Enhancing Adversarial Examples on Deep Q Networks with Previous Information

Korn Sooksatra* and Pablo Rivas†, *Senior, IEEE*
School of Engineering and Computer Science
Department of Computer Science, Baylor University
*Email: Korn_Sooksatra1@Baylor.edu, †Email: Pablo_Rivas@Baylor.edu

***Abstract*—Reinforcement learning has been widely used in many applications (e.g., self-driving cars, games, and robots). However, it is not very efficient when there are many actions and states. Thus, many researchers have applied Deep learning to it and call it Deep reinforcement learning (DRL) to address such a drawback. Unfortunately, recently, some researchers discovered a weakness of Deep learning in the test time and called it an adversarial attack. Since DRL is also Deep learning, it has the same weakness. Hence, DRL is vulnerable to adversarial attacks. Moreover, some existing works have created adversarial attacks for DRL. Generally, they first decide whether to create an adversarial example for the current state and then determine how much perturbation to add to the current state. This limits the adversary from the information in the previous steps when determining the perturbation. Also, some of the attacks fixed the adversarial action to the worst action, and then, the behavior of the target agent did not look natural. Therefore, we propose combining those two problems into one problem to allow the adversary to receive the information from the previous steps and let the adversary pick an adversarial action that is worth adding the perturbation by formulating and optimizing the problem. At last, we construct the experiment on Atari games to investigate the behavior of the agent attacked by our approach and compare our approach to the state-of-the-art attacks in terms of the amount of added perturbation and reward. As a result, our approach can make an agent play the games as if there was no adversarial attack and outperform the previous works.**

***Index Terms*—adversarial example, deep reinforcement learning, deep learning**

I. Introduction

Reinforcement learning (RL) has recently been combined with Deep Learning (DL) to improve its performance, and it is called Deep Reinforcement Learning (DRL). Further, DRL [15], [16] has been utilized by many applications (e.g., medical uses [19], autonomous vehicles [4], playing games [16], [20], and robotics [13]). However, the drawback of DRL is that it is vulnerable to adversarial attacks [8], [22] because it is based on DL. Existing work [2], [10], [12], [14], [21] shows and guarantees that there exist adversarial attacks in DRL. [10], [12], [14], [21] have discovered white-box attacks for DRL, and [2] has created a black-box attack for DRL. Further, in the white-box setting, [12], [14], [21] selected only some states to attack to avoid the detection.

Nonetheless, our proposed method focuses on a different solution. We assume that only selecting some states to attack can slightly help to avoid the adversarial-attack detection because the detector usually checks every state if it is an adversarial example. Further, the existing work has formulated two separated problems: finding which states to add perturbation and adding the perturbation to result in the worst action. However, we aim to combine those two problems and build a superior adversarial attack that constructs natural examples. Altering the original action to the worst action makes the target agent look unnatural. In addition, doing so can allow the adversary to observe the perturbation used in the previous states and use it as additional information to determine the amount of the added perturbation in the current state so that it can better optimize the total amount of the perturbation. It is worth noting that we utilize the optimization problem in Carlini and Wagner attack (CWA) [7] for our problem.

In our experiment, we use Deep Q-Network (DQN) [16] as our target. Further, the application that we will attack is Atari games (i.e., Breakout and Spaceinvader) because it is simple and generally used for comparison of the efficiency of several adversarial attacks. In the other words, we will train DQN to play games in Atari, use some state-of-the-art attacks and our attack to reduce the rewards of those models in the test time and make a comparison among them. Our major contributions can be summarized as follows: (i) We combine two problems which the previous works formulated into one problem and implement a method to solve this one problem based on CWA [7]. That is, we aim to reduce the perturbation for each state so that it is difficult for a detector to detect the attack. Further, the other aim is to avoid an observer to notice the abnormality of the agent's behavior by letting the adversary choose the action that is worth adding the perturbation. (ii) We construct the experiments for comparing our approach to the previous works in terms of the average perturbation, the maximum perturbation, the ratio of the perturbed states and the unperturbed states, the obtained reward and the behaviors of the agents.

This paper is organized as follows. Section II describes the adversarial attack on DRL while our approach is detailed in Section III. Experiments and demonstrations are in Section IV, while results and future work are discussed in Section V. Finally, Section VI offers our conclusions.

II. Background

A. Carlini and Wagner attack

This type of attack was named after the authors of [7], and they showed that it is a significantly strong attack for

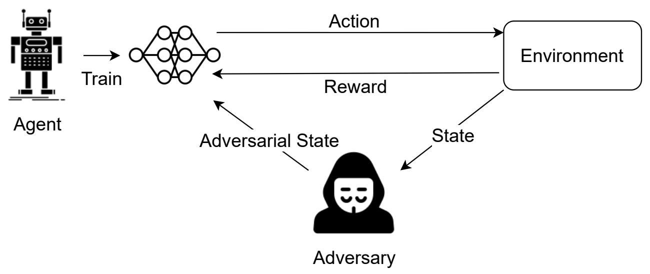


Fig. 1: Adversarial situation of DRL.

generating stealthy adversarial examples. Specifically, this attack was modified from the attack in [22] by omitting all the constraints by converting x^* to a function based on the hyperbolic tangent function of variable w so that the adversary could directly apply a dynamic gradient-based technique on w without clipping any value. That is, the optimization problem of this attack is

$$\min \left\| \frac{1}{2}(\tanh(w)+1) - x \right\|_2 + \alpha \cdot L(\frac{1}{2}(\tanh(w)+1)), \quad (1)$$

where $w \in \mathbb{R}^n$, α balances the distance between the minimization of the perturbation and the one of function $L(x^*)$ and $L(x^*)$ is a loss function; thus, we use $L(x^*) = \max(\max_{j \neq t} Z(x^*)_j - Z(x^*)_t, -\kappa)$, recommended by [7], where t is the targeted class and κ measures how much confidence the adversary desires.

B. Adversarial attack on DRL

In the general scenario illustrated in Figure 1, DRL consists of an agent and the environment. When the agent performs an action in a given state, the environment sends back the next state and a reward. Then, the agent determines the next action for the next state, and this keeps happening until the end of the episode. On the other hand, in an adversarial scenario, DRL also includes an adversary that lies between the agent and the environment. It receives the next state from the environment, alters it to an adversarial state and then relays to the agent.

C. Related works

In 2017, Huang et.al [10] first showed that in Atari games, by applying Fast Gradient Sign Method (FGSM) [8] on DRL's input observation (i.e., images of the games) to craft adversarial examples, a DRL model can be misled and results in much lower reward than it is supposed to obtain with the normal observation. Further, Kos and Song [12] performed FGSM to a state only when its value function is above a specified threshold and achieved the equal result as the one with [10]. That is, an adversary only needs to craft adversarial examples on critical states to mislead his/her targeted DRL and prevent detection by the agent. In addition, similar to DL, black-box attacks have been shown in [2], [10] by using an adversarial attack's transferability property [17], [18]. Then, in 2019, Lin et.al [14] crafted adversarial examples by using Carlini and Wagner attack (CWA) [7] on the states where the difference between the attack resulting highest value of the targeted agent's policy and the attack resulting the lowest value of the policy was above the specified threshold. Also, the authors proposed another attack that lured the agent to the designated state by using a generative model to predict the future states and determining the sequence of future actions. Moreover, Sun et.al [21] proposed an antagonist attack which trained an antagonist policy resulting in a score measuring how much this state impacted the targeted agent's reward and designated action to minimize the agent's reward.

Although, in a white-box setting, the works in [12], [14], [21] can be very effective because they only pick states in some time steps that is critical to add the perturbation to mislead the agent. Therefore, this can avoid the detection of adversarial examples when the agent is not aware of the threats in the environment at every time step; nonetheless, if the agent is aware of the threat for the environment at every time step, these works are not different to the work in [10] (which perturbs every state) in term of avoiding the detection. In addition, for an observer, it is easy for those attacks to be noticed since they suddenly alter the actions in totally different behaviors. For an example of an autonomous car, in a straight way, if the car suddenly turns left and hits a wall, this is very obvious that this car may be adversarially attacked. On the other hand, if the car gradually steps out from its own lane and hits the wall, the agent and the observers may not notice the attack. Noticeably, even though they use different strategies, the car has the same consequence which is hitting the wall. However, the latter can be less noticeable by an observer and also requires less perturbation for each state. Moreover, when those works add the perturbation, they directly apply the adversarial attack created for classification models (e.g., FGSM [8] and CWA [7]). Hence, this can limit the attack to consider only what happens in the current state and ignore the global view. Furthermore, some of the state-of-the-art attacks [21] relied on an additional learning model (e.g., a prediction model). However, we do not desire our attack to depend on any learning model because it is uncertain. Therefore, we will not compare our attack to the one in [21].

To the best of our knowledge, our approach is the first one to combine two separated problems into one problem to let the attack consider the global view. Further, our approach is not strict to alter the original action to the worst one when adding the perturbation. Instead, it can choose another action if the perturbation is too much for the worst action. This can make the behavior of the target smooth and similar to the one without adversarial examples.

III. Our approach

In this section, formalize the problem of the adversary in Section III-A and explain the method to minimize the added perturbation and the number of perturbed states by utilizing the information from the previous time steps in Section III-B.

A. Problem formulation

First, we have the target Deep Reinforcement Learning model or agent defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability of the state given a current state, a performed action and a next state, r is the reward function given a state and an action and γ is the discount factor. At each time step t, the agent is given an observation or state $s_t \in \mathcal{S}$ and choose the optimal action a_t according to s_t and its policy $\pi(s_t)$ which is the distribution over a_t in the case of stochastic and is the best action in the case of deterministic. To achieve the goal of the task, the agent would like to maximize $R = \sum_{t=0}^{T-1} \mathbb{E}_{a_t \sim \pi(s_t)} \gamma^t r(s_t, a_t)$ which is the total reward over T time steps. Note that γ is for reducing the significant of the future rewards.

In contrast, the adversary would like to reduce R by adding some perturbation to each s_t as $R_a = \sum_{t=0}^{T-1} \mathbb{E}_{a_t \sim \pi(s_t+\delta_t)} \gamma^t r(s_t, a_t)$ where δ_t is the perturbation added to s_t. However, at time step t, the adversary does not have information about the future. Thus, it needs to decide at the time step if it will add the perturbation and how much perturbation it would like to add. Based on CWA [7], these problems for each time step t can be formulated as

$$\min ||\delta_t(w_t)||_2 + \lambda_t \cdot \sum_{a \in \mathcal{A}, a \neq a_{true}} [D(F, a, s_t, \delta_t(w_t))] \quad (2)$$

where

$$D(F, a, s, \delta) = \max_{a' \in \mathcal{A}} [F(a', s+\delta)] - F(a, s+\delta),$$

and $\delta_t(w_t) = \frac{1}{2}(\tanh(w_t)+1) - s_t$, a_{true} is the action without the perturbation, λ_t is the regularizer parameter at time step t and $F(a,s)$ is the output of a DRL model given action a and state s. In the case of DQN, $F(a,s) = Q(a,s)$ that is a Q value given action a and state s. Noticeably, this formulation is a modified version of the one of CWA [7] for attacking a DRL model. The first part of (2) (i.e., $||\delta_t(w_t)||_2$) is to minimize the perturbation for this time step, and the other part is to decide which action is worth perturbing the state s_t. That is, the chosen action (i.e., a) makes its subtraction (i.e., $\max_{a' \in \mathcal{A}}[F(a', s_t+\delta_t(w_t))] - F(a, s_t+\delta_t(w_t))$) become zero. Additionally, the value of λ_t will be determined and discussed in Section III-B, and this involves the information from the previous time steps.

B. Method

Based on CWA [7], we use a gradient-based algorithm (e.g., SGD [5], Adam [11] and RMSprop [9]) to solve problem (2). Specifically, in each time step t, we set λ_t according to the ratio of the number of previous perturbed states (denoted by r_t) and the number of previous states and the average of $||\delta_\tau||_2$ where $0 \leq \tau \leq t-1$ (denoted by avg_t). The former can be calculated as

$$r_t = \frac{p}{t} \quad (3)$$

where p is the number of previous perturbed states. The latter can be calculated as

$$avg_t = \frac{\sum_{\tau=0}^{t-1} ||\delta_\tau||_2}{p}. \quad (4)$$

Algorithm 1: Method of the adversary at time step t.

```
Input: $s_t$, $avg_t$, $r_t$, $\eta$
Compute $\lambda_t = \frac{1-r_t}{avg_t}$;
Determine $\delta_t$ by optimizing problem (2) using a
  gradient descent method;
Pass $s_t$ to the target model and obtain an action $a$;
Pass $s_t + \delta_t$ to the target model and obtain an
  adversarial action $adv$;
if $a \neq adv$ then
  | $s_t = s_t + \delta_t$;
end
else
  | $\delta_t = 0$;
end
Result: $s_t$, $\delta_t$
```

Therefore, at time step t, to minimize the perturbation over the whole episode, λ_t should be proportional to $1 - r_t$ and inversely proportional to a_t. This can be calculated as

$$\lambda_t = \frac{1-r_t}{avg_t}. \quad (5)$$

To summarize the method of the adversary to optimize (2), at time step t, the adversary obtains state s_t. Then, it uses (5) for computing λ_t and can optimize problem (2) for η iterations. After that, it obtains perturbation δ_t and then uses it to check if this δ_t can alter the action. If not, just result the clean state s_t and zero perturbation. On the other hand, if δ_t can alter the action, result an adversarial example $s_t + \delta_t$ and δ_t. This can be summarized in Algorithm 1.

IV. Experiment

In this section, we explain how we construct the experiments for the comparison between our approach and other attacks. We use Python3, Tensorflow [1] and OpenAI Gym [6] for implementing the experiment and run it on Google Colaboratory Pro [3]. First, we specify what the target model is in Section IV-A and then discuss the attacks used in the experiment in Section IV-C. At last, the criteria for the comparison are demonstrated in Section IV-B, and the result will be shown and discussed in Section IV-D.

A. Target model

We use the Deep Q Network (DQN) model specified in [16] and train it with two Atari games (i.e., Breakout and Spaceinvader). The training setting of DQN for Breakout is described as follows. The learning rate and discount factor are respectively 0.00001 and 0.99 because these two values results in the best DQN in our experiment. Further, we convert the resolution of each frame of Breakout to the grayscale with the size of 84 by 84 to reduce the training time and improve its performance. To create a state in each time step, it uses four frames consisting of the current frame and the previous three frames, After training, we evaluate this model for five episodes, and the model can obtain an average reward of 278.4 which is considered as a strong player.

Furthermore, the training setting of DQN for Spaceinvader is as follows. The learning rate and discount factor are respectively 0.00001 and 0.99. We convert the color of each frame of Spaceinvader to grayscale with a size of 84 by 84. Also, similar to the setting in Breakout, we use four frames to create a state. As a result, we achieve the average reward of 414.44.

B. Criteria

In our experiment, there are four criteria used to make a comparison among the three attacks: the average perturbation (avg), the ratio of the number of previous perturbed states and the number of states (r), the maximum perturbation (max) and the reward. These first three criteria can be computed as:

$$avg = \frac{\sum_{\tau=0}^{T} ||\delta_\tau||_2}{p},$$
$$r = \frac{p}{T},$$
$$max = \max_{\tau=0,1,..,T} ||\delta_\tau||_2,$$

where T is the last time step, and p is the number of perturbed states.

C. Attacks

The attacks that we use in this experiment are our approach, simple attack [10] and strategically-timed attack [14]. Simple attack is naive because it creates an adversarial example for every state. In [10], the authors used FGSM [8] for the attack in each state. However, we strengthen this attack by using CWA [7] instead since it is the most powerful attack for classification models. Further, we use this attack to target the least-value action.

Strategically-timed attack is similar to Simple attack; nonetheless, it chooses the state where the difference of the highest-value action's value and the least-value action's value exceeds the specified threshold. In our experiment, we use three thresholds: 0.1, 0.15 and 0.2.

At last, in our approach, there is only one hyperparameter which is η. Thus, for Breakout, we have further experimented on it, and Figure 2a and 2b respectively show that when the average perturbation is inversely proportional to the ratio of the number of perturbed states and the number of clean states. Since we focus on reducing the average perturbation, we use $\eta = 50$ for the experiment. Further, we did not find significance of max perturbation and obtained reward over η as seen in Figure 2c and 2d.

Furthermore, I need to determine η used in DQN for Spaceinvader. According to Figure 3a, 3c and 3d, the best η is 140 because the agent achieved the lowest average perturbation, maximum perturbation and reward. Although the ratio is high, it is not constant over η. Therefore, we decided to use $\eta = 140$ for DQN for Spaceinvader.

D. Result

We let each attack play five episodes of Breakout and determine the averages of the criteria resulted from the five episodes. The results can be demonstrated in Table I where SimA is Simple attack, and n-StratA is Strategically-timed attack with the threshold of n.

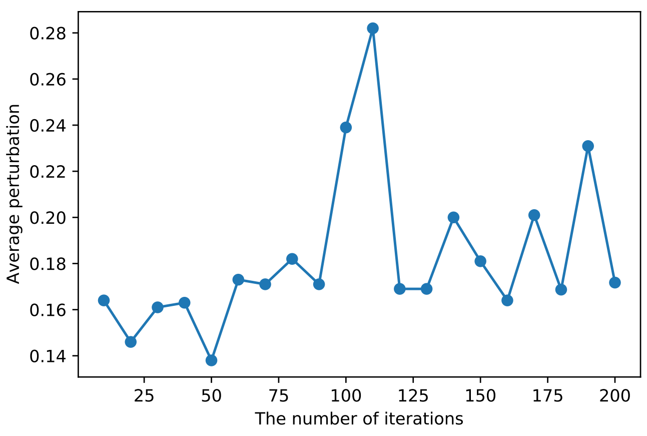
Noticeably, although SimA can result in the reward of 0, SimA performs the worst among the attacks since max and r are the highest among the attacks. Next, 0.1-StratA has similar results as SimA; however, r is much lower than the one of SimA because it does not add perturbation to every state. Further, 0.15-StratA and 0.2-StratA have a higher reward than one of 0.1-StratA because they have significantly less perturbed states as shown in r. However, avg of those three attacks are still at the same level; hence, they are not very different against an adversarial-example detector.

In addition, our approach has a similar result as the ones of 0.15-StratA and 0.2-StratA in a term of max. Nevertheless, it has more perturbed states than the ones of 0.15-StratA and 0.2-StratA because our approach does not focus on reducing the number of perturbed states. Explicitly, our approach has the best result in a term of avg since the adversary has the information from the previous time steps and aims to minimize the overall perturbation. Also, our approach can achieve a significantly low reward compared to the rewards achieved by 0.15-StratA and 0.2-StratA and essentially outperforms SimA and 0.1-StratA.

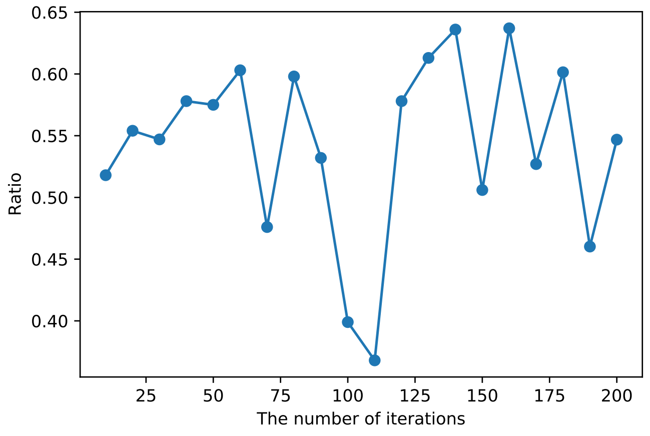
We also experiment on Spaceinvader by letting each attack play three episodes and obtain the average value of each criterion. The result of this experiment is shown in Table II. As expected, the results obtained by SimA and 0.1-StratA are similar; however, their ratios are significantly different. The rewards obtained by 0.15-StratA and 0.2-StratA are worse than (higher than) the ones in SimA and 0.1-StratA. At last, our approach can achieve the best results among all the attacks in terms of the average perturbation, the maximum perturbation and the reward.

V. Discussion and Future Works

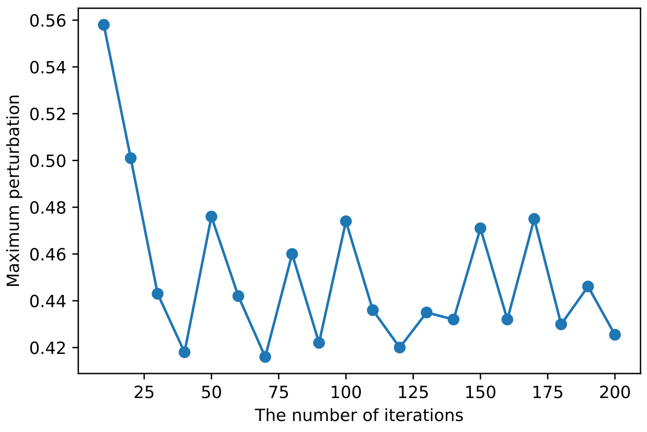
From the experiment of Breakout, the agents in SimA, 0.1-StratA, 0.15-StratA and 0.2-StratA try to avoid catching the balls on purpose. On the other hand, the agent in our approach misses catching the ball as if it was a mistake from the training step. The reason is that our approach does not restrict the adversary to choose only the worst action; however, it can choose any action that is worth adding the perturbation. Therefore, our approach has higher chances to fool an observer from noticing the strange behavior of the agent caused by adversarial examples. For example, when we run the learning model attacked by our approach, we randomly pick a state at time step t shown in Figure 4 with $r_t = 0.47$ and $avg_t = 0.12$. Noticeably, the ball is going down to the left, and we use the normal DRL model without any attack. It results in moving left as the action for this state. Certainly, when we alter this state with CWA which simple attack and strategically-timed attack use, the model results in moving right which is the worst action. However, when we use our approach to alter the state, the target model results in staying still as the action. As can be noticed, either moving right or staying still results in the same thing which is that the ball will be dropped. Further,



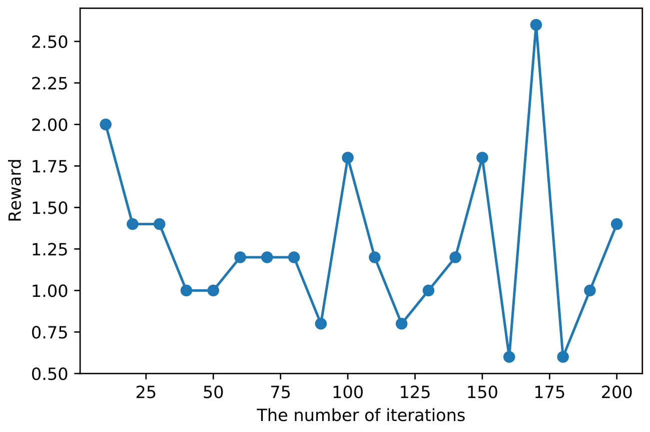
(a) Average perturbation over the numbers of iterations (η) for Breakout.



(b) Ratio (r) over the numbers of iterations (η) for Breakout.



(c) Maximum perturbation over the numbers of iterations (η) for Breakout.



(d) Obtained reward over the numbers of iterations (η) for Breakout.

Fig. 2: Results of DQN on Breakout over the numbers of iterations (η).

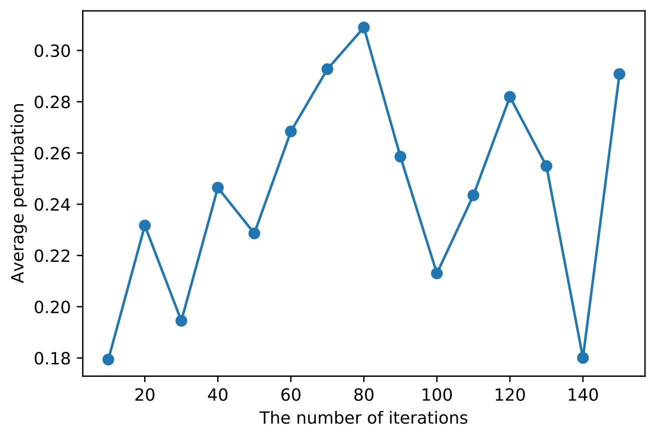
TABLE I: Criterion's values for comparing the attacks in Breakout

Attack Criterion	SimA	0.1-StratA	0.15-StratA	0.2-StratA	**Our approach**
avg	0.205	0.208	0.238	0.278	**0.138**
r	1	0.602	0.238	**0.143**	0.575
max	0.696	0.686	**0.41**	0.42	0.476
Reward	**0**	**0**	3.2	3	1

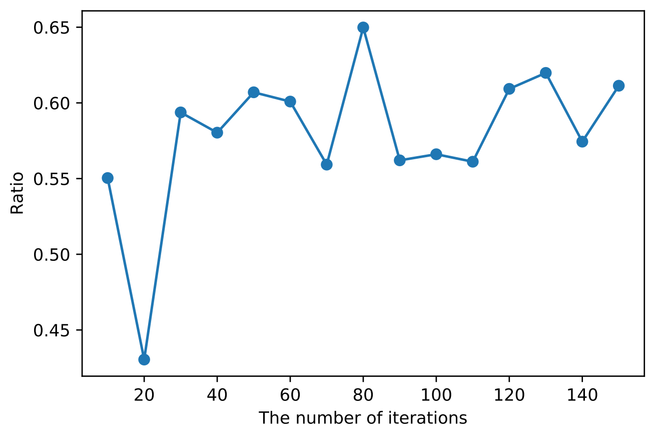
TABLE II: Criterion's values for comparing the attacks in Spaceinvader.

Attack Criterion	SimA	0.1-StratA	0.15-StratA	0.2-StratA	**Our approach**
avg	0.425	0.416	0.635	0.663	**0.180**
r	1	0.414	0.431	**0.305**	0.574
max	1.146	0.817	1.579	1.339	**0.488**
Reward	153	140	375	288	**43**

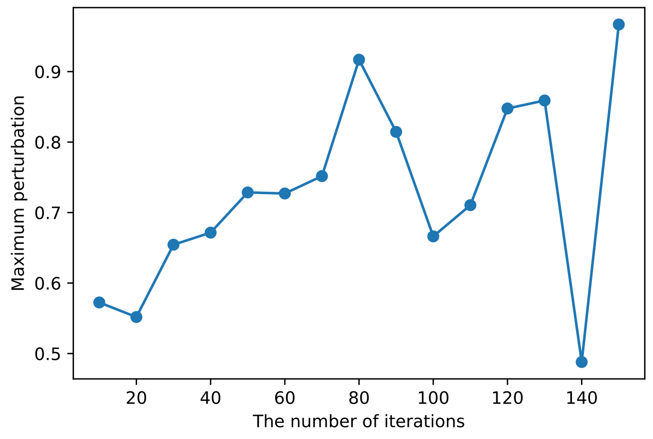
the added perturbation caused by CWA and our approach is 0.39 and 0.37 respectively. Clearly, our approach requires less perturbation, results in the same thing as CWA does and also looks less adversarial than CWA. Furthermore, in the experiment of Spaceinvader, the agent in our approach tried to dodge some lasers shot from the enemies and died by accident. On the other hand, the agents in other attacks went to hit lasers shot by the enemies on purpose. Yet, they sometimes went to the area where there is no enemy in vertical and waited until the enemies successfully came to the bottom as shown in Figure 5. Explicitly, the agents in the other attacks unnaturally lost (as if the DQN was misled by something), and



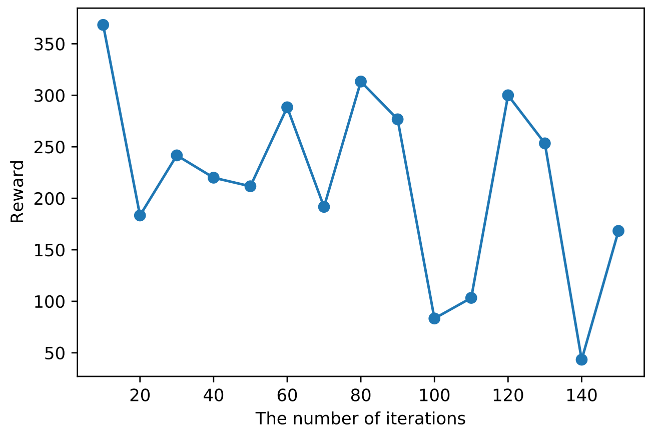
(a) Average perturbation over the numbers of iterations (η) for Spaceinvader.



(b) Ratio (r) over the numbers of iterations (η) for Spaceinvader.

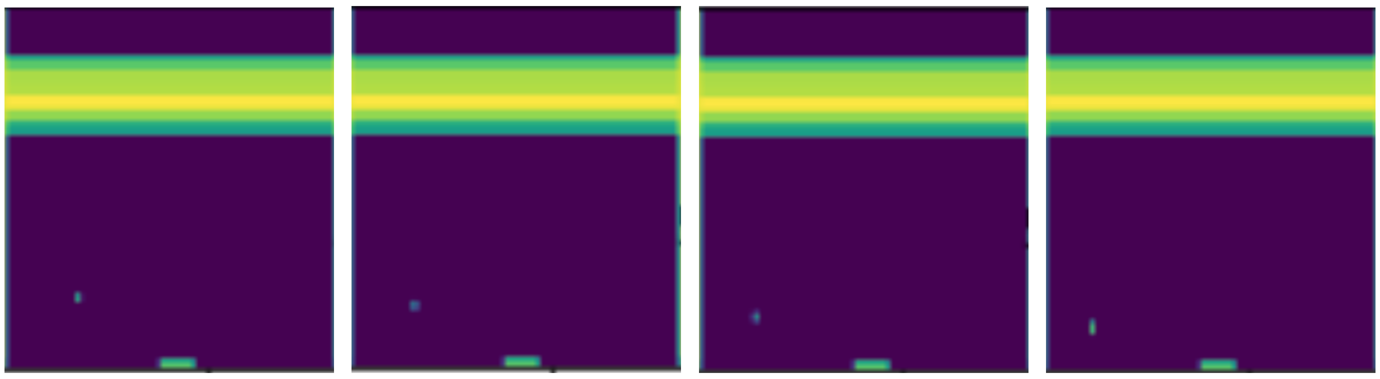


(c) Maximum perturbation over the numbers of iterations (η) for Spaceinvader.



(d) Obtained reward over the numbers of iterations (η) for Spaceinvader.

Fig. 3: Results of DQN on Spaceinvader over the numbers of iterations (η).



(a) Frame $t-3$ (b) Frame $t-2$ (c) Frame $t-1$ (d) Frame t

Fig. 4: A state at time step t in Breakout with our approach.

the agent in our approach lost as if there was no adversarial attack existing. However, the limitation of our approach is that it results in higher max than some state-of-the-art attacks in games that are easy to fail (e.g., Breakout). Thus, further research is needed to reduce this value improve our approach's strength and immunity to any adversarial-example detector.

VI. Conclusion

We investigated the previous state-of-the-art adversarial attacks in both DL and DRL. Then, we used and modified

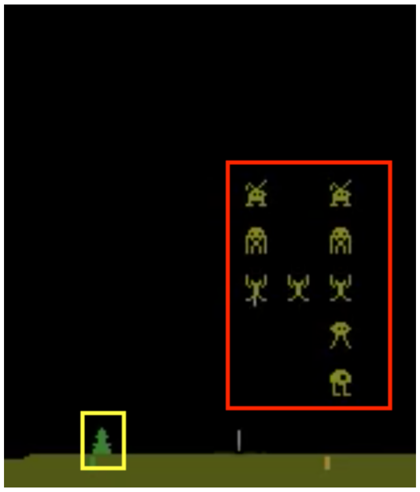


Fig. 5: The agent (in the yellow rectangle) in 0.15-StratA waits in the area where there is no enemy vertically until it loses in Spaceinvader where the enemies are in the red rectangle.

them to improve the efficiency of the attacks in DRL (e.g., reducing the average perturbation in each state). Finally, we have discovered a new technique to create natural adversarial examples to attack DRL in the test time.

Specifically, we combined two separated problems in the previous works into one problem and formulated it as one optimization problem. Then, we proposed a greedy method to solve the problem by applying a gradient-based algorithm. The result from the experiment showed that our approach could outperform the previous works in a term of the average perturbation. Further, the agent in our approach could naturally play Breakout and Spaceinvader as if there were no adversarial examples existing in its states. In contrast, the agents in other attacks explicitly showed their strange behaviors implying that there was something wrong with their states, and an observer would eventually realize that the model was under adversarial attacks.

At last, this work shows a more stealthy attack than the state-of-the-art ones, and we are aware that this work can allow an adversary to create natural adversarial examples for DRL models. Nevertheless, DRL model developers can also acknowledge this attack and make their models more robust to this adversarial attack. Furthermore, by doing so, DRL models can be stronger and especially gain more trust from people who would like to use the DRL models in several critical applications.

ACKNOWELEDGEMENTS

The authors would like to thank the anonymous reviewers for their feedback. The authors also thank the Department of Computer Science and the Baylor AI group at Baylor University for partially supporting this research.

REFERENCES

[1] Martín Abadi. Tensorflow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pages 1–1, 2016.

[2] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[3] Ekaba Bisong. *Google Colaboratory*, pages 59–64. Apress, Berkeley, CA, 2019.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[5] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8), 2012.

[10] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

[13] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[14] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[17] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[18] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[19] Pablo Rivas-Perea, Erich Baker, Greg Hamerly, and Bryan F Shaw. Detection of leukocoria using a soft fusion of expert classifiers under non-clinical settings. *BMC ophthalmology*, 14(1):1–15, 2014.

[20] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[21] Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang Liu. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5883–5891, 2020.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.