the new action space $\mathbf{\Pi_i} = \mathbf{\Pi_i} \cup \{\pi_i'\}$. The σ distribution is then updated based on this new join action space $\mathbf{\Pi}_1^{t+1}, ..., \mathbf{\Pi}_n^{t+1})$ after all players update their own action spaces. The continuous case may approximate the action space by discrete sampling. This algorithm for zero-sum games takes polynomial in the action space size $|\mathbf{\Pi}|$.

2) Regret Minimization: The other method to approach equilibria is based on regret-minimization and online learning, a relatively well known problem within the bandit feedback community. However, because there are no current works (under the author's best knowledge) which combine no-regret algorithms with games where the actions are RL-policies, the idea is still based on modularizing the policies as merely actions, with multiple simulations to provide a normal form game's values. We may represent the zero-sum game G under two players with action inputs $\theta_1^\mu, \theta_2^\nu$ where μ, ν are policies and the θ's are the parameters for the policies. If the parameters θ's are from a continuous domain, we may either discretize the domain spaces, or perform gradient methods (if available), as shown below.

The most notable reports on this topic come from zero-sum games on [DDK15] (which is then improved in [RS13] for convex-concave games, faster algorithms from [SALS15] and lastly, stochastic bandit feedback from [FLL$^+$16]. This has been currently explored in other settings as well, such as training of GAN's [DISZ17] and [GLL$^+$17]

Unfortunately, since these games do not have a convex-concave nature, there is no guarantee for when we use Optimistic Mirror Descent (OMD) from [RS13]. Here, we describe the basic setup under non-stochastic settings. For a zero-sum game, the OMD algorithm makes both players update their strategy under observation of the sequence of pay-off vectors $U(\cdot, \Pi_2^t), U(\Pi_1^t, \cdot)$ as t moves forward. Given a strongly convex regularizer $\mathcal{R}$, then the protagonist may play moves in the form of:

$$\sigma_t = \arg\min_{\sigma} \eta_t \langle \sigma, M_t \rangle + \mathcal{D}_{\mathcal{R}}(\sigma, \tau_{t-1})$$

$$\tau_t = \arg\min_{\tau} \eta_t \langle \tau, \nabla U(\sigma_t, \Pi_2^t) \rangle + \mathcal{D}_{\mathcal{R}}(\tau, \tau_{t-1})$$

where $\mathcal{D}_{\mathcal{R}}$ is the Bregman Divergence with respect to $\mathcal{R}$ and $\{\eta_t\}$ is a sequence of step sizes. M_t is a vector that is based on previous observations as well. In our setting, both players (protagonist and antagonist) will both use their own OMD algorithms. The algorithm is closely related to the well known *multiplicative weights*(MWU) algorithm, in the case $\mathcal{R}$ is the entropy regularizer. We note that the **average** of the $\{\sigma_t, \}_t$ for the protagonist will be the equilibrium strategy, not the ending strategy σ_T at some time T. The convergence guarantee given in [RS13] is given by finding a $\mathcal{O}\left(\frac{\log n + \log m}{T}\right)$-approximate equilibrium, where m, n are the dimensions of the protagonist and adversary, respectively. For ordinary games, the $\log n + \log n$ factors is almost always irrelevant compared to the T factor. However, for the case of RL-algorithms, m, n may consist of the number of policies used to approximate the policy space for both the protagonist and adversary, which is dependent on the setting. Furthermore, under a natural stochastic setting for RL-agents, [FLL$^+$16] proved that even stochastic feedbacks allow for efficient convergence, which grants promise to these newer lines of approaches for adversarial RL.

3) Best-Response: In RARL's approach, the method for achieving equilibrium is through best-response play (i.e, both players alternate using best-response oracles against the other player's **current** strategy). Its ideas are based on newer works in stochastic games such as [PSPP15], rather than in more traditional approaches in fixed-value games (when interpreting rewards in expected value).

A criticism is that there is no strong justification in their work [PDSG17] for its algorithm to converge to equilibrium - intuitively, it should in fact, not be able to converge to equilibrium due to cyclic properties in many example games.

III. Approach

From the alternating best-response approach, we present the general form of the RARL algorithm, as shown below:

Note that N_{tier} needs to be defined in advance, which can be problematic when it is unknown the convergence rate. Indeed, a better method in the future may be instead to check for convergence. Furthermore, $policyOptimizer$ can allow for any policy to be used, as seen later in the implementation details when PPO is used, rather than TRPO.

IV. IMPLEMENTATION DETAILS

The original paper tests the RARL framework on five different OpenAI gyms [BZ16] control environments with the MuJoCo [TT12] physics simulator, which include InvertedPendulum, HalfCheetah, Swimmer, Hopper and Walker2D (description of the environments can be found in the original paper).

Algorithm 1: RARL Algorithm

```
Input     : Environment $\mathcal{E}$, Stochastic policies $\mu$, $\nu$
Initialize: Learnable parameters $\theta_0^\mu, \theta_0^\nu$ for $\mu, \nu$.
for $i = 1, 2, ..., N_{iter}$ do
    $\theta_i^\mu \leftarrow \theta_{i-1}^\mu$
    for $j = 1, 2, ..., N_\mu$ do
        $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow (\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_{i-1}^\nu}, N_{traj})$
        $\theta_i^\mu \leftarrow policyOptimizer(\{(s_t^i, a_t^{1i}, r_t^{1i}), \mu, \theta_i^\mu\})$
    end
    $\theta_1^\nu \leftarrow \theta_{i-1}^\nu$
    for $j = 1, 2, ..., N_\nu$ do
        $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow (\mathcal{E}, \mu_{\theta_i^\mu}, \nu_{\theta_i^\nu}, N_{traj})$
        $\theta_i^\nu \leftarrow policyOptimizer(\{(s_t^i, a_t^{2i}, r_t^{2i}), \nu, \theta_i^\nu\}$
    end
end
return $\theta_{N_{iter}}^\mu, \theta_{N_{iter}}^\nu$
```

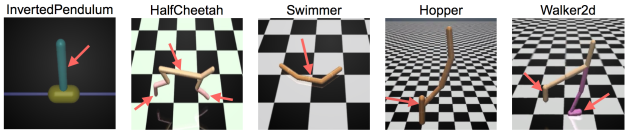

Fig. 1. The paper evaluates RARL on a variety of OpenAI gym problems. The red arrows denote the disturbance force exerted by the adversary

The paper uses TRPO ([SLA$^+$15]) policy optimizer as baseline, and demonstrates that RARL outperforms TPRO in the following three evaluation settings:

1) Test without any disturbance
2) Test with learned disturbance
3) Test with predefined disturbance (varying test conditions)

Our implementation of the framework exactly follows what is described in the paper, but we did not follow the exact same evaluation procedure. We used PPO ([PDSG17]) policy optimizer as the baseline instead of TRPO, because we wanted to check if RARL framework works for any policy optimizer. PPO is also more sample efficient and easier to implement. We implemented the experiment on InvertedPendulum and Hopper environments and tested the framework with no disturbance on the environment.

The authors of the original paper provide MuJoCo environments with added adversarial forces, but there is no implementation for the RARL framework itself. We used Tensorflow to implement RARL, and we will use it again for this problem.

V. Results - Replicated

Figures from 2, 3 show that RARL outperforms PPO baseline in both InvertedPendulum and Hopper environments.

VI. Results - Extensions

Besides the reinforcement learning problems discussed in the original paper, we also hope to show that the RARL

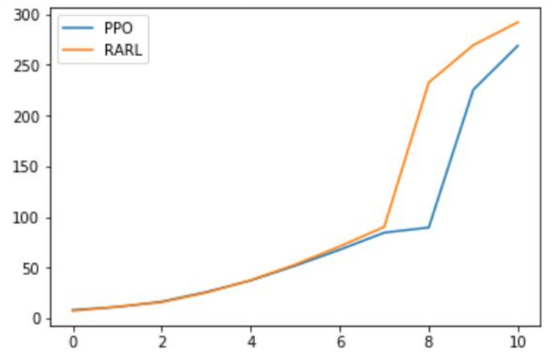

Fig. 2. PPO(baseline) and RARL performance over number of episodes in default InversePendulum environment with no disturbance

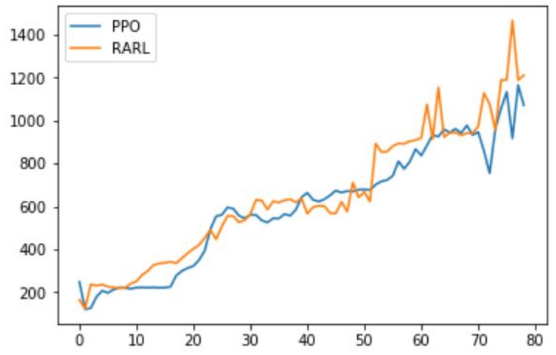

Fig. 3. PPO (baseline) and RARL performance over number of episodes in default Hopper environment with no disturbance

framework can work on general deep learning problems.
We observe that the loss of adversarial agent in the RARL framework is simply defined as negating the loss of the protagonist. However, this definition works under reinforcement learning setting because the expected reward is defined as

$$R = E_{s_0 \sim \rho, a_1 \sim \mu(s), a_2 \sim \nu(s)}(\sum_{t=0}^{T-1} r(s_t, a_1, a_2))$$

which enables optimizer to take gradient with respect to both a_1 and a_2. For deep learning problems, we formulate the disturbances as noise applied to inputs. However, usually the outputs of the protagonist are labels: if these labels were passed into the loss, then they would be treated as constants and the gradient operator would diminish them. Furthermore, it is also not known how much power to give to the adversary in this problem for optimal protagonist training (e.g. the protagonist may overfit if adversary were given a large noise).
To address this problem, we propose a new definition of loss for the adversarial agent from regularization which penalizes large noise given by adversary (this is not a strictly ℓ_2-regularization problem as seen):

$$loss_{adv} = -loss_{pro}(||\hat{\epsilon}||_2) + \lambda ||\hat{\epsilon}||_2$$

where $loss_{pro}(||\hat{\epsilon}||_2)$ is the loss defined in VI-A, which is the loss given when an adversary applies a disturbance $\hat{\epsilon}$. The intuition behind this formulation is that we can:

- treat the prediction error of the protagonist model as a mask
- regulate the noise input so that the adv model would output the smallest noise possible if the same result can be achieved.

We experimented on pedestrian detection problem using a LiDAR dataset which is low dimensional, as well as image classification problem using a benchmark image dataset which is high dimensional.

Note that this is not a zero sum game between the protagonist and adversary anymore, which makes the RARL algorithm unjustifiable from a theoretical point of view, even though this regularization makes sense from a practical point of view. To the best of the authors' knowledge, this method has not been done in any previous works related to game-theoretic reinforcement learning.

A. *RARL on Pedestrian Detection*

Pedestrian detection is a key problem in the field of autonomous driving. We test the RARL framework on a LiDAR-based dataset collected on a real vehicle from Berkeley DeepDrive (BDD). The goal is to model the expected trajectory of a vehicle given that the ground truth trajectory is produced by a human driver, so that the system may learn how to deal with pedestrian-car cases. The dataset contains the trajectories of both the vehicle and the pedestrians. More specifically, as shown in we represent state as $s_t = \langle x_p, y_p, y_v \rangle$, where x_p and y_p define position of the pedestrian and y_v defines position of the vehicle. We don't include x position of the vehicle because the vehicle is only moving in one direction and we align it with y-axis.

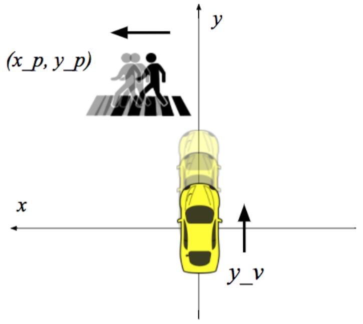

Fig. 4. Predicting the trajectory of the vehicle after seeing a human.

The protagonist aims at imitating human driver's trajectory while the adversary tries to learn a constrained noise on the state environment. We use 2-fully-connected-layer regression model as the policy for both protagonist and adversary. The protagonist minimizes a standard square-loss

$$loss_{pro} = (\hat{y_v} - y_v)^2$$

where $\hat{y_v}$ is the predicted next position of the vehicle and y_v is the ground truth next position; the adversary minimizes protagonist-based loss

$$loss_{adv} = -loss_{pro}||\hat{\epsilon}||_2 + \lambda||\hat{\epsilon}||_2$$

where ϵ is a learned noise exerted on the state, and in order to imitate an action space of reinforcement learning scenario, we clip the noise by a value of 0.2.

The results in Fig. 5 shows that RARL performs better than the baseline in the three evaluation settings proposed by the original paper: a) test without disturbance, b) test with learned disturbance, and c) test with predefined disturbance (in this case we use a random generated noise). In addition, Fig. 6 shows that RARL framework converges faster than the baseline.

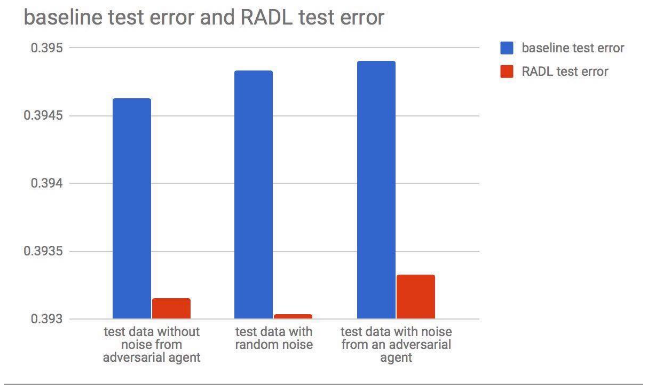

Fig. 5. RARL testing on Berkeley Deepdrive.

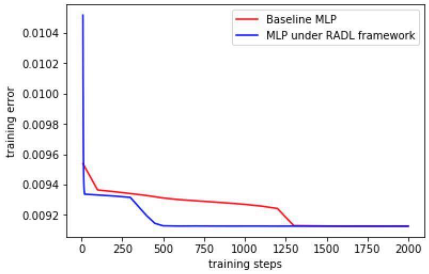

Fig. 6. Training error against the number of training steps used.

B. RARL on Benchmark Image Dataset

We test RARL on image classification problem using CIFAR-10 [AKH]. We use a convolutional neural net with 2 convolution layers and 3 fully connected layers as the model for both protagonist and adversary. Protagonist minimizes a standard softmax cross entropy loss, while adversary minimizes a protagonist-based loss:

$$loss_{pro} = -\sum_i \hat{y}_i log(y_i)$$

$$loss_{adv} = -loss_{pro}||\hat{\epsilon}||_2 + \lambda||\hat{\epsilon}||_2$$

Again, the results in Fig. 7 shows that RARL performs better than the baseline in the two evaluation settings: a) test without disturbance, b) test with learned disturbance. In addition, Fig. 8 and 9 shows that RARL framework converges faster than the baseline.

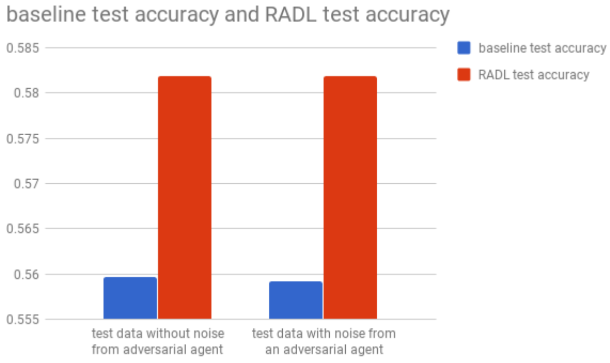

Fig. 7. RARL testing on cifar10.

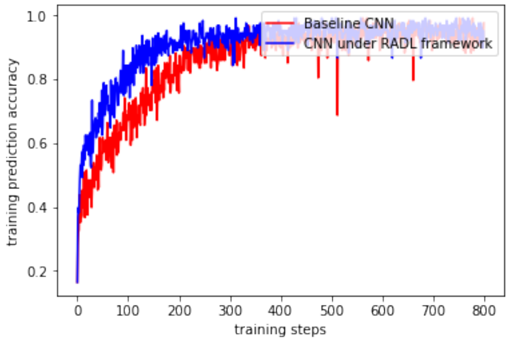

Fig. 8. Accuracy on training set against the number of training steps used.

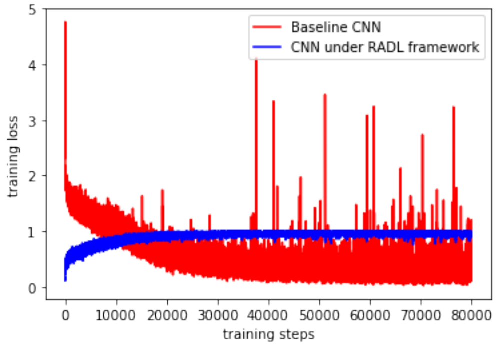

Fig. 9. Training loss against the number of training steps used.

VII. Codebase

The code can be found at `https://github.com/Jekyll1021/RARL`, which consists of the reimplementation and the extension codings. The original author's code may be found in `https://github.com/lerrel/rllab-adv`.

References

[AI17] Open AI. OpenAI Dota 2 Bot, 2017.

[AKH] Vinod Nair Alex Krizhevsky and Geoffrey Hinton. The cifar-10 dataset.

[BZ16] Cheung Vicki Pettersson Ludwig Schneider Jonas Schulman John Tang Jie Brockman, Greg and Wojciech Zaremba. OpenAI Gym. volume abs/1606.015409, 2016.

[DDK15] Constantinos Daskalakis, Alan Deckelbaum, and Anthony Kim. Near-optimal no-regret algorithms for zero-sum games. *Games and Economic Behavior*, 92:327–348, 2015.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *Commun. ACM*, 52(2):89–97, 2009.

[DISZ17] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. *CoRR*, abs/1711.00141, 2017.

[FLL+16] Dylan J. Foster, Zhiyuan Li, Thodoris Lykouris, Karthik Sridharan, and Éva Tardos. Learning in games: Robustness of fast convergence. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4727–4735, 2016.

[GLL+17] Paulina Grnarova, Kfir Y. Levy, Aurélien Lucchi, Thomas Hofmann, and Andreas Krause. An online learning approach to generative adversarial networks. *CoRR*, abs/1706.03269, 2017.

[HLS15] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 805–813, 2015.

[LZG+17] Marc Lanctot, Vinícius Flores Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *CoRR*, abs/1711.00832, 2017.

[MGB03] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 536–543, 2003.

[PDSG17] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2817–2826, 2017.

[PSPP15] Julien Pérolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. Approximate dynamic programming for two-player zero-sum markov games. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1321–1329, 2015.

[RS13] Alexander Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. *CoRR*, abs/1311.1869, 2013.

[Rub15] Aviad Rubinstein. Inapproximability of nash equilibrium. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 409–418, New York, NY, USA, 2015. ACM.

[SALS15] Vasilis Syrgkanis, Alekh Agarwal, Haipeng Luo, and Robert E. Schapire. Fast convergence of regularized learning in games. *CoRR*, abs/1507.00407, 2015.

[SHM+16] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1889–1897, 2015.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms.

CoRR, abs/1707.06347, 2017.

[TT12] Erez Tom Todorov, Emanuel and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference*, page 50265033, 2012.