IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 29, 2022, accepted May 9, 2022, date of publication May 13, 2022, date of current version May 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174963

Black-Box Reward Attacks Against Deep Reinforcement Learning Based on Successor Representation

KANTING CAI, XIANGBIN ZHU, AND ZHAO-LONG HU
College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, China

Corresponding author: Xiangbin Zhu (zhuxb@zjnu.cn)

This work was supported in part by the National Natural Science Foundation of China, under Grant 62103375; in part by the Social Development Project of Zhejiang Provincial Public Technology Research, China, under Grant LQ19F030010; and in part by Ministry of Education in China (MOE) Project of Humanities and Social Sciences, China, under Grant 19YJCZH056.

ABSTRACT Although the deep reinforcement learning (DRL) technology has been widely adopted in various fields, it has become an important research hotspot to study the vulnerability of DRL for improving the robustness of DRL agents. The adversarial attack methods based on white-box models, where the adversary can access all the information of victims, have been intensively investigated. However, in most practical situations, the adversary cannot obtain the internal information of the victim's neural network. Furthermore, for reward-based attacks, the agent can perform anomaly detection on the perturbed rewards to detect whether it has been attacked. In this paper, we propose a black-box attack method with corrupted rewards, which employs DRL exploration mechanisms to improve the effectiveness of attacking agents. The adversary builds a deep neural network in advance to learn the successor representation (SR) of each state. Then, the adversary can determine the timing of attacks and generate imperceptible adversarial perturbations based on the values of the SR. Experimental results show that the black-box attack algorithm based on SR proposed in this paper can effectively attack agents with fewer adversarial samples.

INDEX TERMS Black-box attacks, corrupted rewards, deep reinforcement learning, successor representation.

I. INTRODUCTION

In the past few years, deep reinforcement learning (DRL) technology has been widely applied in various fields, such as self-driving cars [1] and go [2]. However, at the same time, we need to pay attention to the fact that when a newly developed technology is applied to the real world and widely promoted, the small flaws in the technology will be magnified, which can cause enormouse damage. As a combination of deep learning (DL) and reinforcement learning (RL), DRL inherits the shortcomings of DL, which is that it is vulnerable to adversarial samples. Therefore, when an adversary attacks the neural network of DRL agents, it will prevent the agent from learning the optimal policy during training or from choosing the optimal action during testing. Especially in some high-stakes areas, these weaknesses can often cause massive unintentional damage. To this end, it is necessary to study the vulnerability of DRL to improve the robustness of DRL technology.

The attacks against DRL agents can be divided into black-box attacks and white-box attacks according to the attacker's knowledge of the victim's model. The main adversarial DRL research so far focused on white-box attack models. In this model, adversaries need obtain the parameter values of the agent network and query the Q-value of each state-action pair, and then leverage this information to generate adversarial samples for attacks, which can often achieve great effectiveness. For example, strategically-timed attacks need to obtain the corresponding Q-value of different actions of the current state to determine the timing of attacks [3]. The perturbation attack against action space also needs to obtain the model of the agent to change the agent's action [4]. Although the above attack methods can achieve great attack results, there are many real applications where the adversary

The associate editor coordinating the review of this manuscript and approving it for publication was Hao Wang.

cannot obtain the above information. The attacks carried out in such situations, that is, without the information concerning the victim's model, are called black-box attacks.

The reward poisoning attack occurs when the attacker perturbs the reward received by the agent during training so that the agent learns a wrong policy. In the recent research works, the attack against rewards often only considers the success rate of the attack, and there is no research on whether the perturbed rewards are imperceptible. The point is, these corrupted rewards that fall outside the range of environmental rewards are very easy for the agent to detect and to realize that it is under attack.

The exploration method is a very important part of the learning process of RL agents. In general, the exploration mechanism is mainly used to help agents speed up the exploration of the environment. In this paper, we consider exploiting the agent's exploration strategy to improve the efficiency of attacks. We propose a black-box attack method with corrupted rewards in training time, which uses the successor representation (SR) to attack agents effectively. The SR [5], [6] implicitly represents the expected discounted sum of access frequencies for each successor state from the current state in the future. It can be combined with rewards to provide information for decision-making. Machado *et al.* proposed a positive correlation between the sum of the absolute values of the elements in the SR value of the state-action pair and the number of times the state has been visited [6]. Therefore, the adversary can use the SR value of each state-action pair as the basis for determining the attack timing and attack direction and generate adversarial perturbations.

Compared to previously proposed methods that use the Q-value difference of local states for attack timing determination, our approach uses the successor representation to determine the attack timing. This makes the attack determination from a holistic perspective and maximizes the impact of a single attack on the learning of an intelligent agent. At the same time, in contrast to other black-box attacks, attacks on the returns of the agent can act directly on most contexts without the need to design specialized attack samples, as in the case of attacks on states. In contrast, in past studies of adversarial reward attacks, attackers often only pay attention to the effect achieved after the attack without considering whether the attack is reasonable in the current state and whether the value domain is different between different states in a given context. Our attack approach takes this into account by limiting the size of the postattack reward to the rewards that can be obtained in the current state.

The adversary first establishes a neural network, called the SR-Network, by using the knowledge of the environment that the adversary has. Then, the adversary trains the SR-Network by interacting with the environment. During the attack process, the SR value from the SR-Network can be utilized to determine the attack timing to reduce the number of attacks. The SR-Network can also be used to generate the perturbation against the reward of the environment. We have verified the algorithm proposed in this paper through simulation experiments. The results show that our algorithm can improve the efficiency of attacks, and significantly reduce the number of attacks. This paper makes the following contributions:

1) We utilize exploration mechanisms to design adversarial black-box attacks against DRL.
2) We propose a covert reward poisoning attack method, by strictly limiting the size of the perturbation on the reward which makes the attack difficult to detect.
3) The SR value is employed to determine when to inject the adversarial sample into the reward of the environment to reduce the number of attacks and help to design the size of the perturbation.

II. RELATED WORKS

In the past research, various research results have been proposed with respect to black-box attacks.

Behzadan *et al.* [7]proposed policy induction attacks. The attack conducts adversarial perturbations to the observations of the DQN model. The adversary can obtain the environment information of the attacked agent and the rewards of the environment. The attacker builds a deep neural network according to the input type of the attacked agent, and trains the network to generate adversarial perturbations, so that the trained policy of the attacked agent tends to choose an action other than the optimal action in a specific state. This attack method uses the idea of black-box attacks for deep learning in classifier fields.

Gleave *et al.* [8]proposed a new black-box attack method, which attacks agents through policies. The attack is applied in the zero-sum game scenario. The adversary cannot change the observation results of the attacked agent on the environment, but establishes a ''natural observation'' as the adversarial perturbation. There are no ''natural observations'' in the training samples of the victim. Thus when the victim learns through these samples and faces the normal training data again, it will choose bad actions. The experimental results show that the attacker can achieve a successful attack through ''natural observation''. In this experiment, if the victim successfully evades the opponent's attack, the victim wins; otherwise, the opponent wins. After a period of training, the winning rate of the victim to the ''natural observation'' opponent is approximately 0.86, while that of the normal opponent is only 0.47.

Inkawhich *et al.* [9]proposed a snooping attack method in which the attacker can only monitor the states, the actions, and the rewards of the attacked agent. The attack trains and obtains a model of the agent by following this information and uses the model to generate adversarial samples. This method is more feasible for some attacks on an agent deployed on the server side.

Majadas *et al.* [10]proposed flipping reward attacks (FRS), which is a black-box attack method on rewards during training. The adversary first defines an attack probability p, $0 < p < 1$. When the reward returned by the environment is not 0, the attacker generates a random number between

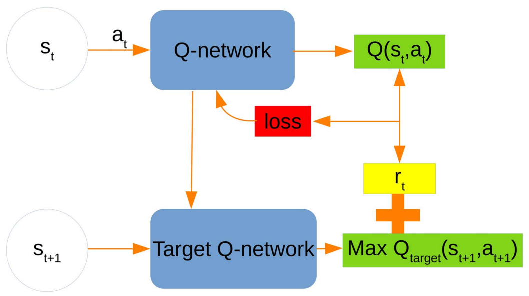


FIGURE 1. **DRL learning model.**

0 and 1, and then when the random number is less than p, the attack is performed. This will change the sign of the environmental reward, and feed this corrupted reward back to the agent. Experiments show that when the attack probability p is greater than 0.4, it can effectively prevent the agent from learning a bad policy for the environment. When p is less than 0.4, the agent can also learn a suboptimal policy.

III. BACKGROUND

A. DEEP REINFORCEMENT LEARNING

Deep reinforcement learning(DRL) adopts the Markov decision process(MDP) model as its environment model, which is described by states, actions, rewards and transition probabilities. At each time step, the agent takes the state of the environment as the input, obtains the Q-value of each action at the current state through deep neural networks, and chooses one of the actions for execution according to the action selection algorithm. After the action is executed, the agent combines the reward returned by the environment with the new state, the previous state and the action to form a tuple, which is stored in the replay buffer. Then, the data in the buffer are used for learning. The goal of DRL is to find an optimal policy to maximize its long-term cumulative reward through learning.

Deep reinforcement learning can be divided into model-based methods and model-free methods. The model-based method establishes the corresponding state transition model through the observed knowledge, while the model-free method directly trains a deep neural network to find the optimal policy. Here, we mainly study the deep Q-network (DQN) algorithm. In the DQN method, the agent obtains the optimal policy by solving the Bellman equation. The update process of its Q-value can be recursively defined as:

$$Q_t(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q_{t+1}(s_{t+1}, a')) \quad (1)$$

where s_t is the state at time step t, a_t is the action that the agent chooses to execute, s_{t+1} is the next state at time step $t+1$, r_t is the reward from the environment, $Q_t(s_t, a_t)$ is the Q-value of state-action pair (s_t, a_t), α indicates the learning rate and γ is the discount factor.

B. BLACK-BOX ATTACKS

In terms of attacks against DRL, because the learning of the agent is extremely dependent on various types of information observed, the adversary can attack the agent by perturbing the states and the rewards of the agent, and make the agent learn a wrong policy. The attack method of the agent can be divided into white-box attacks and black-box attacks according to the knowledge possessed by the attacker. In past research, scholars have proposed many attack methods.

In the white-box attack, the attacker can fully access the structure, parameters and training data of the target neural network. Most of the existing attack algorithms are white-box attacks, such as adversarial attack methods based on transformation networks [11] and path vulnerable point attacks [12]. However, this assumption is overidealized. In the real world, attackers often cannot obtain the above information. When the adversary cannot obtain the information concerning the target neural network, this attack method is usually called a black-box attack [8], [13].

We usually meet the following limitations when we want to perform attacks on deep reinforcement learning agents on some commercial platforms.

- The data inside the target model are not visible to us, and we do not have access to the relevant model structure, parameters, or training data.
- When collecting information about the interaction of the agent with the environment, only the state, actions, and returns are available, and it is impossible to determine the magnitude of the target intelligence's Q-value for each action.

Papernot *et al.* [14] proposed a black-box attack method using surrogate models and adversarial sample transferability to solve the above problems. Transferability refers to when an adversarial example can successfully attack a model, it is also likely to successfully attack another similar model. The surrogate model means that when we want to carry out a black-box attack on a model, we can first train a local model with a similar decision boundary to the target model, and then perform a white-box attack on the model to obtain adversarial samples, and then use the transferability to attack the target. The attack on the model, the model deployed locally is the surrogate model. Using the above two properties, we can successfully attack an agent without obtaining internal information of the target agent.

Black-box attacks against deep reinforcement learning can be classified from the perspective of the attack target. State-based attack methods include policy induction attacks [7], the attack method based on policy imitation [15], snooping attacks [9], etc; Reward-based attack methods include Trojan attacks [16], CopyCAT algorithms [17], and U2 algoithms [18]; Adversarial policy attacks [8] is a policy-based attack method; Gradient band-based adversarial training attacks [19] is an environment-based attack method.

C. EXPLORATION MECHANISM

The goal of the exploration mechanism is to minimize the learning time of the agent. When an agent efficiently explores environmental information, it can have more knowledge about the environment; therefore, it can speed up the learning process and converge to the optimal policy quickly. At the same time, the exploration mechanism is also able to reduce the resources consumed by the agent's learning process. At the heart of the exploration mechanism is balancing exploration and exploitation, and it has been shown that it is usually beneficial to find a suitable balance between exploration and exploitation. Exploration means that the agent uses the current knowledge to identify and find the policy or action with the highest long-term reward. If the agent does not explore and only chooses the most rewarding action at present, that is, using the exploitation strategy, then the agent may not be able to find a truly optimal policy.

In recent years, many kinds of exploration strategies have been designed. In general, exploration strategies can be divided into undirected exploration and directed exploration [20]. The main feature of undirected exploration is that the action is generated based on a certain random distribution without considering the learning process itself. These strategies include random exploration, semiuniform distributed exploration and Boltzmann distributed exploration [20]. The directed exploration strategies make full use of the information generated in the learning process, and leverage this knowledge to guide the exploration process, such as counter-based exploration [20] and E-value based exploration [21].

However, most of the exploration methods, such as the counter-based exploration mechanism and Boltzmann distribution exploration mechanism, are localized and hence can only reflect the knowledge of the current state but cannot reflect the information of adjacent states, and are difficult to be applied in continuous space. Choshen *et al.* [21] proposed a novel exploration strategy based on the exploration value, called E-value, which establishes a neural network for exploration in addition to the agent's neural network, in which the E-value represents the unknown knowledge contained in state-action pairs.

IV. BLACK-BOX ATTACKS BASED ON SUCCESSOR REPRESENTATION

A. SUCCESSOR REPRESENTATION

Successor representation is a generalization of the correlation between the current state and the subsequent state, where the successor state refers to the state that occurs subsequently for a given policy.

For the convenience of presentation, we have the following definitions:

Definition 1: $\psi^{\pi}(s_t, j)$. *A value represents the successor representation of state j along trajectories starting from state s_t under policy π.*

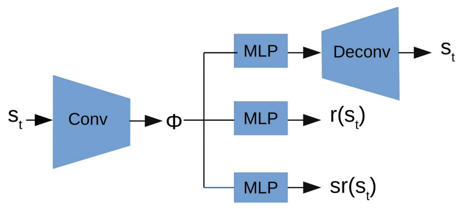


FIGURE 2. SR-Network.

Then the update of $\psi^{\pi}(s_t, j)$ can be expressed as:

$$\psi^{\pi}(s_t, j) = (1-\alpha)\psi^{\pi}(s_t, j) + \alpha((I(s_t == j) + \gamma\psi^{\pi}(s_{t+1}, j) \quad (2)$$

for all $j \in S$, and η denoting the step-size. The SR also corresponds to the Neumann series of γP_{π}:

$$\Psi_{\pi} = \sum_{t=0}^{\infty} \gamma^t P_{\pi}^t = (I - P_{\pi})^{-1} \quad (3)$$

where $P_{\pi}(s'|s) = \frac{n(s'|s)}{n(s)}$, $n(s'|s)$ means the number of times from state s to state s', $n(s)$ means the number of times agent visit state s.

Barreto *et al.* [22] proposes successor features (SF), which is an extension of the successor representation for state-action pairs. SF can be represented by $\psi^{\pi}(s_t, a, j)$.

Definition 2: $\psi^{\pi}(s_t, a, j)$. *Represents the successor representation of action a in state s_t with respect to state j.*

For convenience, we make the following definitions:

Definition 3: $SR(s_t, a)$. *Represents the L_1 norms of the SF value of each action in state s_t.*

The SR-Network can learn the rewards of each action in each state. According to the SR-Network, the agent can calculate the Q-value, denoted as Q_{sr}, through the successor representation $SR(s_t, a)$ and the reward $r(s_t)$. Figure 2 shows the deep network structure of the SR-Network.

From Figure 2, we can observe that the total loss is from three aspects. Thus, for the training of the SR-Network, it is necessary to calculate the loss value for each aspect, and then add the three loss values to obtain the total loss value, so the update step can be carried out.

Machado *et al.* [6] proposed substochastic successor representation(SSR):

Definition 4: SSR. *Let $\tilde{P}_{\pi}$ denote the substochastic matrix induced by the environment's dynamics and by policy π such that $\tilde{P}_{\pi}(s'|s) = \frac{n(s'|s)}{n(s)+1}$. For a given $0 \leq \gamma < 1$, the substochastic successor representation,$\tilde{\Psi}_{\pi}$, is defined as:*

$$\tilde{\Psi}_{\pi} = \sum_{t=0}^{\infty} \gamma^t \tilde{P}_{\pi}^t = (I - \tilde{P}_{\pi})^{-1} \quad (4)$$

After calculation, Machado presented the following equation:

$$\frac{\gamma}{n(s)+1} - \frac{\gamma^2}{1-\gamma} \leq (1+\gamma) - \left\|\tilde{\Psi}_{\pi}(s)\right\|_1 \leq \frac{\gamma}{n(s)+1} \quad (5)$$

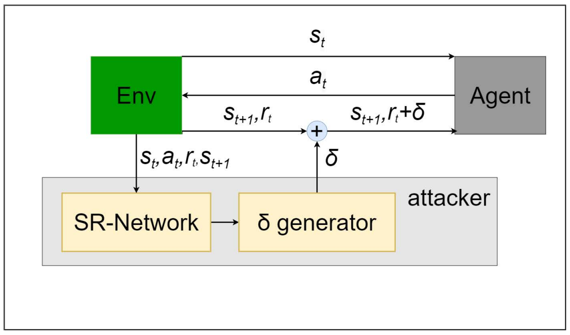


FIGURE 3. The attack model.

where $0 \leq \gamma < 1$. From this, we can see that the size of the SSR L_1 norms is proportional to the number of visits. Since $n(s)$ is approximately equal to $n(s) + 1$ in the environment of continuous space, Therefore, Equation (3) is approximately equal to Equation (4). Thus SR can be used to recover an estimate of the visit counts, which can be used for exploration.

B. THE ATTACK MODEL

Our goal is to attack the agent's neural network with black-box models. In this work, we study reward-based attacks in the training stage. To achieve the goal of the attack, we assume that there are the following assumptions:

1) The adversary can inject perturbations into the rewards of the environment. In other words, an adversary can access the reward information returned by the environment to the agent and change the reward value.
2) The adversary can obtain the tuple $< s_t, a_t, r_t, s_{t+1} >$, generated by the interaction between the agent and the environment in each time step t.

Under the above assumptions, the adversary conducts training the SR-Network in the environment where the attacked agent is located before the attack starts. The SR-Network is employed as the basis for determining the attack timing and the size of perturbations. During the training of the agent, the attacker collects the interaction information tuple $< s_t, a_t, r_t, s_{t+1} >$ between the agent and the environment, and uses state s_t of the agent and the size of the SR value corresponding to action a_t to determine whether to attack, and to calculate the size of the perturbation against the reward.

Figure 3 shows the attack process based on the attack model. When the agent obtains the state of the environment, it will choose an action based on the action-selection rules, and the environment will change because of this action and return a reward. The adversary obtains this information through observation to determine the timing of the attack through the SR-Network, and generates disturbances to modify the reward returned by the environment.

C. ATTACK METHODS

In this section, we introduce SR-based reward attacks against black-box environments during the training phase. For each state, we can use the value of SR to determine the number of visits of each action in that state[7]. Generally, after sufficient training, the more times an action is performed in the same state, the greater the Q-value corresponding to the action. Meanwhile, the value of SR will increase with the number of visits, and each visit will perform the action with the largest Q-value in the current state with a high probability. Thus, we can infer that after sufficient training, when the agent visits a state more times, the greater the number of times the action corresponding to the maximum Q-value in this state is performed, the greater the corresponding SR value. Therefore, we can estimate the visit times of a state by the maximum SR value of the state. Obviously, after sufficient training, when a state is visited more times, the state becomes more important in the process of the agent's leaning. Attacking the agent in this state allows us to obtain a better attack effect. Thus, we can determine the timing of the attack by the maximum SR value in each state. In addition, the adversary can determine the size of the perturbation through the SR value. The adversary can also reduce the reward of the action with the larger SR value and increase the reward of the action with the smallest SR value so that the attacked agent is biased to perform the action with a small original Q-value. Therefore, the purpose of the attack can be achieved with fewer attacks. Because this is to induce the agent to learn a bad policy by changing the value of the reward, it is a reward poisoning attack.

When an adversary attacks, there are two important factors. One is the timing of the attack and the other is the size of the perturbations. For the timing of an attack, if the adversary attacks every time step, it will have a huge attack cost. In the process of agent learning, the attack effect of most states is relatively low. Hence we need to determine the best time for adversarial attacks to reduce the number of attacks and the cost of attacks. On the other hand, when the attacker carries out a reward poisoning attack and generates an adversarial perturbation to the reward, the size of the perturbation must be considered. If the corrupted reward exceeds the maximum reward or is below the minimum reward that the original environment can provide, the attack behaviour is easily detected by the agent. However, if the perturbation is too small, the purpose of the attack may not be achieved.

In summary, our goal is to find an attack method to complete effective attacks with as few attacks as possible and with as little perturbation as possible.

1) ATTACK TIMING

Because the agent's neural network is a black box to the adversary, the attacker needs to rely on its own trained network to determine the attack timing.

From the properties of the SR-Network, the action with the most execution times in the same state has the largest SR value, while the action with fewer execution times has a smaller SR value. Obviously, the Q-value of the action with the largest Q-value is always greater than that of other actions, so the difference between the SR value of the action with the largest Q-value and the action with the smallest Q-value in this state will become increasingly larger. By judging the size of the difference of the SR value in a state, we can determine whether the state should be attacked.

Definition 5: Y(s,a). A value represents the effect of adding a disturbance when state s performs action a.

Theorem 1: The larger the difference between the maximum and minimum values of SR in a state, the higher the profit of attacking the state when the disturbance is the same.

Proof: Suppose there are two states s_1, s_2, their actions with the maximum SR value and the minimum value are s_{1max}, s_{1min}, s_{2max}, s_{2min}, respectively, and s_{1max}-s_{1min} > s_{2max}- s_{2min}. Because the SR value represents the connection to the successor state, $SR(s, a) = \sum_{j \in S} \psi(s, a, j)$ and $\psi(s, a, j) = \sum_{i=1}^{\infty} r^i(s_i == j)$. Then when we add a disturbance δ to the action, the effect of the disturbance on state j is $Y(s, a, j) = \sum_{i=1}^{\infty} r^i(s_i == j) * \delta = \psi(s, a, j) * \delta$. The effect on all states is $Y(s, a) = \sum_{j \in S} \psi(s, a, j) * \delta = SR(s, a) * \delta$. Therefore, the larger the $SR(s, a)$ is, the greater the impact after the disturbance is added. At the same time, as seen from the above, if the difference between the maximum value and the minimum value of SR is larger, the number of visits will be greater. At the same time, as the number of visits increases, the number of executions of each action in the same state will also increase, so if s_{1max}-s_{1min} > s_{2max}- s_{2min}, then for every action a, $SR(s_1, a) > SR(s_2, a)$. Obviously $\sum_{a \in A} SR(s_1, a) > \sum_{a \in A} SR(s_2, a)$. In the case of adding the same perturbation, one can obtain $\sum_{a \in A} Y(s_1, a) > \sum_{a \in A} Y(s_2, a)$. Therefore, the greater the difference between the maximum value and the minimum value of SR in a state, the greater the impact, and the higher the income of the attack. □

We make the following definitions:

Definition 6: dis_SR:A value represents the difference between the maximum SR value and the minimum SR value in a state.

$$dis\_SR(s_t) = \max_{a \in A} SR(s_t, a) - \min_{a \in A} SR(s_t, a) \tag{6}$$

Definition 7: Max_dis_SR. A value represents the maximum value of the difference between the maximum SR value and the minimum SR value in the same state in all states.

$$Max\_dis\_SR = dis\_SR(s_t) \; s_t \in S \tag{7}$$

The timing of the attack, whether the state needs to be attacked, can be determined by the following formula:

$$dis\_SR > \beta Max\_dis\_SR \tag{8}$$

where β is an adjustment parameter. The attacker's hyperparameter β is a predetermined number, and its size affects the attack frequency. For reward-intensive environments, the value of β needs to be smaller so that the attack frequency increases. In contrast, for environments where most of the rewards are 0, β can be set larger so that the attack frequency is lower.

This approach has two main advantages. The first is that the more times a state is accessed, the more important the state is, so higher benefits can be obtained by attacking the state. On the other hand, we can stably attack some of these states through the difference of SR since these differences do not change over the training process, which can effectively ensure that the agent converges to the wrong policy.

2) PERTURBATION SIZE

In the existing research, the attack based on observations often only considers reducing the size of the perturbation as much as possible to avoid being detected. However, this is not enough for attacks based on rewards. For instance, if the reward of the environment is limited to the set {0, 1}, in a time step with a return reward of 0, the corrupted reward after adding the perturbation to the original reward becomes 0.05, or the reward range is [-1,1], and the corrupted reward after adding the perturbation is -1.1. These corrupted rewards obviously exceed the range of the original environmental reward, even if the attack perturbation is small, so that they are easily detected by the attacked agent. In this paper, we impose strict constraints on the perturbation size of the reward poisoning attack. We make the following definition:

Definition 8: (Covert Reward Poisoning Attacks). In reward poisoning attacks, when the reward after the attacker adds perturbation belongs to the normal reward value of the environment, we call this attack covert reward poisoning attacks (CRPA).

The reward poisoning attack could be successful only if the Q-value of the target action after being attacked is greater than the Q-value of other actions. Thus in order for the agent to learn the wrong policy, we need to add a sufficiently large perturbation to the reward of the environment to the agent. Zhang *et al.*[22] show that when the added perturbation is greater than a certain threshold, we can successfully attack the agent's policy. However, the threshold in this paper is $\frac{(Q^* - Q^\pi)}{2}$, where Q^* and Q^π represent the Q-value of the optimal policy and the Q-value of the target action without attacks, respectively. According to the nature of the SR-Network, we can obtain the range of real Q-values through the SR networks trained in advance. However, it is obvious that the newly generated reward does not conform to the CRPA, so we need constrain the size of perturbations.

Before the attack, we let the adversary train the SR network on the environment where the attacked agent is located, obtain the SR network under normal conditions to approximate the Q-value of each state learned by the attacked agent and the reward value fed back by different actions in each state, and record the maximum reward value R_{max} and minimum reward value R_{min} during the training.

For the convenience of explanation, we define the following:

Definition 9: $a_{sr}(s_t)$. The action with the minimum SR value when the state is s_t

$$a_{sr}(s_t) = arg \min_{a \in A} SR(s_t, a) \tag{9}$$

Definition 10: $a_q(s_t)$. The action with the maximum real Q-value when the state is s_t.

$$a_q(s_t) = arg \max_{a \in A} Q(s_t, a) \tag{10}$$

Definition 11: $R(s_t)$. The value set of the rewards when the status is s_t.

$$R(s_t) = \{r(s_t, a)|a \in A\} \tag{11}$$

Definition 12: $Q_{sr}(s_t)$. When the state is in s_t, the SR-Network is used to calculate the Q-value obtained by approximating the real DQN.

Firstly, we obtain the $R(s_t)$ of the current state s_t, which represents the reward of each action in this state. For instance, we assume that there are four actions and $R(s_t) = \{A, B, C, D\}$. We add R_{max} and R_{min} to $R(s_t)$ to form the new reward set. Therefore, we define the set by the following:

Definition 13: $RD(s_t)$. The set consists of R_{min}, R_{max} and the elements of $R(s_t)$ in the state s_t.

When sorting $RD(s_t)$, the sorted $RD(s_t)$ is $\{R_{min}, A, B, C, D, R_{max}\}$. We also obtain $SR(s_t)$ from the SR-Network. According to the properties of the SR-Network, we can obtain $Q_{sr}(s_t, a_q(s_t))$ and $Q_{sr}(s_t, a_{sr}(s_t))$ through $Q_{sr}(s_t)$. According to the definition in [22], when the size of the perturbation is greater than

$$d_t = \frac{Q_{sr}(s_t, a_q(s_t)) - Q_{sr}(s_t, a_{sr}(s_t))}{2} \tag{12}$$

the attack can be established. However, to meet the conditions of a covert reward poisoning attack, that is, the size of the corrupted reward is selected from the set $RD(s_t)$, so the perturbation needs to be obtained from the set formed by subtracting the environmental reward from the elements in $RD(s_t)$. For the convenience of explanation, we make the following definitions:

Definition 14: $DR(s_t)$. The set is formed by subtracting the environmental reward r_t from the elements in $RD(s_t)$ in the state s_t.

$$DR(s_t) = \{x - r_t|x \in RD(s_t)\} \tag{13}$$

When the action is $a_{sr}(s_t)$, the perturbation δ_t is set as:

$$\delta_t = min\{x|x \in DR(s_t) \text{ and } x > d_t\} \tag{14}$$

However, it is possible that d_t is too large and there is no qualified reward in $RD(s_t)$, so we directly define:

$$\delta_t = R_{max} - r_t \tag{15}$$

When the action is not $a_{sr}(s_t)$,the perturbation δ_t is set as:

$$\delta_t = min\{-x|x \in DR(s_t) \text{ and } x < -d_t\} \tag{16}$$

If there is no eligible reward in $RD(s_t)$, we directly define:

$$\delta_t = r_t - R_{min} \tag{17}$$

Thus, by adding this perturbation δ_t to the original reward, the agent can be effectively prevented from learning an optimal policy.

In summary, the generation of perturbation δ_t can be represented by the Figure 4.

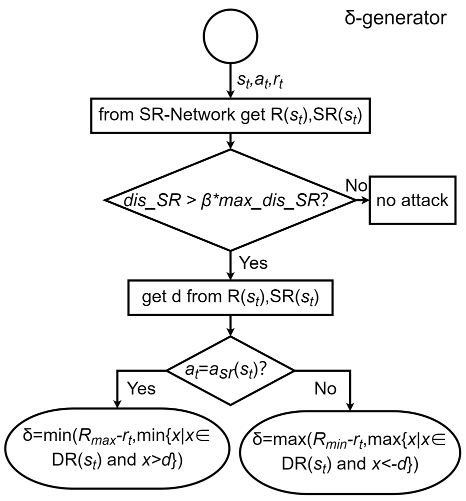

δ-generator
s_t,a_t,r_t
from SR-Network get R(s_t),SR(s_t)
dis_SR > β*max_dis_SR?
No
no attack
Yes
get d from R(s_t),SR(s_t)
$a_t=a_{sr}(s_t)$?
Yes
No
δ=min(R_{max}-r_t,min{x|x∈DR(s_t) and x>d})
δ=max(R_{min}-r_t,max{x|x∈DR(s_t) and x<-d})

FIGURE 4. The δ-generator.

D. FEASIBILITY ANALYSIS

From Theorem 1, we can conclude that using SR to determine the timing of an attack maximizes the impact caused by a single attack. Because the SR network is trained in advance and does not change its parameters as the agent is trained, the attack will be carried out at a fixed state-action pair.

Suppose state s satisfies Equation (8) and action a_1 is the action with the largest Q value and action a_2 is the action with the smallest SR value; then, the attacker will attack the agent in that state. According to Equation (12), we derive $d = \frac{Q(s,a_1)-Q(s,a_2)}{2}$. Since the attacker will perform a stable attack on the action of that state, suppose the Q value of a_1 after the attack is $Q'(s, a_1)$, and the Q value of a_2 is $Q'(s, a_2)$. Then according to Equation (14) and Equation (16), we can derive

$$Q'(s, a_1) \leq Q(s, a_1) - d,$$

and

$$Q'(s, a_2) \geq Q(s, a_2) + d.$$

Since $Q(s, a_1) = Q(s, a_2) + 2d$, we have

$$Q'(s, a_1) \leq Q(s, a_1) - d = Q(s, a_2) + d \leq Q'(s, a_2).$$

Therefore, the agent then selects the action with the smallest SR value.

The SR-based CRAP(SR-CRAP) algorithm is shown in Algorithm 1.

V. EXPERIMENT RESULTS

In this section, we investigate the efficacy of the SR based covert reward poisoning attacks(SR-CRPA) algorithm on the Pong environment, Pendulum environment and Freeway environment of Gym.

A. EXPERIMENTAL ENVIRONMENTS

To verify the effectiveness of the attack algorithm, we conduct experiments in Atari's Pong environment, Freeway

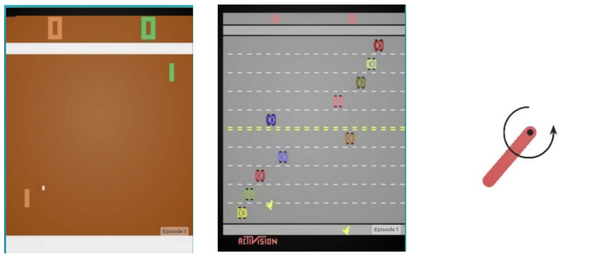
Algorithm 1 SR Based Covert Reward Poisoning Attacks

```
Require: hyperparamter $\beta$, the maximum reward value $maxr$,the minimum reward value $minr$; trained SR network SR; the maximum steps $maxepoch$
  $maxsr \leftarrow 0$
  for t = 0,1,...,maxepoch: do
    the victim generates data $(s_t, a_t, r_t, s_{t+1})$
    the attacker observes data $(s_t, a_t, r_t, s_{t+1})$
    obtain $Q(s_t)$,$SR(s_t)$,$R(s_t)$ through the SR network.
    $dis\_SR = \max_{a \in A} Q(s_t, a) - Q(s_t, argmin_{a \in A} SR(s_t, a))$
    if $dis\_SR > maxsr$ then
      $maxsr = dis\_SR$
    end if
    if $\beta maxsr < dis\_SR$ then
      $d_t = \frac{\max_{a \in A} Q(s_t,a) - Q(s_t, argmin_{a \in A} SR(s_t,a))}{2}$
    end if
    $DR(s_t) = \{x - r_t | x \in R(s_t)\}$
    if $a == argmin_{a \in A} SR(s_t, a)$ then
      if $\{x \mid x \in DR(s_t)$ and $x > d_t\}$ is not empty then
        $\delta_t = \min\{x \mid x \in DR(s_t)$ and $x > d_t\}$
      else
        $\delta_t = maxr - r_t$
      end if
    else
      if $\{x \mid x \in DR(s_t)$ and $x < -d_t\}$ is not empty then
        $\delta_t = \max\{x \mid x \in DR(s_t)$ and $x < -d_t\}$
      else
        $\delta_t = minr - r_t$
      end if
    end if
    the victim receive data $(s_t, a_t, r_t + \delta_t, s_{t+1})$
  end for
```

environment and the classic control environment Pendulum, and compare the experimental results of the agent under SR-CRPA attacks with the experimental results with other attack methods. All the three environments are shown in Figure 5.

In the Freeway environment, a chick needs to get from one side of the highway to the other within a specified time, but there will be cars passing by constantly on the highway, and when the chick touches these cars, it will back off by a certain distance. When the chick reaches the end, the player will get a point, and the chick will instantly return to the starting point and start crossing the road again. After a period of time, we counted the total number of times the chick has successfully crossed the road, that is, the score.

In the Pong environment, the agent and the opponent are located at both ends to hit a small ball together. When the ball is hit by the agent and the opponent does not hit it, the agent scores one point. When the opponent hits the small ball and the agent does not, the opponent gets one point. The game ends when either side's score reaches 21 points. The total score of the agent is its score minus the opponent's score.



(a) Pong (b) Freeway (c) Pendulum

FIGURE 5. Experimental environments.

In Pendulum, the agent applies a force to a fixed length wooden stick, namely the pendulum, making the pendulum move around the centre, and the environment gives a reward according to the angle of the pendulum. When the angle is smaller, the greater the reward given, that is, when the pendulum is just at the direction of 0 o'clock, the reward given by the environment is 0. When the angle is larger, the reward given by the environment is smaller. In particular, the reward given by the environment in 6 o'clock direction is approximately -16.2736. Over a period of time, we calculate the sum of the rewards as the score for this run. Because the environment is a continuous action space, it is not suitable for processing in DQN. Hence, the original action space $[-2, 2]$ is discretized into 11 actions, namely, $[-2, -1.6, -1.2, \ldots.., 1.6, 2]$.

B. EVALUATION OF ATTACK EFFECTS

To compare the effectiveness of the attack, we conducted experiments in Freeway, Pong and Pendulum. The Freeway and Pong environments are environments with discrete reward values. In this environment, the agent is rewarded only when it reaches the goal, and the rewards are all 0 at the rest of the time. In the Pendulum environment, the agent will receives a reward at each time step. Obviously, the latter is often more difficult to attack than the former.

We also added two kinds of attacks for comparison: The first one is the flipping reward attack (FRS) mentioned above. The other is the attack that uses the Q-value as the basis to determine the timing of the attack. This attack employs a pretrained Q network, and finds the maximum difference of between the maximum of the Q-values and the minimum of the Q-values in each state. Then, a hyperparameter c, which is less than 1, is defined. When the agent reaches a certain state, where the difference between the maximum of the Q-values and the minimum of the Q-values is greater than the product of the maximum Q-value and c, the adversary attacks. The idea follows the strategically-timed attack proposed by Lin et al[5]. We call this attack a black-box DQN attack. Due to the lack of an SR-Network, we cannot scale the size of the perturbation. Hence we directly use R_{min} and R_{max} as the corrupted reward. For comparison with our proposed attack method, we adjust the size of the hyperparameter c so that the attack frequencies in the Pong, Freeway and Pendulum environments are 0.038, 0.01 and 0.6, respectively. The above three values are the attack frequencies that we obtain through the SR-CRPA attack.

Figure 6 is a comparison of the attack performance of the agent in the Pong, Freeway and Pendulum environments with different attack methods. The solid line represents the average value of multiple experiments, and the shadow around the solid line represents the standard deviation of multiple experiments in this time step.

Figure 6(a) shows the comparison with different attack methods in the Pong environment. In this experiment, we executed 1500 episodes under different attack methods. We define the agent learning rate as 0.9999, and each data point is the average of the scores of this episode and the scores of the next 19 episodes. In this game, the best score of the agent is 21 points, and the worst score is - 21 points. As shown in Figure 6, the red line is the score of the agent without attacks. From the figure, we can see that the agent basically achieves the optimal 21 points after 700 episodes of training, and the blue line is the score after the SR-CRPA attack. Under the SR-CRPA attack, the agent's score always remains below - 20 points. Therefore, the SR-CRPA attack can effectively attack in the Pong environment. The green line is the experimental result under the flipping reward sign attack. In this experiment, we set the attack probability to 25%. It can be seen that the learning speed of the agent decreases significantly under the FRS attack, and the score curve becomes more unstable, but it will ultimately approach the best policy. The yellow line is the experimental result under the black-box DQN attack. It can be seen that in the Pong environment, the effect of the black-box DQN attack is good, but it is slightly less than our SR-CRPA attack.

Figure 6(b) shows the experiential results in the Freeway environment. In this experiment, we executed 1000 episodes under different attacks, and the agent learning rate was defined as 0.9999. Each data point is the average of the scores of the current episode and the subsequent 19 episodes. In this game, it is difficult to define the optimal score because of the randomness of vehicles. Generally, a score of more than 22 points is regarded as success. In the worst case, the chicken does not cross the road successfully even once, that is, the score is 0. As shown in Figure 6(b), the red line is the training score of the agent without attacks. From the figure, we can see that without attacks, the agent's score in each game after 800 training rounds has basically exceeded 22 points. The blue line is the score under the SR-CRPA attack. The agent's score is 0, from which we can conclude that the SR-CRPA attack can effectively attack in the Freeway environment. The result of the Freeway experiment under the FRS attack is shown by the green line. In this experiment, we also set the attack probability to 25%. The standard deviation of the agent's score under the attack is very large. It can be seen that the experimental result is unstable, but it will slowly converge and tend to a suboptimal strategy in the end. The yellow

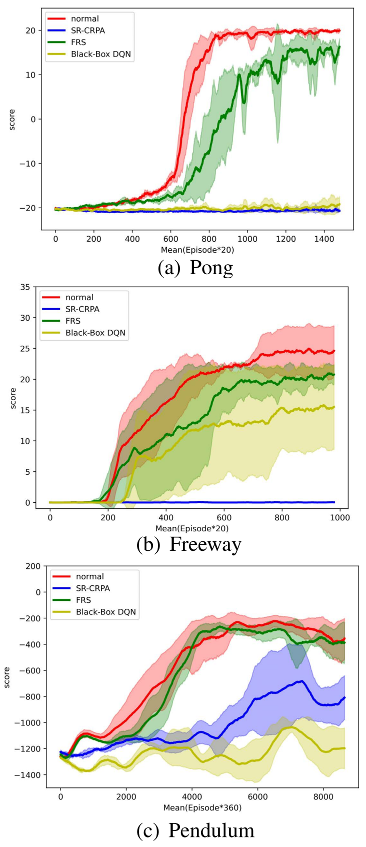


FIGURE 6. Experimental results of the three environments.

area is the performance of the agent being attacked by a black-box DQN attack. In this environment, the performance of black-box DQN attacks is relatively poor. The reason may be that there are many variables in the environment in the Freeway environment, so that the attacker cannot completely attack the agent only by determining the attack timing with the Q-value.

Figure 6(c) shows the experimental results in the Pendulum environment. In this experiment, we executed 9000 episodes under different conditions. The learning rate of the agent is defined as 0.9999, which is different from that of the previous two. The difference between this environment and the previous two is that in the Pendulum environment, the initial position of the clock pendulum at the beginning of each episode is randomly generated, which has a particularly large impact on the total reward of the environment. Hence we increase the number of averages for each data point, using the score of this episode and the scores of the following 359 episodes are averaged. The results are shown in the figure. The red line is the training score of the agent that has not been attacked. From the above figure, we can see that the score of the agent in each game after 800 training sessions before being attacked basically exceeds 22 points. The blue line is the score under the SR-CRPA attack. After being attacked, compared to the result of the discrete reward environment, the environment is still fitted in the Pendulum environment, but we can see that although part of the learning is successful, the final result does not reach the normal level. The green line is the result of the FRS attack, where the attack probability is 0.25. It can be seen that it is very close to the normal level, and the learning process of the agent has not been greatly affected. The yellow curve is the result of the black-box DQN attack. It can be seen that the attack is better than the SR-CRPA attack at the same attack frequency. This is because Pendulum has multiple reward values in each state, while the SR-CRPA attack is different from the Pong and Freeway environments. It will choose to inject less disturbance into the reward, and the black-box DQN attack does not have the SR network, so it directly uses R_{min} and R_{max} to attack. However, in the Pendulum environment, if the reward frequently returns the maximum reward 0 and the minimum reward -16.2736, the attack is very easy to be detected by the victims. Thus, compared with the black-box DQN attack, the SR-CRPA attack requires more attacks, but the attack is not easy to detect.

We generally judge the degree of adaptation of the agent's policy to the environment by the score in the game, and the lower the score, the poorer the fitness of the agent's policy to the environment. For an untrained agent, when it comes to a new environment, the score it gets is often the lowest, e.g., -21 in Pong environment and 0 in Freeway environment, with only a tiny chance to get a relatively high score. Therefore, we can let the agent perform multiple games in the environment and record the corresponding score data to reflect the fit of an agent's policy and the environment.

We first recorded the scores of the agents in each environment when they were untrained and then performed Black-Box DQN or SR-CRPA, or FRS attacks on the agents, respectively, while training them. After saving the trained models, we recorded the scores of these models performing the tasks in the environment compared with the scores of the untrained agents, where each dataset was collected with 500 data points. The Kolmogorov-Smirnov test(a.k.a. K-S test) was used to verify the distributional similarity between the datasets. In general, two data sets are considered from the same distribution if the Kolmogorov-Smirnov test yields a p-value greater than 0.05.

In the Pong environment, with the model trained with FRS attacks, the p-value of the Kolmogorov-Smirnov test is 1.94e-119, which shows that the difference is very significant. For the model trained with Black-Box DQN attacks, the p-value is 6.75e-10, which is better than that of FRS attacks, but the difference with the data before training is relatively significant. With the model trained with SR-CRPA attacks, the p-value is 0.11, which is greater than 0.05, and it can be decided that it is the same distribution. Therefore, by the Kolmogorov-Smirnov test, we can determine that the fit of the agents to the environment after being attacked by SR-CRPA in the Pong environment is similar to the fit of the untrained agents and the environment.

In the Freeway environment, the p-values of FRS, Black-Box DQN, and SR-CRPA attacks are 1.06e-239, 1.34e-199, and 1.0, respectively, and it is evident that the score distribution of SR-CRPA attacks is the same as that of the untrained agents. At the same time, the score distributions of FRS, Black-Box DQN, and untrained agents differed significantly.

In the Pendulum environment, the p-values of FRS, Black-Box DQN, and SR-CRPA attacks are 0.0, 7.30e-220, and 0.0, respectively. The FRS and SR-CRPA attacks did not prevent the agents from fitting the environment, while the Black-Box DQN attack effect was relatively improved, but still very poor.

C. THE RATE OF BAD ATTACKS

In the process of an attacker's attack on an agent, in most cases, the agent cannot determine whether it is under attack. Nevertheless, it can be judged by some particular states. For example, in the Pong and Freeway environments, if the agent receives the message that the round is not yet over. Nevertheless, the reward is not 0, or in the Pendulum environment, the agent is in a state other than the 12 o'clock direction, but the reward is 0. Then, the agent can quickly determine that it is under attack. We call this attack Bad Attack, and use the rate of bad attacks (RoBA), to express the number of bad attacks.

$$RoBA = \frac{the\ number\ of\ bad\ attacks}{the\ number\ of\ actions} \tag{18}$$

We attacked with SR-CRPA and black-box DQN with similar attack frequencies in Pong, Freeway, and Pendulum, respectively, and calculated their RoBA. The results are shown in Figure 7.

The x-axis in Figure 7 represents the experiments conducted in the three environments, and the y-axis represents the RoBA. Because the RoBA in Pendulum is too large, we divide the RoBA of the two attacks in that environment by 10 for comparison. From the figure, we can see that the RoBAs of SR-CRPA and Black Box-DQN in Pong and Freeway are not significantly different. SR-CRPA changes the reward value after the attack to $RD(s_t)$. Black Box-DQN directly sets the reward after the attack to the maximum or minimum value in the reward domain, while the rewards in Pong are only -1, 0, and 1. The rewards in Freeway are only 0 and 1. Therefore, the RoBAs of the two are not very different when the attack frequencies are similar. The RoBA of SR-CRPA in Pendulum is significantly smaller than that of Black Box-DQN because the reward domain of each state in Pendulum is a set. The rewards of Black Box-DQN after the attack are only the maximum and minimum values of the reward domain and can be easily identified. At the same time, the SR-CRPA will choose the values in the original reward domain. Therefore, the RoBA of SR-CRPA is smaller than that of Black Box-DQN. Because the rewards in the Pendulum environment are more significant as they go up, the RoBA of SR-CRPA appears slightly larger, and the attack will maintain a lower value in a similar environment in Freeway.

D. EFFECTS OF THE β VALUE ON ATTACK TIMES

In Section 4.2.1, we choose the attack timing, that is, when $dis\_SR > \beta Max\_dis\_SR$, the attacker will attack the agent.

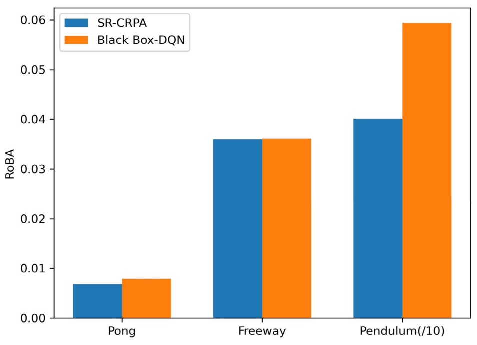


FIGURE 7. The rate of bad attacks in different environments.

Obviously, the larger the β value is, the fewer states that meet the attack conditions and the lower the attack frequency. We use the rate of attacks (RoA) to express the number of attacks. RoA is the ratio of the total number of attacks by attackers to the total number of actions performed by agents:

$$RoA = \frac{the\ number\ of\ attacks}{the\ number\ of\ actions} \tag{19}$$

Figure 8 shows the effect of different β values on the RoA. Figure 8(a) shows the result in the Pong environment. Obviously, as the β value increases from 0.55 to 0.61, the ROA decreases from 0.042 to approximately 0.03. Figure 8(b) shows the RoA affected by β in the Freeway environment. As a result of β increasing from 0.6 to 0.9, the value of RoA decreases from 0.14 to less than 0.01. Figure 8(c) shows the result in the Pendulum environment, where when β increases from 0.2 to 0.32, the RoA decreases from 0.65 to approximately 0.32. In summary, the attack frequency of the SR-CRPA attack will decrease with increasing β.

E. EFFECTS OF THE β VALUE ON THE ATTACK RESULTS

In the previous section, we have shown that the number of attacks by adversaries increases with increasing β. However, with a decrease in the number of attacks, it must also have an impact on the efficacy of attacks.

Figure 9(a) shows the results of the agent in the Pong environment. The experiment was repeated 100 times in the environment to generate data. From this figure, we can see that when the value of β is less than 0.56, the attack is very successful, which can make the agent score completely at the lowest score. As β increases, when β is 0.57, it is still at a very low score. When β is 0.58, the score oscillates approximately 0, there are many extreme data points, and the performance of the agent is very unstable. When β is 0.59, it is basically stable at 0 points, and when β is equal to 0.6 and 0.61, the agent has a certain advantage over the opponent in the Pong game, but it still cannot reach the highest score.

The experimental results in the Freeway environment are shown Figure 9(b). The experiment was also repeated

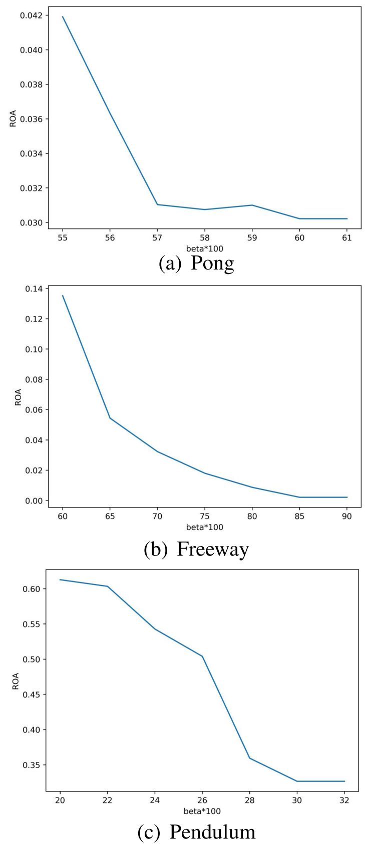
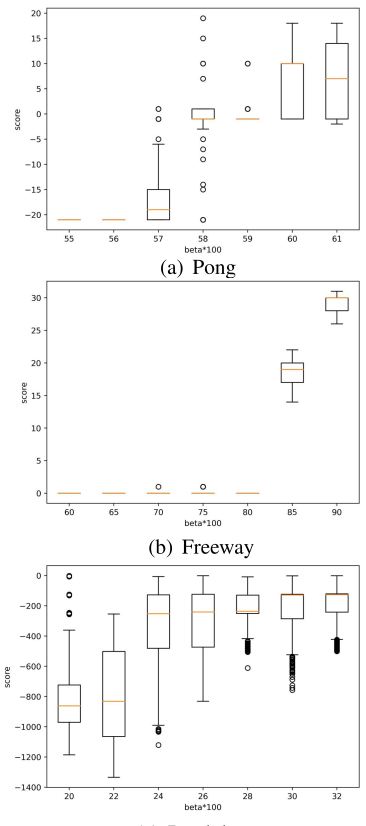


FIGURE 8. The rate of attacks affected by β.

100 times in the environment to generate data. By analysing the figure, we can observe that when the value of β is less than 0.8, the attacked agent cannot learn the optimal policy from the environment. When the β value is 0.85, the agent can learn a better policy from the environment, and the chicken can cross the road about 18 times in each game. When the β value is 0.9, the attack has little effect on the learning of the agent, and the agent learns a policy with a very high score.

Figure 9(c) shows the experimental results in the Pendulum environment, where we conducted this experiment 100 times to generate the experimental data. From this figure, we can see that when the value of β is less than 0.24, the score of the victim on the environment is relatively low, and the median is around 900 points. When the β value is 0.24 to 0.26, the agent has a certain improvement compared to the previous, but it is lower than the results without attacks. When the β value is above 0.28, the agent can learn a good policy from the environment normally. By comparing the impact of β on the attack frequency in the previous subsection, we can find that in the Pendulum environment, only when the RoA is greater than 50% can the attack have an impact on the learning of the agent. There are two reasons for this. The first is, that to prevent the attack from being discovered, we impose certain constraints on the size of the attack, which cannot have a large enough impact through a single attack. On the other



(a) Pong

(b) Freeway

(c) Pendulum

FIGURE 9. The attack effects affected by the β value.

hand, each state of the Pendulum environment can generate rewards, and the closer the direction is to the zero point, the greater the reward. This leads to the fact that the Q-value of the action in the direction closer to the zero point is much larger than that of other actions in all states. Thus, a large RoA is required in the Pendulum environment to generate an effective attack.

VI. CONCLUSION

In this paper, we use the properties of SR to propose a new reward poisoning attack method, which can achieve effective attacks on DRL agents. This method uses the trained SR network to determine the attack time and the attack size, so that the agents learn a bad policy. The main idea of this attack is that the norm of the SR can be used to guide exploration, and it should also be used to guide attacks. We have conducted extensive experiments with our attack method in different DRL environments. The results show that the SR-CRPA attack method has demonstrable effects in a variety of experimental environments, especially in environments with sparse rewards, and it can achieve the attack target through a small number of attacks. However, an environment with dense rewards needs more attacks. The main advantage of out attack method is that it is a critical moment attack. In future work, our research will focus on targeted attacks.

REFERENCES

[1] A. R. Fayjie, S. Hossain, D. Oualid, and D.-J. Lee, "Driverless car: Autonomous driving using deep reinforcement learning in urban environment," in *Proc. 15th Int. Conf. Ubiquitous Robots (UR)*, Jun. 2018, pp. 896–901.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. [Online]. Available: https://www.nature.com/articles/nature16961

[3] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," Nov. 2017, *arXiv:1703.06748*.

[4] X. Y. Lee, S. Ghadai, K. L. Tan, C. Hegde, and S. Sarkar, "Spatiotemporally constrained action space attacks on deep reinforcement learning agents," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 4577–4584. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5887

[5] E. Vértes and M. Sahani, "A neurally plausible model learns successor representations in partially observable environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 13714–13724.

[6] M. C. Machado, M. G. Bellemare, and M. Bowling, "Count-based exploration with the successor representation," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 5125–5133. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5955

[7] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *Proc. Int. Conf. Mach. Learn. Data Mining Pattern Recognit.* Cham, Switzerland: Springer, 2017, pp. 262–275.

[8] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," 2019, *arXiv:1905.10615*.

[9] M. Inkawhich, Y. Chen, and H. Li, "Snooping attacks on deep reinforcement learning," Jan. 2019, *arXiv:1905.11832*.

[10] R. Majadas, J. García, and F. Fernández, "Disturbing reinforcement learning agents with corrupted rewards," Feb. 2021, *arXiv:2102.06587*.

[11] E. Tretschk, S. J. Oh, and M. Fritz, "Sequential attacks on agents for long-term adversarial goals," Jul. 2018, *arXiv:1805.12487*.

[12] X. Bai, W. Niu, J. Liu, X. Gao, Y. Xiang, and J. Liu, "Adversarial examples construction towards white-box Q table variation in DQN pathfinding training," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2018, pp. 781–787.

[13] X. Pan, C. Xiao, W. He, S. Yang, J. Peng, M. Sun, J. Yi, Z. Yang, M. Liu, B. Li, and D. Song, "Characterizing attacks on deep reinforcement learning," Jul. 2019, *arXiv:1907.09470*.

[14] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, New York, NY, USA, Apr. 2017, pp. 506–519, doi: 10.1145/3052973.3053009.

[15] V. Behzadan and W. Hsu, "Adversarial exploitation of policy imitation," Jun. 2019, *arXiv:1906.01121*.

[16] P. Kiourti, K. Wardega, S. Jha, and W. Li, "TrojDRL: Trojan attacks on deep reinforcement learning agents," Feb. 2019, *arXiv:1903.06638*.

[17] L. Hussenot, M. Geist, and O. Pietquin, "CopyCAT: Taking control of neural policies with constant attacks," Jan. 2019, *arXiv:1905.12282*.

[18] A. Rakhsha, X. Zhang, X. Zhu, and A. Singla, "Reward poisoning in reinforcement learning: Attacks against unknown learners in unknown environments," 2021, *arXiv:2102.08492*.

[19] T. Chen, W. Niu, Y. Xiang, X. Bai, J. Liu, Z. Han, and G. Li, "Gradient band-based adversarial training for generalized attack immunity of A3C path finding," Jul. 2018, *arXiv:1807.06752*.

[20] S. B. Thrun, "Efficient exploration in reinforcement learning," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-92-102, 1992.

[21] L. Choshen, L. Fox, and Y. Loewenstein, "DORA the explorer: Directed outreaching reinforcement action-selection," Apr. 2018, *arXiv:1804.04012*.

[22] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," 2016, *arXiv:1606.05312*.

KANTING CAI received the bachelor's degree from Anhui Polytechnic University, in 2018. He is currently pursuing the master's degree with the College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua, China. His research interests include deep reinforcement learning and adversarial learning.

XIANGBIN ZHU received the B.S. degree from the Power Engineering Department, Southeast University, Nanjing, in 1993, the M.S. degree from the Department of Computer Science and Engineering, Zhejiang University, Hangzhou, in 2001, and the Ph.D. degree from the Computer Science Department, Fudan University, Shanghai, in 2005. He is currently an Associate Professor with the Department of Computer Science and Technology, Zhejiang Normal University, Jinhua, China. His research interests include multi-agent systems and deep reinforcement learning.

ZHAO-LONG HU received the B.S. degree from the Department of Physics and Computer Science, Xingtai University, China, in 2011, the M.S. degree from the College of Management, University of Shanghai for Science and Technology, Shanghai, in 2014, and the Ph.D. degree from the School of System Science, Beijing Normal University, Beijing, in 2017. He is currently an Associate Professor with the Department of Mathematise and Computer Science, Zhejiang University, Jinhua, China. His research interests include source localization in complex networks and optimization theory.